WIDE AREA TECHNIQUES

IN A LOCAL AREA NETWORK

A thesis presented by

MICHAEL MURPHY

in partial fulfilment of the requirements

for the degree of

Master of Science in Computer Science

at

Massey University

February 1986

ACKNOWLEDGEMENTS

The production of this thesis has been made possible by the co-operation and efforts of many people.

My sincere thanks go to my two supervisors, Mr Tony McGregor and Mr Paul Lyons, for their guidance during the development of MASSEYNET and for their helpful criticisms in the presentation of this thesis.

I would like to extend my special thanks to Mr John Anson for his encouragement and assistance, particularly during the implementation and testing stages of this research.

My thanks also go to the many members of the Computer Science Department who have willingly assisted in the testing of MASSEYNET and production of this thesis.

Finally, a very special thank you to my family for their support, patience and love during my two years in New Zealand. For this, I will always be grateful.

Michael Murphy

CONTENTS

LIST OF FIGURES

ABSTRACT

Packet switching networks have been developed on a large scale (national or even international) as a means for enabling inter-computer communication. They spread the responsibility for data handling amongst a number of "nodes".

In contrast, local area networks are restricted to a single site or organisation, and depend on the reliable operation of a central facility (sometimes a computer, sometimes a shared communication cable).

This thesis describes the design and successful implementation of MASSEYNET, a local area network embodying wide area techniques. The aim of the network is to provide a cheap system which will permit communication between devices located throughout Massey University, and permit individual terminals to interface to any one of the attached devices.

CHAPTER 1

INTRODUCTION

1.1 Local Area Networks

In the early years of computing, the long term vision was of a single large computer serving all of an organisation's computational needs. This model is rapidly becoming obsolete, making way for systems of smaller, interconnected computers. The development of such systems is motivated by the existence of a large number of mini and micro-computers, and their need to communicate with each other and share central resources. These systems are called local area networks because they enable communication between devices within a small geographical area (a diameter of not more than a few kilometres [Tane.1981]). The short distances involved have made it economically feasible for such networks to use a central, high-speed cable to carry all the data within that site. Devices attach to the cable via interface message processors (IMPs) and communicate by following a common set of procedures called the communication protocol. Local area networks are usually classified by their shape or topology because it governs how each device attaches to the network and how information is passed amongst them. Commonly used topologies include the star, loop, ring and bus configurations.

1.1.1 Star Networks

Star networks involve a direct connection between each network device and a centralised computer. The computer at the centre of the star is generally used as a processing resource but may also be used as an IMP to switch messages from one network line to another.

A common example is a minicomputer installation where terminals, printers, disk drives, and other peripherals are directly connected to the main computer (as shown in figure 1.1). The information exchange (communication protocol) is generally interrupt-driven but, in some cases the central processor may use round-robin polling to determine if each device has any information to send.

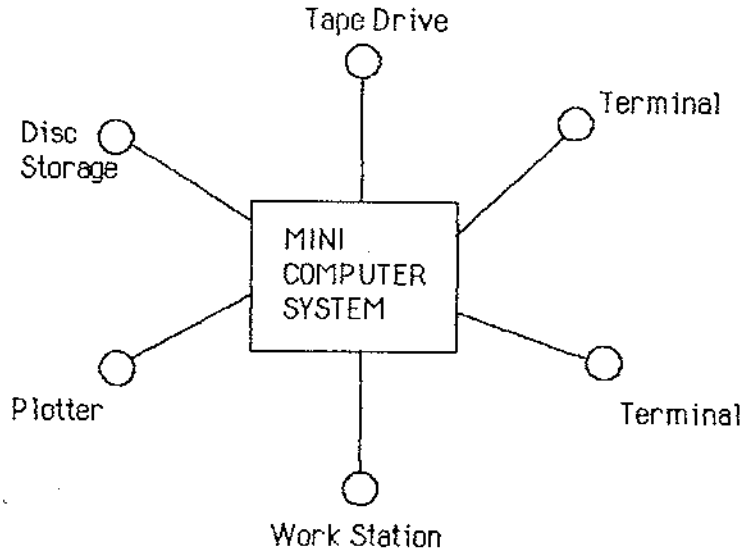


Figure 1.1 Star Network - Minicomputer Installation

The shared resource in star networks is the central computer not the transmission medium as with typical local area networks.

1.1.2 Loop Networks

A loop network consists of a controlling device positioned at some point in a loop of cable. Network devices are attached to the cable at various points and share it for message transmission as shown in figure 1.2 [Gee.1982]. There are two communication protocols commonly used in loop networks.

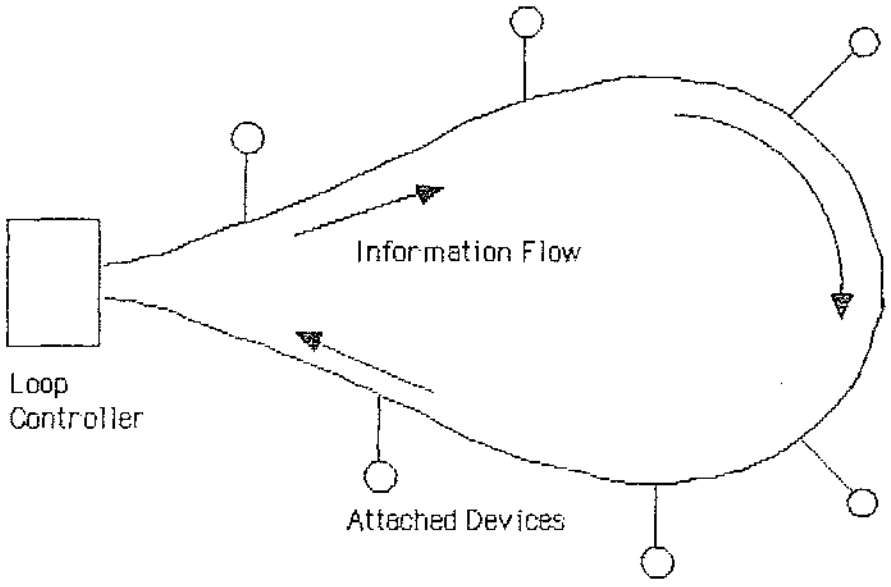


Figure 1.2 Loop Network

The first protocol requires the controlling device to poll each of the other devices by means of a specially addressed packet. A packet is a block of data which has a fixed structure and a fixed maximum size. The packet contains addressing information which specifies its destination, control information which includes the packet type, and a data field for transporting information between network users. If the polled device has data to send, it fills that packet with data and places the name of the required destination into the packet's address field. The controlling device is then responsible for ensuring the safe delivery of this data packet to the destination device. Normally, each of the network devices is polled in a strict, cyclic sequence (round-robin polling) so that each device gets equal opportunity to use the line.

The round-robin polling technique can be inefficient because channel capacity is wasted polling devices which have no data to send. The second protocol eliminates the inefficiencies of polling idle devices by having a single empty packet circulating around the loop available to all devices. Devices communicate as above by filling this empty packet with data and placing the required destination name in the packet's address field. The packet is passed around the loop to the destination device where it is emptied and placed back in the loop ready for re-use. With this latter technique, the controlling node must ensure that a single device does

not "hog" the network resources.

1.1.3 Ring Networks

Ring networks consist of a series of point-to-point cables linking adjacent nodes (IMPs) in a circular arrangement (figure 1.3) [Gee.1982]. Point-to-point cables provide a direct link between the devices attached at either end but do not allow devices to tap onto the cable at any point in between. Each node in the ring has equal status and there is no controlling device as in loop networks (although a special node is sometimes used to monitor the ring traffic and recover from erroneous transmissions). The communication protocol is responsible for ensuring that each node gets its fair share of the available network capacity. Three common communication protocols, described below, are empty slot, token rings, and register insertion.

Empty Slot

The empty slot technique involves one or more packets circulating around the ring. These packets may be either "available" or "in use" as indicated by a flag in the packet header. When a node has data to transmit, it waits for an "available" packet to arrive, fills that packet with data, and inserts the address of the destination node into the

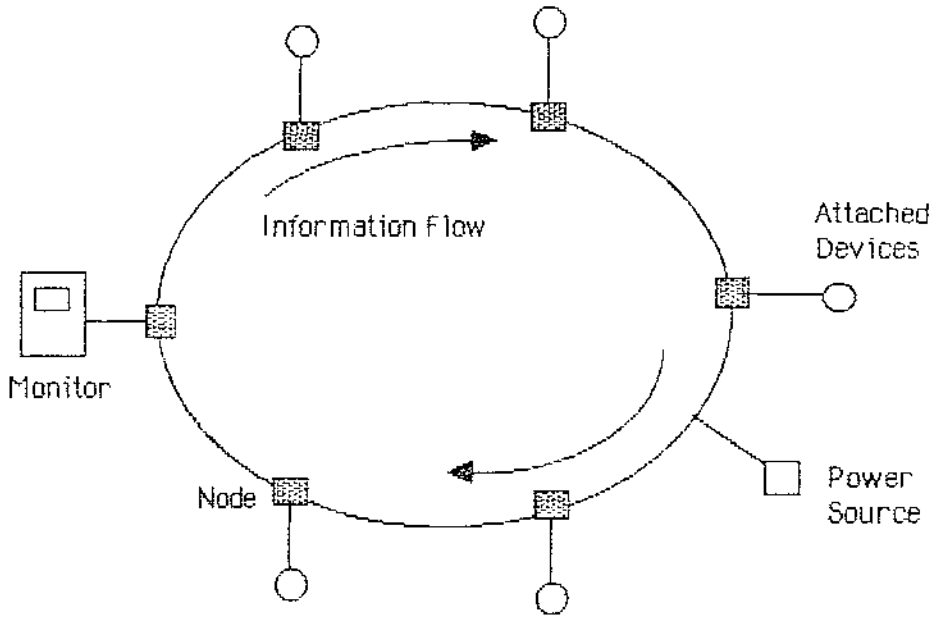


Figure 1.3 Ring Network

packet's address field. The packet is marked as being "in use" and passes around the ring from one node to the next until it reaches the destination. This node reads the data and marks the packet as being "received". The packet then circulates back to the originating node where it is marked as being "available" [Blea.1982]. Usually, the same node is not permitted to reuse the packet immediately, since this would enable a single node to "hog" the network resources. This communication sequence is illustrated in figure 1.4.

A monitor is often used in empty slot rings to spot and correct defective packets. This monitor is different from the controlling device used in loop networks because it does

not control the communication protocol. Its primary purpose is to detect and correct corrupted packets but it may also be used to collect statistics on ring traffic.

The empty slot ring is often refered to as the Cambridge Ring because the technique was first developed at the Cambridge University Computer Laboratory [Blea.1982].

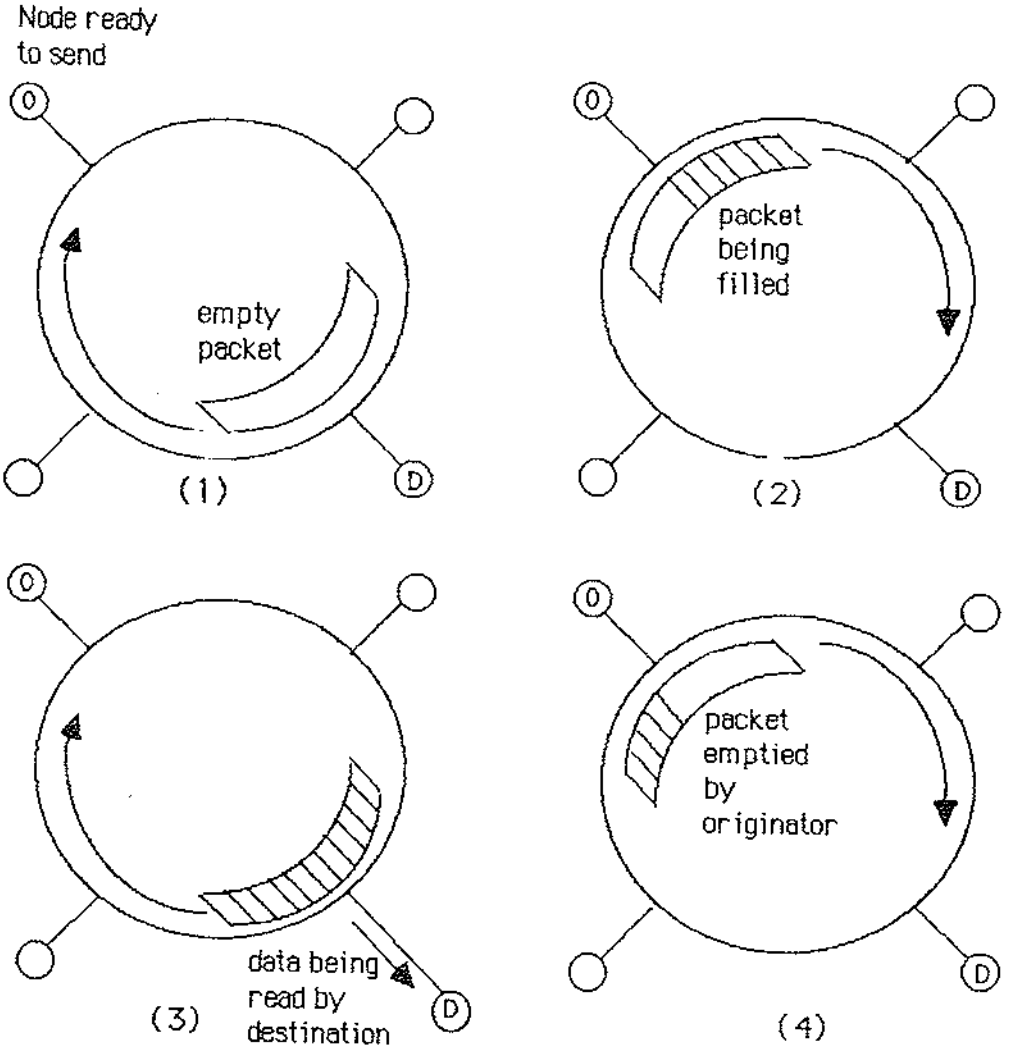


Figure 1.4 Operation Of An Empty Slot Ring

Token Passing

The token passing communication protocol is similar to the empty slot technique because a special bit pattern called a token circulates around the ring from one node to the next. Any node wishing to transmit must wait until the token is passed to it by the previous node in the ring. That node then removes the token temporarily and begins transmitting its own data. Having completed transmission or after using its maximum time allocation, the node places the token back into the ring and transmits it. The data received by the next node thus consists of one or more packets followed by the token. If that node also wishes to transmit, it must first of all relay the circulating packets to the next node, and then transmit as described for the previous node. The resulting situation after two nodes have each transmitted one packet of data is shown in figure 1.5(c). This string of messages makes a complete revolution of the ring. As each packet passes its destination, that node reads in the packet's data and acknowledges it as being received (by setting a flag in the packet's control field). When the packet returns to its origin node, it is removed from the ring. If no errors have occurred, the packet to be removed should be the first one in the returned string as shown in figure 1.5(e) [Gee.1982].

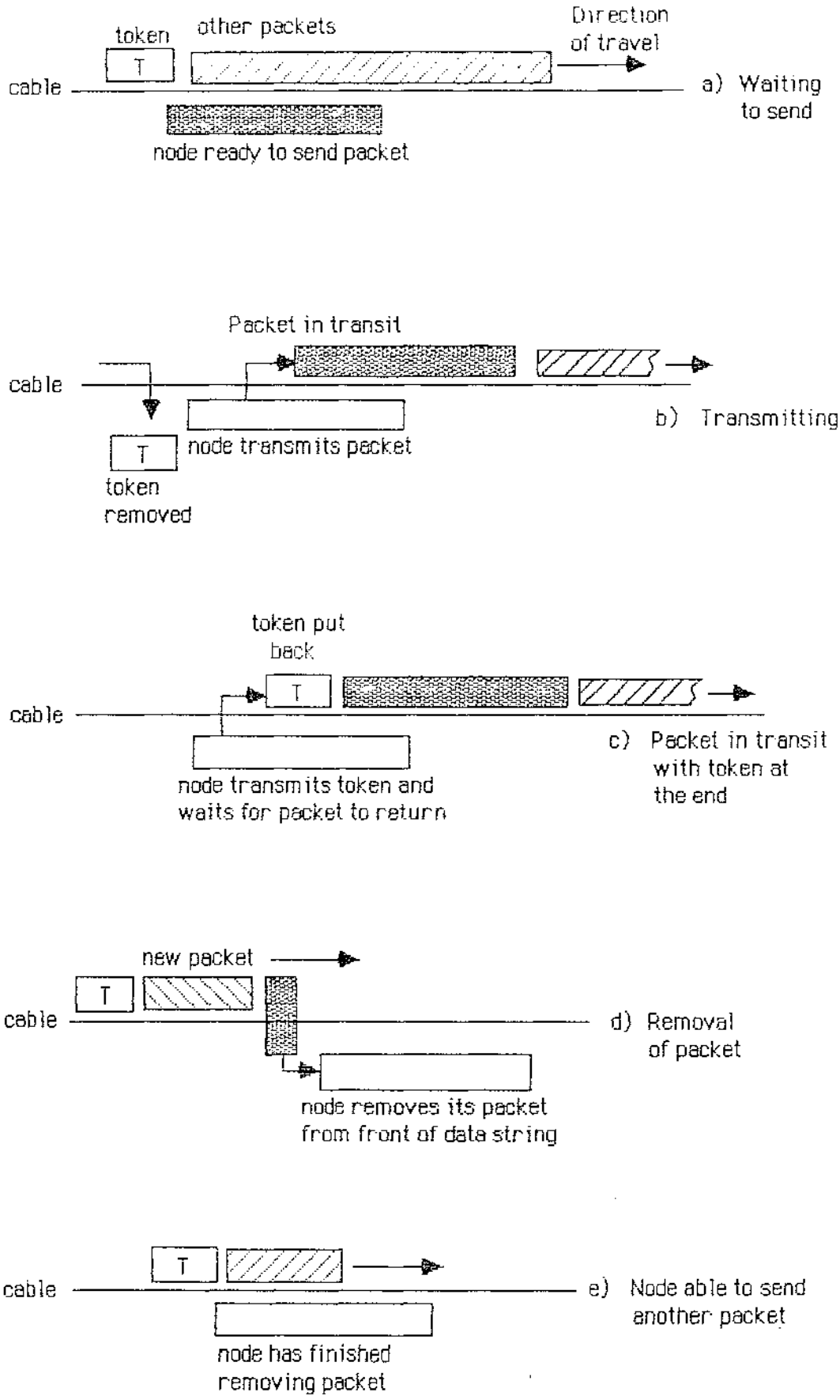


Figure 1.5 Token Passing

Register Insertion

The register insertion protocol operates with multiple packets on the ring. Each node has two registers (i.e. buffers) which can be switched into and out of the ring as required. The first buffer called the shift register, is used to store, check and forward information passing around the ring. If the check finds that the packet is destined for that node, then the packet is diverted to the associated device and removed from the ring. Otherwise, the packet continues its journey via the next node.

When a node has a packet of data to send, it stores it in the output buffer. (Packets may be variable in length up to the size of this buffer). At the end of each transit packet, the shift register checks to see if there is a packet awaiting output and if the number of empty slots in the shift register is at least as large as that output packet. If both these conditions hold, the output buffer is switched into the ring and its data is emptied. During this time, new input is stored in the shift register. The second condition guarantees that there will be enough room in the shift register to accomodate the maximum amount of input data arriving during the output process. At the end of output, the shift register is switched back into the ring and emptied as before. The ring interface which controls the switching between the shift register and the output register is

illustrated in figure 1.6 [Tane.1981].

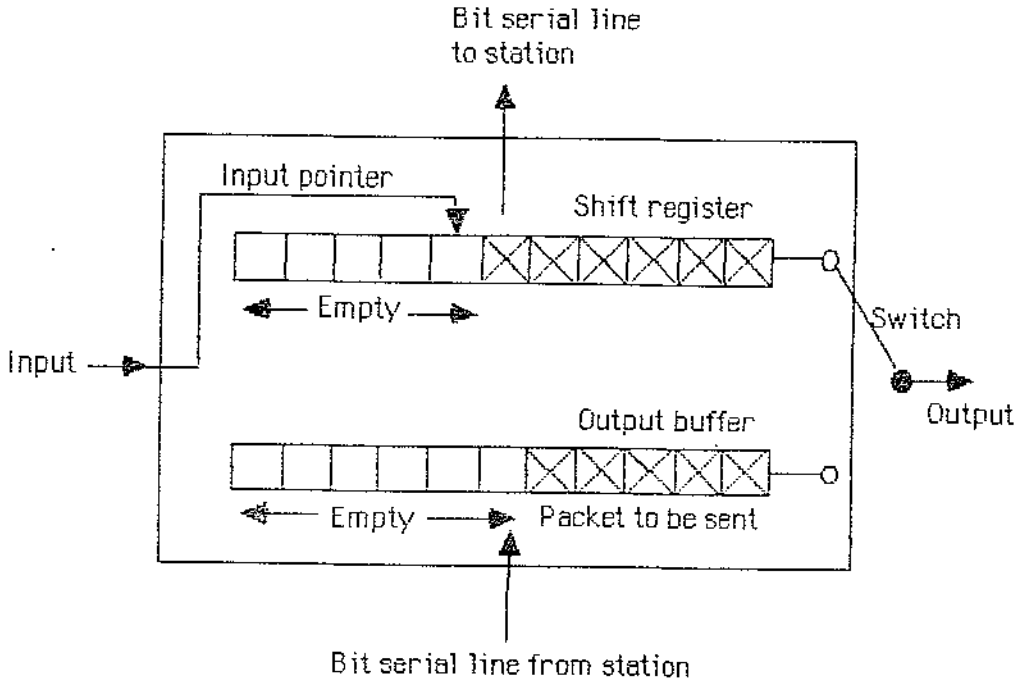


Figure 1.6 Register Insertion Ring Interface

1.1.4 Bus Networks

Bus networks consist of a central, linear channel which transports information between devices attached to it. (A channel is logically equivalent to a cable). Nearly all computer systems have a built-in bus network for connecting various components such as the processor, the memory, and peripheral controllers. Since many devices share the one transmission medium, an important issue is the allocation and sharing of that channel.

Ethernet

Ethernet is a well known bus network developed at the Palo Alto Research Centre of the Xerox Corporation in California during the 1970's [Gee.1982]. It was originally an experimental network but proved so successful that it has been installed in many business organisations and universities particularly in the United States. The main design objective of Ethernet focussed on the office environment which required a cheap, reliable network with the capacity for steady growth and easy handling of bursty data traffic.

Ethernet uses a high-speed co-axial cable as its central bus channel. Devices tap onto the co-axial cable through an IMP and communicate by "broadcasting" messages to all other IMPs on that channel (figure 1.7). Each IMP "listens" to the broadcasts extracting data for itself while ignoring everything else. There is no automatic acknowledgement of packet receipt. If this is required, it must be added explicitly by the system/program using the network.

The transmission protocol used by Ethernet is called Carrier Sense Multiple Access with Collision Detect (CSMA/CD). The "carrier sense" phrase is more easily understood as "listen before transmitting" while the "multiple access" phrase

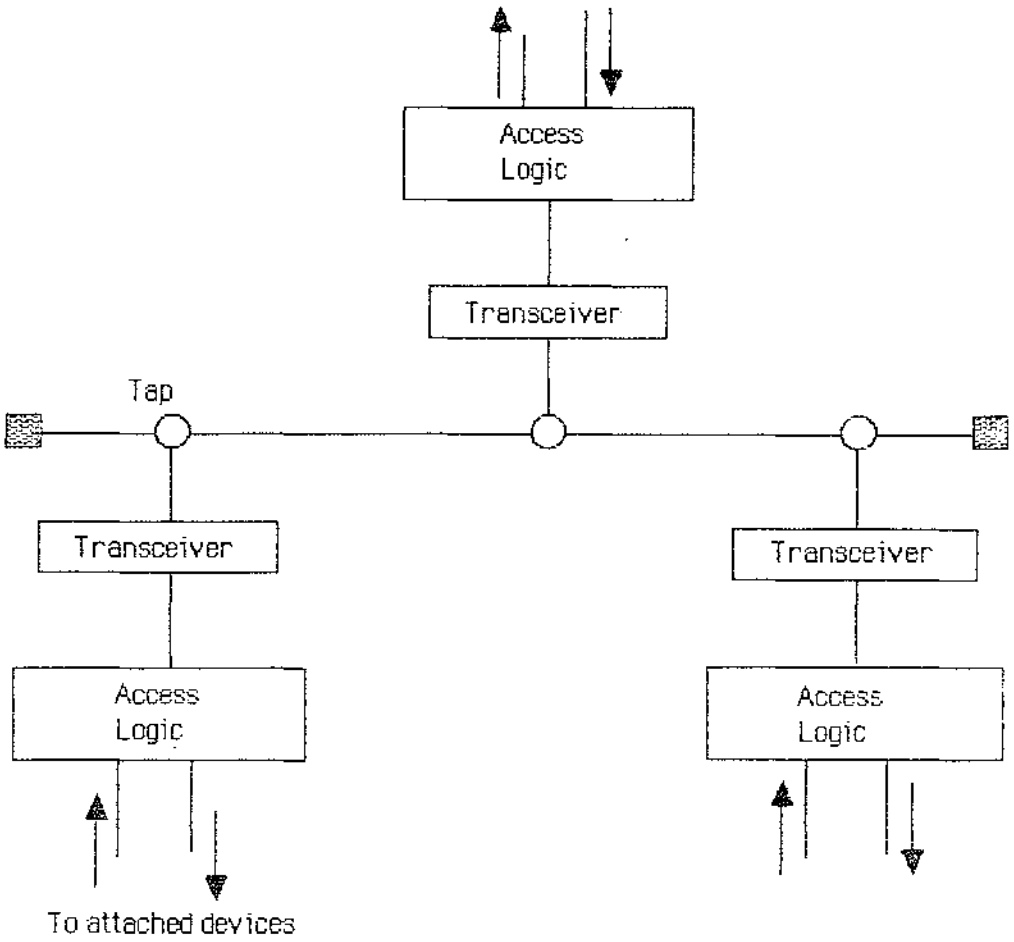


Figure 1.7 Ethernet Bus Configuration

simply means that there are several devices sharing the one communication medium. So, with only a CSMA network, any device wishing to transmit must first of all "listen" to determine if any other device is currently using the network. If so, the device must wait until the network is free. This "listen before transmitting" rule greatly reduces but does not eliminate the chance of colliding packets. Collisions will still occur if two or more IMPs transmit simultaneously.

The "collision detect" phrase can be reworded as "listen while transmitting"; i.e. once an IMP starts transmitting, it must continue listening to the channel. If it detects a collision, it must cease transmission immediately and wait a random period of time before trying to retransmit. Ceasing transmission immediately upon collision detection minimises the amount of channel bandwidth (channel capacity) which is wasted. Bass [Bass] suggests that effective bandwidths of 97% of the total bandwidth can be achieved with the CSMA/CD technique. In contrast, Petitpierre [Peti] states that "local area networks based upon a bus architecture, like Ethernet, suffer severe distance/throughput limitations related to the collision-detection mechanism." Unfortunately, Petitpierre does not support his claim with experimental data.

1.2 Wide Area Networks

Wide area networks transport information over large distances. In contrast to the freedom of local area networks, wide area networks rely heavily on existing public telecommunications facilities because it is too expensive to lay high-speed cables over large distances (it is also illegal in most countries). Since the topology of wide area networks is governed by the distribution of existing public facilities, they usually develop as an irregular arrangement of nodes connected in a point-to-point fashion by telecommunications cables. Communication involves the step-by-step transfer of information through any of the possible paths involving the intermediate nodes which separate the end users. Since the pathway taken has a large bearing on network performance, a lot of time has been spent analysing various routing algorithms for optimising this choice. In general, wide area networks have a fairly static configuration because they usually link stable entities such as different branches of a single organisation. This property is often used to reduce the complexity of the routing algorithm.

The dependence on existing telecommunications facilities introduces other problems including a greatly increased error rate. The error rate difference between a local area computer-to-computer connection and a dial-up telephone line

has been calculated as at least eleven orders of magnitude (base 10) [Tane.1981]. Consequently, designers of wide area networks must include quite sophisticated error checking and error recovery algorithms in the communication protocol.

Problems also arise due to the finite storage capacity of the intermediate nodes in a wide area network. Increased demands from one user group may saturate some network resource and degrade the quality of service provided to other users. This resource saturation is referred to as congestion. Congestion control procedures must be incorporated into the communication protocol to prevent this degradation of service.

Furthermore, if resources in each node are allocated ad-hoc upon demand, congestion may lead to deadlocks, in which messages can make no forward progress because the resources of two or more interdependent nodes have been exhausted (e.g. no more buffer storage). Each node is waiting on the other and so the network is locked in that situation forever. To prevent such deadlocks, the communication protocol must incorporate algorithms for controlling resource allocation.

1.2.1 Public Telephone Networks

The oldest and most familiar wide area network is the public telephone system. It uses a circuit switching approach to establish a temporary point-to-point connection between end users. The physical connection is established at dial-up time and is terminated at the end of the conversation. To establish the path, a hierarchical fixed numbering scheme is used. This routing scheme involves specifying from top-down the particular groups to which a subscriber belongs. E.g. the first four digits specify the subscriber's country, the next three specify the district, and the last five digits specify the particular line within that district. Procedures are provided to shorten this dialed number for subscribers belonging to common groups. Each link in the temporary circuit is dedicated (reserved solely) to that one conversation so that intermediate nodes (or switching exchanges) simply relay the input data to the corresponding output line. The nodes do not store or process the data. Since there is no resource sharing between different temporary circuits, congestion control and deadlock prevention procedures are not required.

1.2.2 Arpanet

Arpanet was developed in the late 1960's by the Advanced Research Projects Agency (ARPA) of the USA Department of Defence. It was the first wide area network to use the store-and-forward, packet switching technique [Blea.1982]. This technique differs markedly from a circuit switching approach because the sequence of point-to-point links is not dedicated to any one particular transmission. Instead, messages are broken up into fixed length packets and routed through the network one link at a time as circumstances allow. The intermediate nodes receive and store each packet in its entirety before forwarding it on towards its destination (hence the term store-and-forward). Since network resources are shared amongst temporary connections, the communication protocol must include congestion control and deadlock prevention mechanisms. Restricting messages to a fixed length simplifies these control procedures and also facilitates easier handling and queueing at intermediate nodes. Arpanet uses a distributed adaptive routing algorithm to determine the best path for transmitting packets through the network. This algorithm is described in Section 3.2.

In 1969, an experimental four-node network was set up [Tane.1981] and since then, Arpanet has grown to well over one hundred network nodes, providing communications around most of the world. Arpanet is of historical importance

because it was the pioneer of congestion control and deadlock mechanisms associated with store-and-forward networks. Indeed, Arpanet's communication protocols have formed the basis of many fundamental concepts in networking and data communications. Arpanet's design has been used in many subsequent networks including Canada's Telenet (the first public packet switched network), the UK Public Packet Switched Service, and the International Packet Switched Service [Blea.1982].

1.3 Merging Wide Area Techniques Into Local Area Environments

The local area networks discussed above depend on the reliable operation of a central facility; a central computer in the case of the star network, and a shared communications channel in the loop, ring and bus networks. As well as introducing problems of reliability, the central facility also leaves the network vulnerable to eavesdropping and computer hackers. The capacity of the central facility also places an upper limit on network expansion.

In contrast, wide area networks spread the responsibility for data-handling amongst a number of nodes. There is no need for a centralised facility and no practical limit on expandability. These properties would be ideal within a local area environment. Several attempts have been made to incorporate the wide area networking principles into a local area environment. Three such examples are Floodnet [Peti], Mininet [Neri.1984], and Localnet [Suns.1983]. Each of these networks was developed in response to the networking requirements of a specific environment.

Floodnet is an experimental network developed at the Lausarine Poltechnic School in Switzerland with the aim of providing a low-cost reliable and expandable system using only point-to-point links. Of major importance was the design of a routing technique which leads to very simple

nodes.

Mininet is a local area packet switching network developed at the Polytechnic of Central London in collaboration with the University of Bologna [Neri.1984]. It has as its objective "the interconnection of a heterogeneous assortment of laboratory monitoring and control devices, data acquisition equipment and minicomputer installations".

Localnet was designed and implemented by Sytek [Suns.1983] with the aim of "interconnecting multiple broadband networks to form an integrated packet transport system".

Floodnet, Mininet and Localnet each use communication techniques which have previously been associated only with wide area networks. The three networks are examined in more detail in Section 2.5 to determine the feasibility of using wide area techniques in a local area and to assess how easily each one can be adapted to other networking environments.

1.4 ISO OSI Reference Model

To reduce their design complexity, most networks are organised as a series of layers or levels. The layered approach was conceived when it was realised that the functions required for data communications can best be considered hierarchically. As data enters the network, it will be subject to a series of operations which will be reversed at the receiving end. Each layer offers certain services to higher layers while shielding them from the details of how those services are actually implemented.

The International Standards Organisation (ISO) has developed a reference model for what it terms Open Systems Interconnection as a first step towards international standardisation of the various protocols. This model consists of seven layers as illustrated in figure 1.8 [Tane.1981]. International standards exist for the four bottom layers but the layers 5 to 7 are still the subject of much research and experiment.

Layer 1, called the physical layer, is responsible for the transfer of raw bits from one end of a communication channel to other. The design decisions in this layer largely deal with the mechanical and electrical interface to the network; e.g. what voltage level should be used to represent bits, and what type of connector is used?

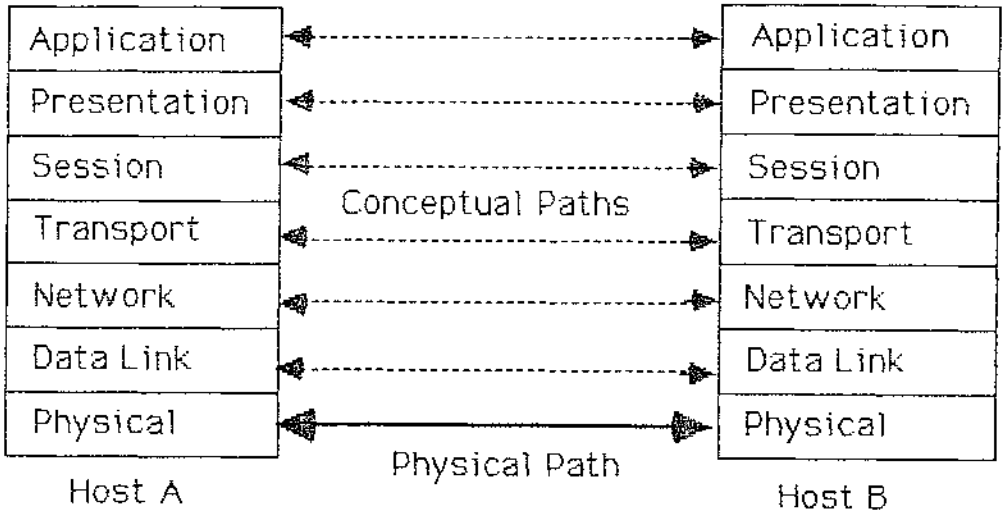


Figure 1.8 ISO OSI Seven-Layer Reference Model

The data link layer (layer 2) transforms the raw transmission facility into an error free channel. It achieves this by breaking the input data from layer 3 up into frames (blocks) and adding error checking information. Using this information, the receiver can detect errors and request retransmission of the relevant packet(s). The data link layer is also responsible for detecting lost or duplicate frames caused by hardware failure, noise bursts etc. To do this, some ordering is required to specify the sequential relationship between successive packets. Another issue handled by layer 2 is to prevent a fast transmitter from drowning a slow receiver of data. Some form of flow control must be used to let the sender know if buffer space is still

available at the receiving end.

The primary task of the network layer (layer 3) is to determine how the packets of information should be routed through the network. Its aim is to communicate data between any pair of nodes regardless of node/link failures. In addition, layer 3 must control the maximum number of packets in the network at any one time. If too many packets are present, the network will become congested and a degradation of network services will result.

The aim of the transport layer (layer 4) is to provide an error-free virtual circuit between communicating devices by introducing end-to-end controls. The lower three layers are handled by each node and its immediate neighbours, not by the ultimate source and destination devices. The transport layer provides end-to-end error control, packet resequencing and flow control to ensure that all pieces of data arrive safely at the destination. This layer is also used to provide a standard interface to the higher layers. This interface includes the packet assembly and disassembly functions (i.e. grouping input characters into packets, and extracting characters from packets ready for output).

The session layer (layer 5) is the user's interface into the network. Its function is to standardise procedures for initialising, controlling and terminating sessions

(connections) between end users. Initialisation of a session often involves using passwords for user identification and authentication. Management of the session once it has been established is achieved by adding application-oriented functions to the communication service offered by the transport layer. For example, it may be crucial that data base transactions are never aborted halfway since this would result in inconsistencies in the data. In such a case, it is the responsibility of the session layer to provide a mechanism for delivering all the messages in a single transaction as a complete unit. If one of the messages is missing, the session layer should not deliver the partial transaction.

The presentation layer (layer 6) provides functions which are used often enough to warrant a general solution to them; for example, transformation services may be provided to convert between different file formats. Another common function provided at this layer is text compression. Text compression supports the use of short codes to replace certain words or phrases which are especially common in a particular application (e.g. the words "withdrawal" and "deposit" in a banking system).

The seventh layer (called the application layer) is controlled by the users themselves. It defines the set of allowed messages and the action taken on their receipt.

Access to network wide data bases can also be controlled at this level.

1.5 Objectives Of This Project

This thesis examines the use of wide area networking principles in the local area environment. It then describes the design, implementation, and testing of MASSEYNET, a local area network embodying wide area techniques. Of foremost importance was the development of a network for establishing, controlling and terminating links between network users (i.e. corresponding to layers 1 to 4 of the ISO reference model). Since the upper three layers provide separate services, it was recognised that the design and implementation of these levels could be added at a later date. Until then, layers 5, 6 and 7 can be bridged by a simple user interface which provides two basic facilities; one for establishing and one for terminating connections.

This research into computer networks was stimulated by the networking requirements of the interactive computing environment at Massey University (described in Section 2.3). While special attention has been paid to the demands imposed upon a network by this environment, it was important that the resulting system be flexible enough for adaptation to other areas.

The use of wide area networking techniques in the local area environment was seen as a possible solution to the requirement for a cheap, reliable system. In addition,

Z80-based micro-computers suitable for use as network nodes were available from within the university at a cost of $240. Since the Z80 is a low-speed processor, it was decided that network implementation should be guided by efficiency rather than the (often inefficient) layered structure of the ISO reference model. While the ISO model was not discarded altogether, it was used as a design guideline, not as an implementation constraint.

Chapter 2 describes the interactive computing environment within Massey University and outlines its networking requirements. It then examines how well the local area networks described in this Chapter meet these needs. Finally, Chapter 2 looks at the wide area networking principles used in both the public telephone network and in Arpanet. In doing so, it explains why the packet switching approach (as used in Arpanet) is better suited to an interactive environment than the circuit switching approach used in the telephone network.

Chapter 3 takes a detailed look at the general packet switching approach and outlines the basic design considerations and decisions associated with using such a technique.

Chapter 4 uses this general overview of packet switching to describe the iterative implementation of MASSEYNET.

Chapter 5 looks at the results achieved during the testing of MASSEYNET in several different network configurations.

Chapter 6 goes on to propose further developments to the existing system.

CHAPTER 2

BASIC DESIGN CONSIDERATIONS AND DECISIONS

As illustrated in Chapter 1, there are many existing types of networks covering a wide range of applications. Before looking at these networks as possible solutions to Massey's interactive computing requirements, it is helpful to analyse and define the needs of the environment in which it will be used. Within Massey University, the most varied and concentrated range of computing facilities exists within the Computer Science Department. Consequently, a network which satisfies the needs of this department would form a strong foundation for a University-wide network (providing it can be expanded). The design, implementation and testing of MASSEYNET has therefore focussed on the networking requirements of the Computer Science Department at Massey.

2.1 Resources and Topology

The Department has a variety of computer equipment ranging from stand-alone micro-computers such as the ICL Personal Computer and Altos 68000 through to intelligent work-stations such as the Apple Macintosh. It has several types of terminals which interface via manual exchanges (patch panels) to the three mini-computers operated by the University Computer Centre - a Prime 9955 (called Sysb), and 2 Vax

11/750's used for research and teaching. The Computer Centre operates a second Prime 750 (Sysa) which can be accessed through Sysb using Primenet (the Prime network). The Department has 24 lines to each of the Vax machines but only 2 lines to the Sysb Prime computer. All lines support RS232 communication and operate at 9600 baud.

The Department's system of three interconnected patch panels supports RS232 communication. Each patch panel forms the central node of a star network which connects Departmental offices and laboratories with the computer resources described above (see figure 2.1) [Lyon.1985]. This system of patch panels enables communication between any two devices but can be unwieldy to use. Contention for lines involves a trip to the patch panel, finding a spare line, and then manually patching the sought-after device through to the correct office. If all the lines to the sought-after device are patched to other offices, further trips to those offices must be made to determine if all the lines are actually being used. Another drawback of the patch panel system is its star configuration. This configuration requires extensive lengths of cabling because each Department office must be connected directly with one of the patch panels. The long lengths of cabling needed means that this network configuration is not conducive to short term connection requirements (e.g. "line for a day" type requirements).

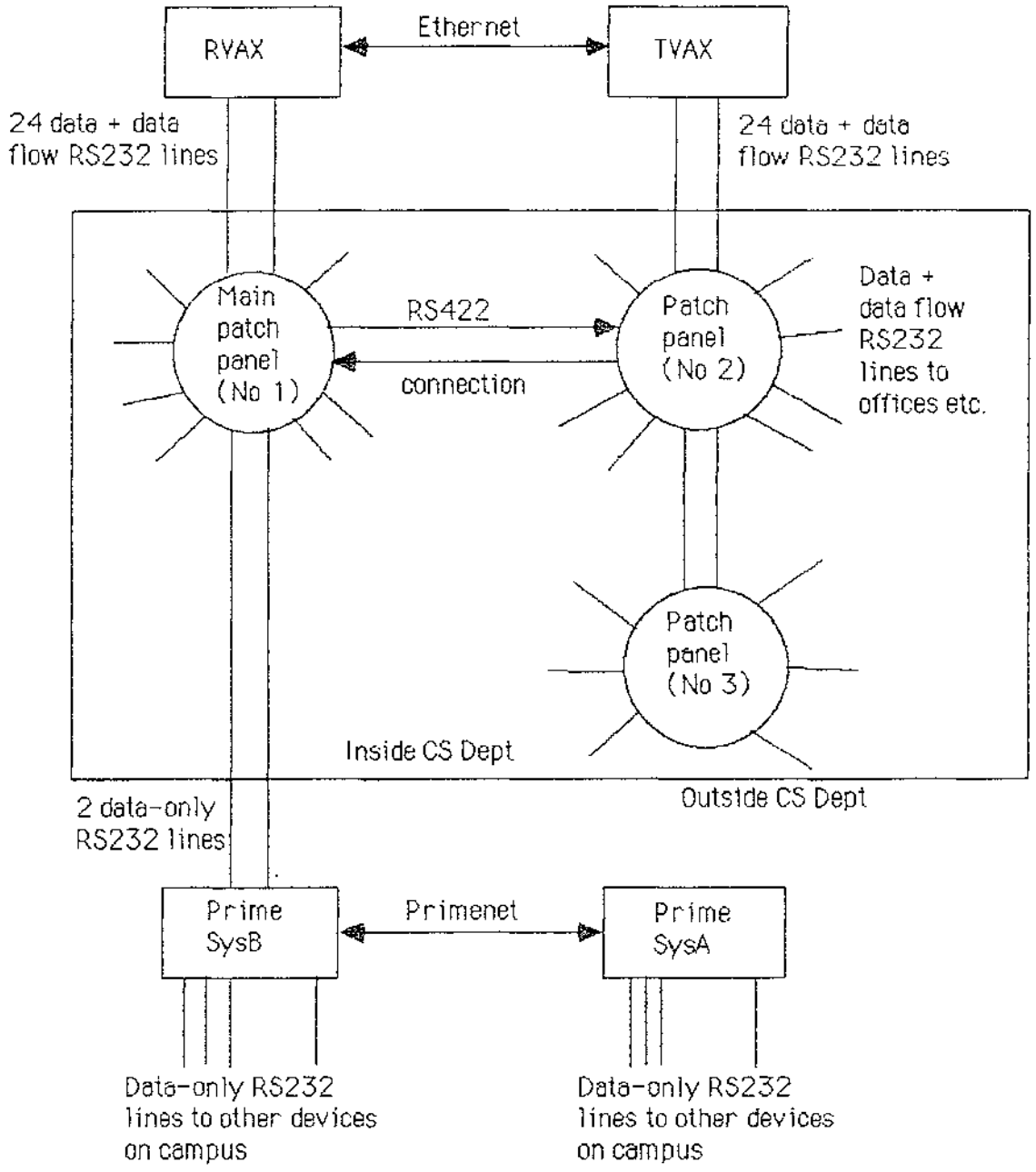


Figure 2.1 Computer Resources Within Massey University Computer Science Department

2.2 Traffic Characteristics

In the design of any type of network, an important consideration is the nature of the traffic (flow of data) it must support. Typically, interactive computing involves sporadic use of the available line resources. The human involvement results in long periods of "silence" (while the user digests the information on the screen) punctuated by commands which result in large data transfers. In most cases, interactive computing results in very low utilisation of the available line bandwidth. This low utilisation suggests that a network should exhibit a high degree of concurrency or multiplexing capabilities so that more efficient use of line resources can be made.

Another important traffic characteristic is the type of message exchange which occurs between the user and the computer (e.g. single character interactions, short word exchanges). If this is known, then the network can be "tuned" to the requirements of that particular message type; for example, in packet switching networks, the most commonly exchanged message size influences the choice of packet size. To determine this traffic characteristic in an interactive environment, a direct line between a regular interactive user and the Prime computer was analysed for a period of one week. The exchange of messages was analysed by keeping cumulative tallies of message lengths in both directions. Successive

characters were considered to be one message if the time difference between them was less than 0.002 seconds. This time interval was chosen because, at 9600 baud, asynchronous characters are separated by fractionally more than 0.001 seconds. The results are plotted in figure 2.2 (a) and (b).

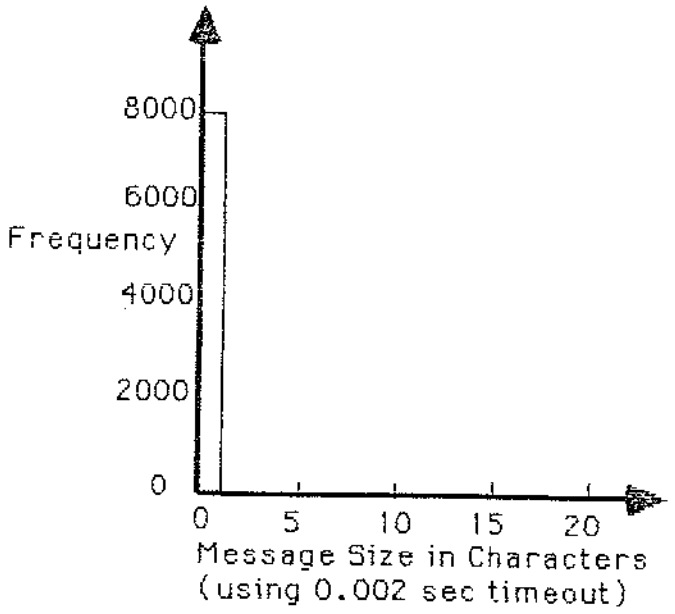


Figure 2.2 a) Graph Of Message Size Versus Frequency For Terminal-to-computer Traffic

The graph for terminal-to-computer traffic (figure 2.2(a)) shows just one peak corresponding to single character messages. This indicates that successive characters entered by the terminal users were always spaced at least 0.002 seconds apart. This makes sense because we would expect it to be humanly impossible for a typist to enter characters at 9600 baud (960 characters/sec).

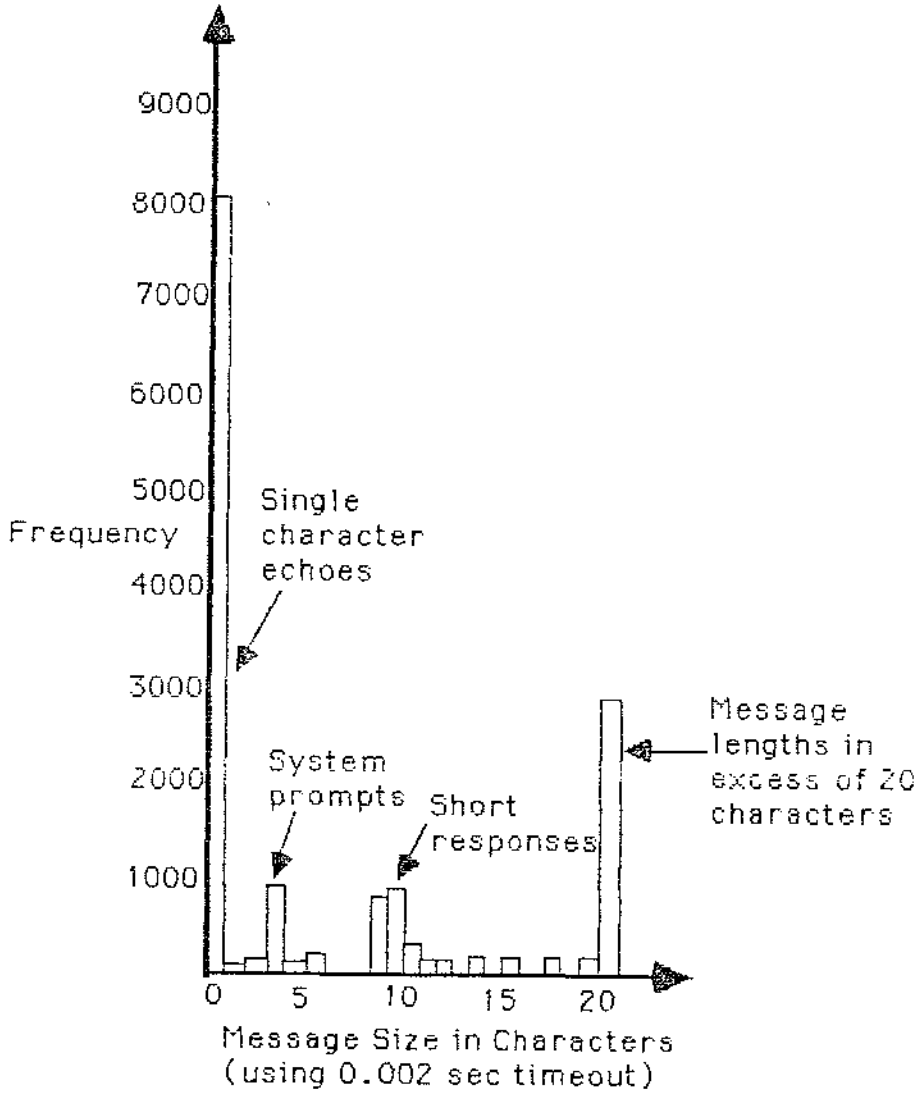


Figure 2.2 b) Graph Of Message Size Versus Frequency For Computer-to-terminal Traffic.

The statistics for computer-to-terminal traffic (figure 2.2(b)) show large peaks corresponding to single character and long messages, with smaller peaks at message sizes of 4 and 10 characters. The single character peak is due to the echoing of characters as they are typed at the terminal and it corresponds to the single character peak on the

terminal-to-computer graph. The shorter peak at 4 characters corresponds to system prompts ("OK, " on the Prime computer) while the 10 character peak corresponds to short system responses such as error messages and dates. The longer messages result from screenfuls of data sent in response to various editing commands, file listings, and descriptions of various operating commands.

In addition to the common interactive editing and operating system commands, people with intelligent work-stations transfer files to the mini-computers. This type of work results in very high line utilisation for a short period of time.

In an interactive environment therefore, a network should be able to support a high proportion of single character messages and an equally high proportion of full-screen messages. Unfortunately, the requirements of these two message types conflict. Single character messages require low delay through the network to ensure quick response times. Longer messages require high throughput so that large volumes of data can be transferred as quickly as possible. As will be shown later, low delay and high throughput are conflicting properties. Hence, a compromise between the two is necessary.

2.3 Network Objectives

This analysis of resources, topology and traffic characteristics highlights the networking requirements of an interactive environment. To satisfy the demands of this environment, a network should aim to meet the following objectives :-

(a) It should allow network-transparent communication between any two devices attached to the network. Any interactive session using a connection through the network should appear to the user as if the terminal was hardwired to the computer, thus allowing full use of facilities such as full-screen editors and graphics packages. To enable this, response times through the network must be minimised.

(b) It should permit individual terminals to interface easily to personal computers at various locations throughout a building and to any centralised computing resources such as the mini-computers operated by Computer Centre. The network should provide the convenience of a fully automated patch panel system so that any two devices can be connected through the network.

(c) It should allow sharing of resources (e.g. file servers and printers). Resource sharing between work-stations, between various personal computers, and even between the mini-computers eliminates the need for resource duplication thus saving money and space.

(d) It should provide more efficient use of and easier access to scarce resources such as the Computer Science Department's two lines to the Prime computer. Ideally, the network should have a high-level user interface which queues connection requests for a particular device if it is busy. When that device becomes available, people waiting on the queue are given the opportunity to "acquire" that resource in the order in which their requests arrived. As well, the network should support a high degree of concurrency to provide more efficient use of line resources.

(e) It should be reliable with graceful degradation upon failure. Networks which depend on a central resource are rendered inoperable whenever that resource fails. Ideally, a network should be a system of interconnected but autonomous units so that failure of any one unit should at worst lead to a small degradation in network performance.

(f) It should provide sufficient security for the needs of network users. This objective is hard to quantify. The manual patch panel system currently used by the Computer Science Department is very insecure because lines can be disconnected in the middle of a session. This can be the result of deliberate "poaching" of a line, or merely of an accidental knock on the plug of one of the patch cables. An automated patch panel system (or network) would eliminate the need for people to use the patch panel room and so it could be locked thus providing a significant improvement in security.

(g) It should be flexible enough to allow incremental growth and indefinite expandability. Interactive environments tend to be highly dynamic because of the ever-changing needs of the people using them. Hence, a network must be able to expand, adapt, and contract to meet these needs.

(h) It should be cost-effective. This criterion is difficult to quantify but ideally, the cost of connecting a device to the network should be a small fraction of the actual cost of that device (say less than 10%).

2.4 Suitability of Existing Local Area Networks

As mentioned in Section 1.1.1, star configurations are usually adopted when a central processor is to be shared amongst several users. The central processor allocates time slices to each of the attached devices so that each one appears to be connected to a dedicated, stand-alone computer. The mini-computers operated by Massey's Computer Centre (described in Section 2.1) are typical examples. Such systems can provide communication between attached devices and allow the sharing of expensive resources such as printers and disk storage.

Loop networks also provide resource sharing and network-transparent communication between any of the attached devices. Furthermore, they facilitate more efficient use of the central channel bandwidth. The loop configuration has been in existance for some time but has really only been used to connect a small number of low-usage terminals to a computer [Gee.1982].

Ring networks have been researched extensively and have been implemented in many environments. Perhaps the best known implementation is the Cambridge Ring network which is over 1km long with 36 nodes (or stations) attached to the 10Mbps central channel [Gee.1982]. The ring has proved very reliable in operation and satisfies the first four objectives

outlined in the previous section.

Ethernet is possibly the most successful of the existing local area networks. It has been implemented in several University environments in the United States. At Stanford University for example, the Ethernet connects 200 computers (100 VAX's and 100 workstations) [Hayw.1985]. In all, 2000 terminals are connected to these computers in star configurations (hence introducing the problems of flexibilty, security and reliability associated with this topology). The Ethernet is used primarily for the transfer of large messages or files between computers. Most of the interactive communication occurs between the terminals and the computer to which they are attached.

Ethernet, star, loop, and ring networks depend on the reliable operation of a central resource - a central computer in the case of a star network and a shared transmission medium in the case of the other three. Although these networks can be rendered reliable by the use of (expensive) resources with a low failure rate, they do not degrade gracefully when the resources do fail (objective (e)).

The dependence on a central resource also poses the problems of privacy and vulnerability to attack (c.f. objective (f)). A sufficiently skilled and knowledgeable attacker can gain control of the whole network simply by controlling the single central resource. The most frequent forms of attack are spurious message injection, unauthorised message reception (eavesdropping), transmission disruption and re-routing of data to fake nodes [DeMi.1983]. Encryption (special coding) techniques are often used to increase the difficulty of such attacks [DeMi.1983]. The most severe type of attack is sabotage or crippling of the central resource. Such an attack results in the failure of the whole network. Fortunately, this type of attack is relatively rare in the University environment.

Star, loop and ring networks have a rigidly defined topology. This often requires large amounts of extra cabling to connect each device to the central computer (in the case of star networks) or to link devices in a circularly closed fashion (as with loop and ring networks). Although the Ethernet is more flexible, the cost of the central co-axial cable usually prevents it from attaching directly to each particular device. The usual approach is to lay a shorter length of co-axial cable in a central location and then lay cheaper cabling between each device and the co-axial cable (as in star networks).

Another drawback of using a shared central facility is cost. To facilitate sharing between several devices, the central resource must have sufficient capacity to handle the demands placed on it during peak times. This means using a high-speed processor in the case of star networks and a high-speed transmission cable in the case of Ethernet, ring and loop networks. The costs involved are fairly high; for example, the co-axial cable used in Ethernet and many ring networks costs between $NZ8 and $NZ10 per metre. Another major expense is the cost of connecting devices to the central resource. With Ethernet, for example, the connection cost per terminal operating at speeds up to 19.2kbps is in the range $US350 and $US500 [Reag.1984].

With these four networks, the size of the central resource places an upper limit on network expansion. The expandability of Ethernet is also restricted by the use of the carrier-sense protocol (CSMA/CD), as this protocol relies upon each node hearing the transmission of any other node within one packet transmission time. For a typical one kilometre co-axial cable, the worst case situation results in a node detecting a collision 10 micro-seconds after it started transmitting its own packet [Tane.1981]. For the CSMA/CD protocol to work in this situation, the transmission time for a single packet must exceed 10 micro-seconds. Hence, the longer the co-axial cable, the longer the packet needs to be.

From this analysis, it seems that Ethernet, star, loop and ring networks do not satisfy the last four networking objectives outlined in the previous section. The use of a central facility in each case leads to problems with reliability, security, flexibility and cost. Expandability is also limited by the size of the central resource installed, and by the CSMA/CD communication protocol in the case of Ethernet.

2.5 Suitability of Floodnet, Mininet and Localnet

2.5.1 Floodnet

As mentioned in Section 1.3, Floodnet was developed in Switzerland with the aim of providing a "low cost design for a reliable and expandable system with high throughput and fibre-optic compatible implementation using only point to point links" [Peti]. It was also important to design a routing algorithm which leads to very simple nodes.

The routing algorithm used in Floodnet is a variation of the flooding process described in Section 3.2. Instead of transmitting packets of data to all parts of the network however, Floodnet floods line signals in search of the required destination. In response to this flood, a full-duplex path is set up for the duration of a single packet transmission. During this time, each node along the path is dedicated to that one connection (as in circuit switching).

The resulting network has been successfully implemented and tested in a local area environment. It is a very simple and flexible network in which control is completely distributed, without complex recovery mechanisms.

Perhaps the major drawback of Floodnet relates to the dedicated path established temporarily for each packet. Since each node along the path is reserved solely for the one connection, no node can support two different connections simultaneously. This is a major limitation in an interactive environment because a high degree of concurrency is needed to make use of scarce resources (objective (d)) and allow many connections at the one time. If Floodnet was used in such an environment, then the centralised nodes would create a bottleneck if several interactive users placed heavy demands on network resources at the same time. This situation would result in unacceptably high response times.

It seems therefore that, although the flexibility of Floodnet is very attractive, its single connection limitation makes it unsuitable for an interactive environment.

2.5.2 Mininet

Mininet was developed for real-time instrumentation purposes but could easily be used in an interactive type environment. Its routing algorithm acquires its information dynamically and so the network handles initialisation, extension and contraction automatically.

The stations (nodes) in Mininet can support up to 64 devices. This large number of connections per station detracts from the network's flexibility and potential for incremental growth. Within Massey's Computer Science Department, for example, two of these stations could connect all the existing devices. The resulting network would be equivalent to two interconnected star networks (an automated version of the current patch panel system). Network expansion could only proceed in large quantities to justify the purchase of another 64-node station. Furthermore, each new device added to the network would need to be connected via a dedicated cable to one of the two stations (a process which might require extensive cabling). As well as these drawbacks, the system would exhibit the problems of reliability and security normally associated with star configurations (as discussed previously).

Hence, although Mininet could be used successfully in an interactive environment, the resulting network topology introduces drawbacks which conflict with the previously outlined objectives.

2.5.3 Localnet

Localnet was developed for the interconnection of multiple broadband networks and so is not directly applicable as a single interactive networking solution. Of particular interest though is the "Discovery" algorithm used to establish a connection between any two devices.

The Discovery process was developed in response to the session-oriented nature of communications in broadband networks. This type of behaviour is also prevalent in interactive environments where a session is defined as the time between a user logging onto and logging off a particular computer. The Discovery process uses the flooding algorithm at the beginning of a session to establish the best virtual (logical) circuit to the required destination. Subsequently, all communication in that session follows that initial virtual circuit. This technique displays the flexibility and expandability of the pure flooding approach but avoids the congestion associated with flooding every packet. In addition, the non-hierarchical addressing scheme allows a device to be accessed without knowing its physical location. This type of routing scheme would go a long way towards meeting the objectives of an interactive environment.

2.6 Suitability of Wide Area Networks

From the discussion in Section 2.4, it seems that the principle drawbacks of existing local area networks are related to their dependence on a single, centralised facility.

In wide area networks however, the responsibility for data-handling is distributed amongst a number of nodes. There is no need for a centralised facility and no practical limit on expandability. Unlike the rigid structure of local area networks, the mesh configuration of wide area networks grows in an irregular manner to meet the fluctuating needs of various sections of its environment. Nodes can be added to and taken away as demand dictates. It seems therefore that the flexible, distributed topology of wide area networks exhibits properties which satisfy objectives (f) to (h) and, in so doing, overcome the shortcomings of existing local area networks.

Two different approaches to wide area networks were described in Section 1.2. Arpanet uses a packet switching technique to break messages up into smaller blocks and transmit each one through the network as a separate entity. In contrast, the public telephone network establishes a temporary sequence of point-to-point links between end users. That sequence is dedicated to the one connection for the duration of the

interaction. This setup is very similar to that of an automated patch panel. As mentioned in the discussion on Floodnet, the dedicated connection is a major limitation in an interactive environment because a high degree of concurrency is needed to make use of scarce network resources and allow many connections at the one time. Packet switching techniques allow this concurrency by supporting several different logical connections on the one physical cable (using some form of multiplexing). Furthermore, Rosner [Rosn.1976] proves experimentally that "it is more efficient to packet switch short messages and circuit switch long messages, or groups of short messages sent frequently enough to justify holding the connection". He states that the crossover between short and long messages is heavily dependent on the network technology, but is usually in the range of 2000 to 50000 bits per message. Since interactive traffic consists of a large proportion of single character messages (Section 2.2), Rosner's conclusions also support the use of packet switching techniques in such an environment.

It seems therefore, that the wide area network configuration combined with the advantages of packet switching presents a possible solution to satisfying the previously outlined objectives. Can these wide area packet switching techniques be incorporated successfully into the local area environment? In particular, will such a network meet the networking needs of the interactive computing environment at Massey

University? This research has involved the design and implementation of such a network in the hope of answering these questions and providing a practically usable system.

CHAPTER 3

DESIGN ISSUES IN PACKET SWITCHING NETWORKS

From the analysis in Chapter 2, it seems that the centralised resources used in existing local area networks introduce problems of reliability, security, flexibility and cost. In contrast, the flexible, distributed topology of wide area networks displays several properties which provide a sound starting point for satisfying the needs of an interactive environment. There is no need for centralised resources and no practical limit on expandability. Of the two wide area networking approaches discussed in Chapter 1, MASSEYNET adopts the packet switching approach used in Arpanet because it is better suited to the short message exchange typical of interactive computing. Before looking at the implementation details of such a network, it is necessary to understand the general problems and considerations associated with the packet switching technique.

In packet switching networks, messages are broken up into short, rigidly defined blocks called packets. These packets enter the network as separate entities and are routed (switched) from one node to the next in the direction of the destination device. Each intermediate node (along the path connecting the source and destination) receives and stores the packet in its entirety before passing it on when

circumstances allow. Hence, packet switching is classified as a store-and-forward technique. On reaching the destination node, the packets may be re-sequenced (if necessary) and delivered to the destination device.

3.1 History

The packet switching technique was first proposed by Paul Baran at the RAND corporation in the early 1960's [Blea.1979]. The concept arose from the requirement of military communication systems to "retain functional capability in the event of partial destruction due to hostile action" [Blea.1979]. Baran proposed the use of a distributed mesh type network through which short bursts of information (digitised speech in Baran's case) could be transmitted. A mesh configuration (figure 3.1) presents several alternative pathways between any two users, and so partial destruction of the network is unlikely to sever communications completely. Networks with this property are described as being robust.

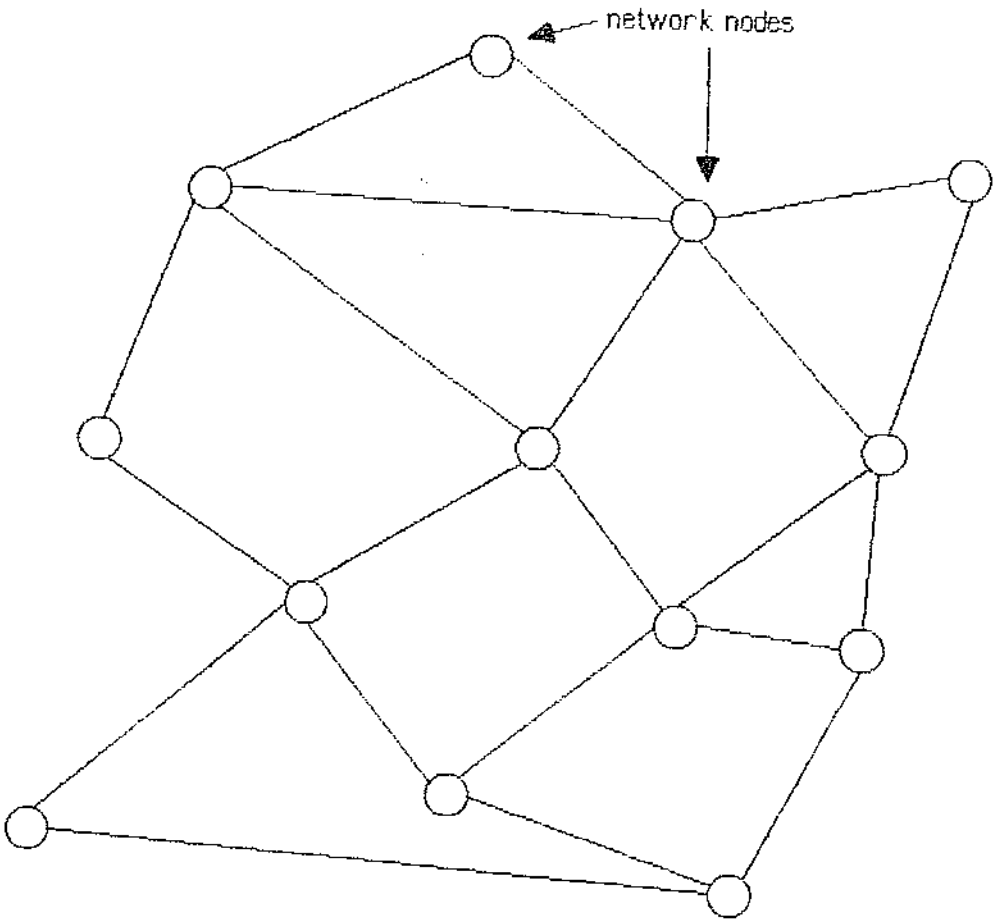


Figure 3.1 Mesh Network Configuration

3.2 Routing Algorithms

Since mesh networks provide many alternative pathways between any two network users, a routing algorithm must be used to determine which path a packet should take. These routing algorithms vary from the simple fixed schemes through to complex adaptive mechanisms. The fixed routing schemes are easier to implement but they are not sensitive to fluctuating network loads. In contrast, the adaptive algorithms attempt to optimise the flow of packets with respect to recent network traffic and changing network topologies and are thus necessarily more complex.

Fixed Routing Schemes

With fixed routing schemes, packets for a particular destination always follow the same route. Each node maintains a table with one entry for each possible destination node. That entry specifies the outgoing line through which packets for that destination should be forwarded. This routing scheme is very simple to implement but performs well only if the network topology is static and traffic loads do not change much. Fixed routing schemes do not allow easy expansion and are not robust because they fail whenever the network configuration changes (e.g. when a network node fails).

Flooding

Another simple routing scheme is the pure flooding algorithm. This algorithm requires that each node in the network transmits a copy of every incoming packet out every outgoing line except the one on which it arrived. In so doing, at least one copy of each packet must reach the required destination. The number of packets generated by this flooding process is unbounded unless some technique is employed to dampen it.

One such dampening technique is to place a "hop counter" in the header of each packet. This counter is decremented at every node and, when it reaches zero, the packet is discarded. The initial value of the hop counter must be greater than the minimum number of hops in the shortest path between the source and destination nodes.

A second technique is for each node to maintain tables of information which allow it to detect duplicate (or cycled) packets. These tables may simply consist of packet sequence numbers (discussed later) or they may even contain complete copies of each packet. Whenever a packet enters a node, these tables are searched to determine if the packet has previously passed that way. If it has, it is discarded, otherwise the flooding process continues.

Flooding is not used in most applications because of the overheads created by duplicate packets. Its tremendous robustness however, makes it very attractive for use in military installations. In such cases, the flooding process guarantees to find the required destination if at least one pathway to that node exists (regardless of how many other nodes have been destroyed).

Distributed Adaptive Algorithms

In distributed routing algorithms, the responsibility for routing packets is shared (distributed) amongst the network nodes. The fixed routing scheme described above is the simplest type of distributed algorithm. It requires each node to maintain a representation of the network configuration and the preferred direction to take to each possible destination. Other distributed routing algorithms store several possible routes for each destination and use random number generation to select which one should be followed at any particular time.

Distributed adaptive routing algorithms "adapt" to fluctuating network loads. In Arpanet, for example, each network node supports this adaptivity by maintaining a representation of the entire network together with the time delays associated with each line. Each network node collects these time delay statistics for its own lines and

periodically "floods" this information to every other node in the network, thus ensuring quick adaptation to changing traffic loads. These dynamically changing tables enable each node to to quantitatively compare alternative paths and thus select the best path for routing each packet towards its destination. In Arpanet, the "quickest" path metric is the estimated time delay, but other possible metrics include the number of hops, the estimated number of queued packets on each path, the number of established virtual circuits on each path, or any combination of these. Distributed adaptive algorithms perform well under fluctuating traffic conditions. However, since each node must know the exact network configuration, they do not allow easy expansion and are not robust because they fail whenever the network configuration changes.

3.3 Transport Mechanisms

Closely related to the routing algorithm is the transport mechanism used. The transport mechanism determines the nature of the service provided by the network as viewed by the user. For example, if one user sends a message to another user, can it be assumed that the message will arrive in order? There are two dominant mechanisms used in packet switching networks - the virtual circuit model and the datagram model.

Virtual Circuits

The virtual circuit model allocates a logical channel to each call so that packets arrive at their destination in order. The virtual circuit is established by the routing algorithm at call initiation time and is used throughout that call for packet exchange. The virtual (logical) circuit does not correspond to a physical circuit although it appears to the users as if they are directly connected via a dedicated cable. A good analogy is the public telephone system. To initiate a call, the telephone subscriber must set up a connection with the required destination. Having done this, the information (conversation) is exchanged (in order) between the two users. At the end of the conversation, the circuit is cleared.

Datagrams

In contrast, the datagram model treats each packet as a separate entity. To facilitate this independence, each packet must contain the full address of its destination. The routing algorithm is executed independently for every single packet with the result that packets may follow different paths and arrive at their destination out of order. A good analogy for the datagram model is the postal system. Each letter is transported as an isolated unit although two letters destined for the same place usually follow the route. There is no guarantee that letters will arrive in the order they were posted.

3.4 Error Control

Error control pertains to the detection of and recovery from errors during message/packet transmission. These errors take many different forms ranging from simple bit transmission errors through to missing or duplicate messages or parts of messages.

The factors affecting circuit quality are numerous and varied. Transmission errors on telephone lines, for example, may be caused by old switching devices, lightning, clumsy telephone repairpersons, surges on power lines, crosstalk between physically adjacent wires etc [Tane.1981].

The most commonly used error correction system is Automatic Retransmission on Request (ARQ). In the ARQ system, the receiver will request that packets in which errors were detected are retransmitted. Each packet contains redundant information to enable the receiver to detect that an error has occurred. After transmitting a packet, each transit node in the source-to-destination pathway maintains a copy of that packet until it receives an acknowledgement from the receiving node. On arrival at the receiving node, the packet is checked against its error detecting code and an acknowledgement is transmitted to the sending node. A positive acknowledgement indicates successful transmission and means that the sending node can discard its copy of the

packet. A negative acknowledgement signals that an error has occured and so the sender must retransmit a copy of the previous packet.

Error detecting codes can be used for both node-to-node and end-to-end (source-to-destination) error control. The node-to-node software uses these codes to eliminate transmission errors occuring in the lines between adjacent nodes. The end-to-end error detecting codes are used as a safeguard against errors introduced by the transit nodes in the source-to-destination pathway (as discussed below).

Inherent in the acknowledgement system of store-and-forward packet switching networks is the possibility of lost or duplicate packets. Packets may be lost if a node goes down after receiving a packet and acknowledging it before transmitting it to the next node in the pathway. Duplicate packets occur whenever a node goes down after receiving a packet and forwarding it on before sending an acknowledgement. The previous node, on not receiving an acknowledgement, retransmits a copy of the packet thus creating a duplicate. Usually, each node follows the same packet transmission/acknowledgement order and so nodes going down will result in either lost packets or duplicate packets but not both.

The problems of lost or duplicate packets can be overcome by including sequence numbers in each packet originating from a particular source node. These numbers specify the relationship between successive packets and allow the destination node to detect which packets, if any, have been lost or duplicated. The sequence number cycle must be sufficiently large so that a packet with a particular sequence number is always acknowledged before that sequence number is used again in a later packet.

3.5 Flow Control

Flow control procedures are used as a "throttle" to prevent a sender from transmitting data at a faster rate than the receiver can handle. These procedures either require the sender to gain explicit permission from the receiver before transmitting or they allow the receiver to discard packets at will and rely on packet retransmission. Two well-known flow control protocols are the stop-and-wait and the sliding-window protocols [Tane.1981].

The stop-and-wait flow control mechanism allows the sender to transmit just one packet and then wait for an acknowledgement before transmitting another. This protocol relies on feedback from the receiving node. When that node receives a packet, it must send a control packet (acknowledgement) to the sender giving it permission to transmit another packet. This means that packet transmission time is effectively doubled because a second packet can only be transmitted after the sender receives an acknowledgement for the first one (as shown in figure 3.2).

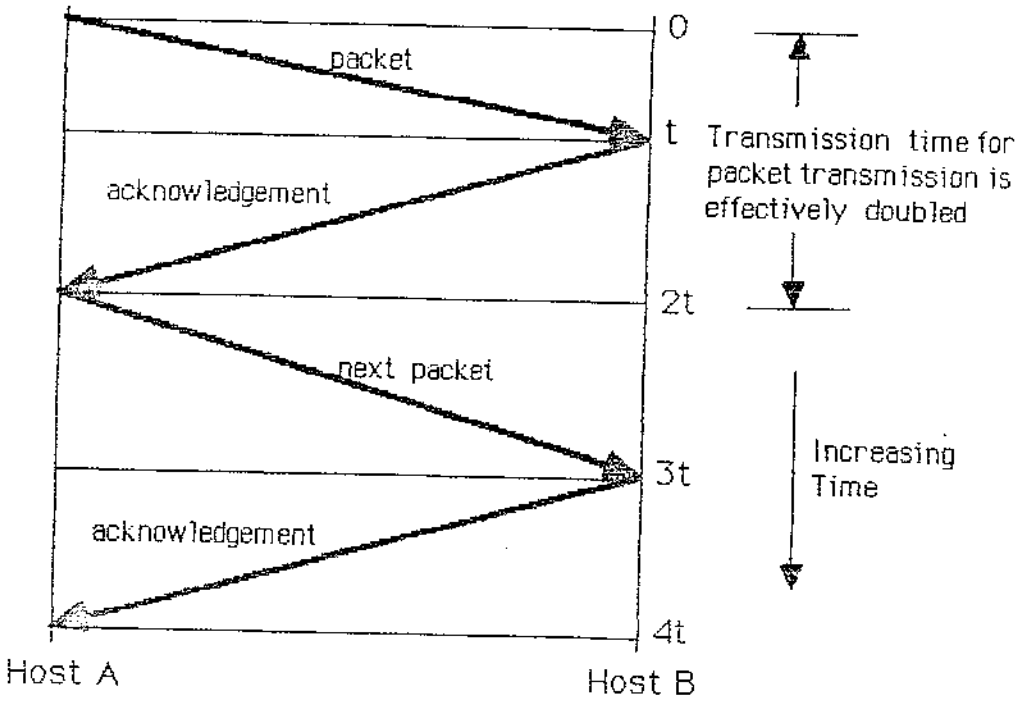


Figure 3.2 Stop-and-wait Flow Control Mechanism

The sliding window protocol extends this stop-and-wait technique to make more efficient use of channel bandwidth. Instead of allowing just one unacknowledged packet per sender, the sender is allowed to transmit several packets before waiting for a reply from the receiver (see figure 3.3). The concurrent passage of packets from one message through the network is referred to as pipelining and it results in far greater use of available channel capacity. The term "sliding window" refers to the set of allowable sequence numbers which can be used for transmitting packets from the one sender (figure 3.4). The sequence cycles as new

packets are sent out and previous packets are acknowledged.

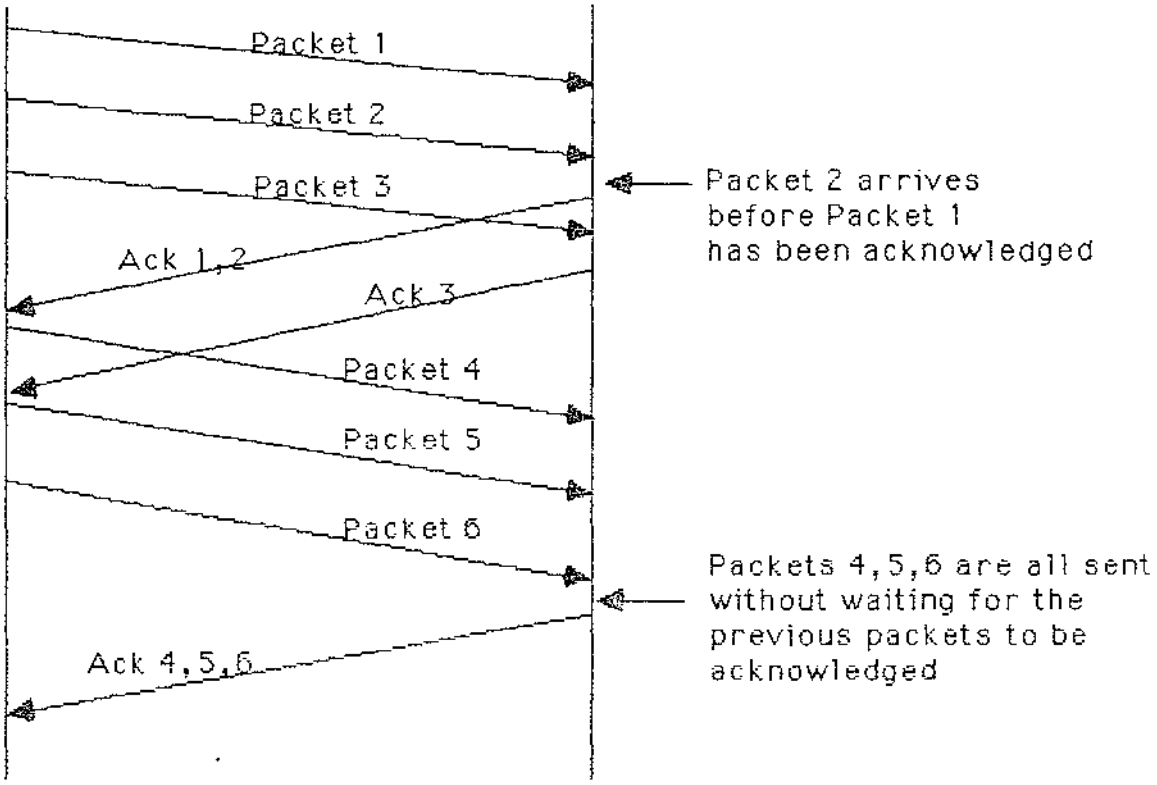


Figure 3.3 Sliding Window Flow Control Mechanism

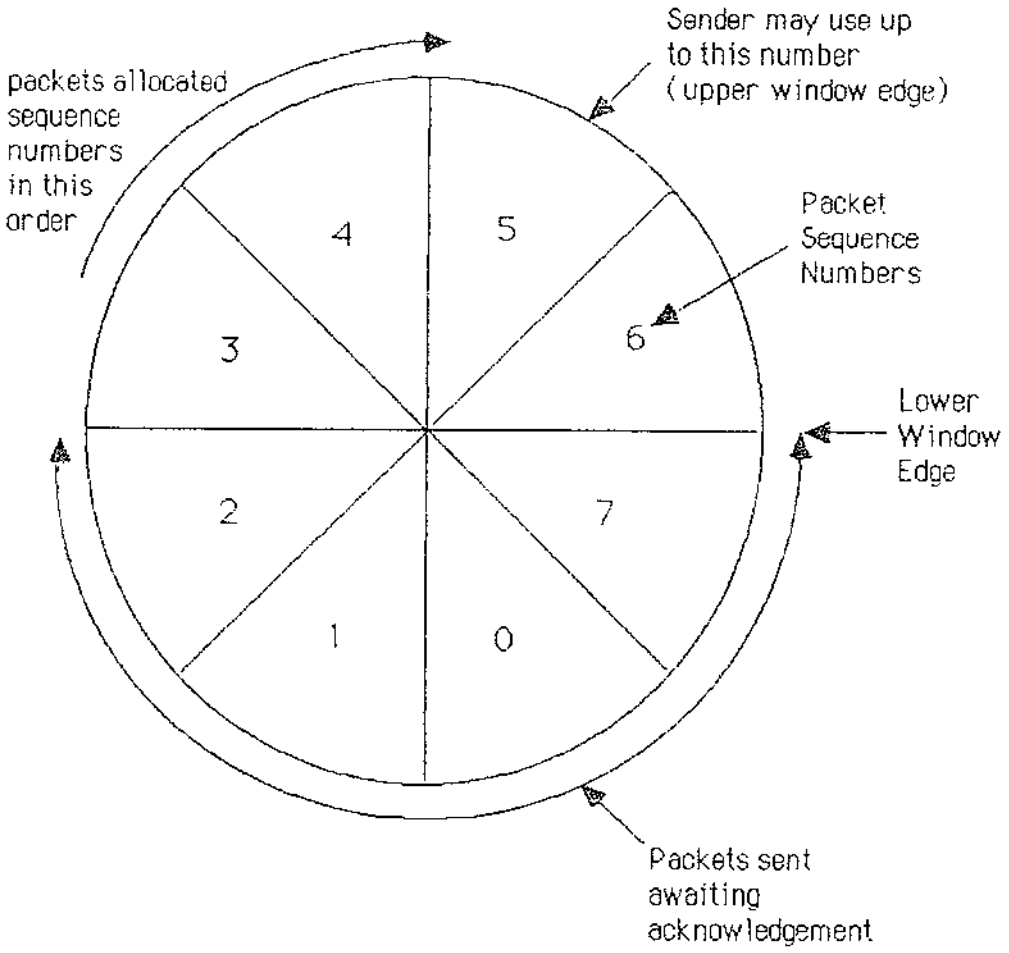


Figure 3.4 Sliding Window Of Outstanding Packets

Another feature of the sliding window protocol is the piggybacking of acknowledgements onto packets. In the stop-and-wait system, a packet receipt is acknowledged by the immediate transmission of a special control packet. Using piggybacking, the receiving node first of all determines if a

packet is currently being transmitted. If so, then the acknowledgement is given a free ride (i.e. piggybacked) in the header of the packet. Otherwise, if the line is idle, a separate control packet is sent as in the stop-and-wait technique. Piggybacking of acknowledgements makes better use of channel bandwidth because it only requires a few bits in the packet header (instead of a separate control packet).

3.6 Congestion Control

Congestion control deals with the problem of more packets arriving at a node than there are buffers to store them. This situation can happen if, for example, three input lines are delivering packets at full speed, all of which are to be forwarded along the same output line (figure 3.5). Since the input traffic rate exceeds the capacity of the output line, this extra data must be queued (buffered) for output. If traffic increases too far, the finite buffer storage will eventually run out and packets will be lost. The end-to-end (source-to-destination) flow control procedures (described in the previous section) prevent one host (network device) from saturating the other but do not control the total amount of traffic in the network. What is needed is some mechanism for limiting the volume of traffic between adjacent nodes. Ideally, the mechanism should not restrict network throughput when there is no threat of congestion.

Congestion can be controlled by permanently pre-allocating buffers to each virtual circuit in each node. If pipelining is used, each node must allocate a full window's worth of buffers to each virtual circuit. Doing this guarantees that there will always be enough space to store incoming packets until they can be forwarded. The buffer space could be reserved at call setup time. This congestion control

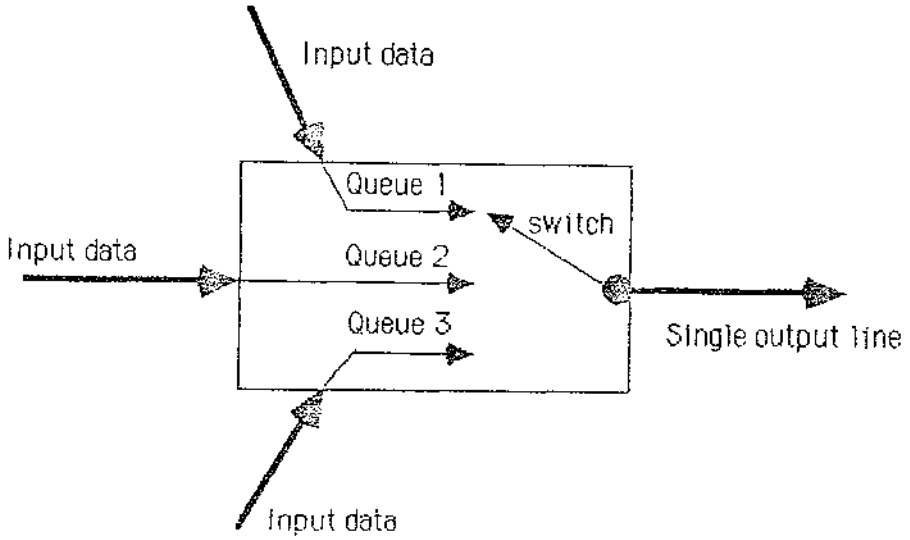


Figure 3.5 Example Cause Of Congestion

mechanism brings packet switching closer to circuit switching because network resources are permanently allocated to each connection. It is potentially inefficient because it restricts network throughput even when there is no threat of congestion (because resources allocated to one virtual circuit are unavailable to anyone else).

Perhaps the simplest congestion control scheme is packet discarding. In contrast to buffer pre-allocation, no resources are reserved in advance. If a packet arrives at a node and there is no room to store it, it is simply discarded. The sending nodes, after waiting for a set time for an acknowledgement, will time out and retransmit a copy

of the discarded packet. This control scheme is very simple but the retransmission of packets requires large amounts of channel bandwidth. Another problem is deciding how long the time out period should be. A compromise must be made between increasing network delay with long timeouts and generating unnecessary duplicate packets when timeouts are very short.

A third congestion control mechanism uses choke packets to regulate the packet flow through the network [Tane.1981]. Each network node maintains some measure of recent utilisation statistics for each of its output lines. If the utilisation measure rises above a certain threshold, a choke packet is sent to the source of each subsequent incoming packet on that line. On receiving a choke packet, the source host reduces its packet flow for a fixed period of time. If more choke packets are received in that time, the packet flow is further reduced; otherwise the packet flow restriction is lifted. This feedback can be made to adapt quickly to fluctuating line utilisation and hence prevent congestion. Furthermore, it does not restrict transmission unless congestion is imminent.

3.7 Deadlock Prevention

The ultimate form of congestion is deadlock. As described in Section 1.2, a deadlock is a situation in which messages can make no further progress because the resources of two or more interdependent nodes have been exhausted (usually no more buffer storage). Each node is waiting on the other for that particular resource and so the network is locked in that situation forever.

To prevent deadlocks from occurring, the communication protocol must incorporate algorithms for controlling resource allocation. During times of heavy network usage, some of the available resources should be reserved for critical control packets which may help undo the deadlock situation. For example, the source node may timeout and abort its previous transmission (by sending a message abort packet), thus avoiding deadlock.

3.8 X25 Recommendations

With the development of several experimental packet switched networks in the early 1970's, the CCITT (The International Telegraph and Telephone Consultative Committee) saw the need for international agreement on a packet switched network access standard [Blea.1982]. Since then, the term "X25" has been closely associated with packet switching because it refers to the set of recommendations which define the interface between the user's equipment and the network node to which it is attached. It makes no assumptions about the network except that packets are delivered to the destination device in order. X25 applies only to user equipment of sufficient intelligence to implement the X25 protocol and connected by a synchronous communication circuit. To enable simple devices to attach to the network, a set of associated standards (namely X3, X28, X29) were developed. X25 proposes standards for the bottom three layers of the ISO OSI reference model.

At layer 1 (the physical layer), X25 uses a standard called X21 which specifies that this level should provide a bit-serial, synchronous, full-duplex, point-to-point circuit for digital transmission [Blea.1982].

The data link layer (level 2) must take the physical communication and convert it into a circuit free of transmission errors. X25 enforces error control and flow control using the frame structure of the High-Level Data Link Control (HDLC). The HDLC frame structure defines the meaning of each byte in a block of data. Each block is preceded and succeeded by a special flag character and contains addressing information, control information, and an error checking code (as shown in figure 3.6) [CCIT.1981]. X25 utilises a Link Access Procedure (LAPB) which specifies that either station may send commands at any time and initiate responses without receiving permission from the connected device.

FLAG	ADDRESS	CONTROL	INFORMATION	ERROR CODES	FLAG
01111110	8-bits	8-bits	N-bits	16-bits	01111110

Figure 3.6 HDLC Frame Structure

The X25 recommendations for the network layer (layer 3) define the packet formats and the actions taken when various types of packet are transmitted or received. It must provide the multiplexing of several logical channels onto a single physical cable, error and flow control at the user/network interface, packet sequencing, virtual circuit control, and packet size conversion. The X25 data packet format, illustrated in figure 3.7, consists of several bytes of control information followed by user data. These packet formats were designed for use in long-haul (wide area) networks where large packet sizes are common. A default of 128 bytes is used but this can be changed to certain other values by the user [Tane.1981].

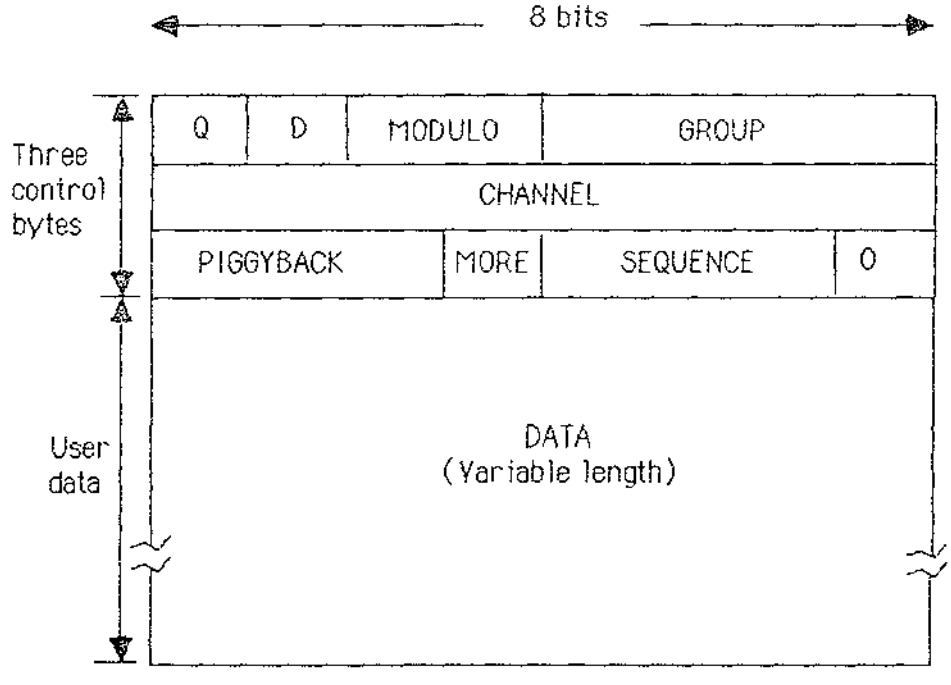


Figure 3.7 X25 Data Packet Format

3.9 Conclusion

This chapter has identified several key issues and possible solutions associated with the packet switching technique. The importance of each of these issues will change as we move from the wide area networking environment into a local area situation. For example, the routing algorithm needs to be more flexible to handle the fluctuating needs of the local area users. As well, the emphasis on error control will diminish with the use of dedicated cables instead of public telecommunications lines.

The traffic characteristics will differ greatly between the local and wide area environments. The X25 recommendations for wide area networks were designed for the transfer of large packets of data (usually 128 characters) over long, error-prone circuits. The X25 packets include complex error control, addressing and sequencing information which ensures that messages entering the network arrive at their destination error-free and in order.

In a local area interactive environment however, response times are of critical importance. The emphasis shifts from providing complex communication control mechanisms to ensuring low delay through the network. However, the issues of error, flow and congestion control must still be dealt with in such and environment. The solutions to these issues

and their subsequent implementation are discussed in the next Chapter.

CHAPTER 4

IMPLEMENTATION OF MASSEYNET

In the development of Masseynet, special attention was paid to the networking requirements of Massey's interactive environment, while at the same time maintaining sufficient generality to be adaptable to other environments. To meet the objectives outlined in Chapter 2, it was decided to use a point-to-point, mesh configuration common to wide area networks. This approach involved the implementation of a local area network using Z80-based communications nodes and a protocol based on the wide area packet switching techniques.

4.1 Hardware Used

The mesh topology involves an irregular arrangement of nodes connected in a point-to-point fashion by some form of cabling.

The network node used in MASSEYNET (figure 4.1) is a single-board computer with a Z80 microprocessor, two Z80-SIOs (serial input/output), a 4Mhz Z80-CTC (counter timer circuit), 60K of RAM and 4K of ROM.

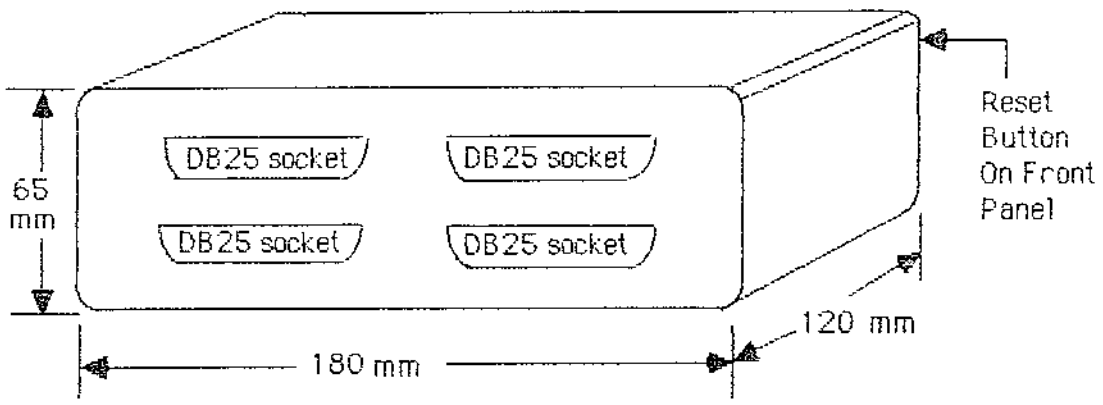


Figure 4.1 Z80-Based Data Communications Nodes

The Z80-SIO is a dual channel component which is capable of a wide variety of serial data communication functions. Each network node uses them to drive a total of four RS232 (socket) ports.

The Z80-CTC is a programmable, four channel device that provides counting and timing functions. The four independent channels can be configured by the CPU to operate under various modes and conditions.

The 60K of RAM is available for storing the network-controlling program and for buffering data as it passes through the network. In addition, three characters may be buffered in each SIO chip. These Z80-based nodes are available from within Massey University at a cost of

approximately $240.

The network nodes are connected by four-wire (individually screened) cabling with a DB25 plug at either end. University standards [Lyon.1985] require the transmit and receive wires to be swapped within the cable. Devices attach to the network via the Department's three patch panels using the same four-wire cable but with a DB25 plug at one end and a DB9 plug at the other. Once again, the transmit and receive wires are swapped. Both types of cables can be made within the University for approximately $12.

Although the University standard is designed around a 4-wire protocol incorporating the RS232 signals Txdata, Rxdata, DTR and CTS, which enables hardware flow control to operate, the present node hardware is only capable of using Txdata and Rxdata. Work on a modified device incorporating hardware flow control is currently under way, but this device was not available for testing in the network as described here.

4.2 Network and Program Design

The design and subsequent implementation of MASSEYNET was an iterative process. The aim was to develop an initial skeleton network program onto which could be added step-wise enhancements. The initial network needed to be usable but did not have to satisfy all of the objectives outlined in Chapter 2. Instead, each new version improved upon the weaknesses of the previous one, thus bringing it closer to the list of ideal objectives.

As mentioned in Chapter 1, the ISO OSI reference model was used as a design guide in dealing with the issues related to packet switching techniques. The design of MASSEYNET has corresponded roughly to the bottom four layers of this model. The use of Z80-based computers for network nodes however, has meant that processing power was at a premium. Consequently, the implementaion of MASSEYNET has been directed by efficiency, rather than the layered structure proposed by the ISO reference model.

4.2.1 Program Structure

The program itself, written in Z80 assembly language, consists of two distinct parts. The low level procedures control the serial input and output through each of the four Z80-SIO ports, while the upper level implements the network's communication protocol. An important consideration was the design of a suitable interface between the two parts. Two approaches were considered - the use a multi-tasking kernel, and the use of common data areas coupled with an interrupt mechanism.

In the multi-tasking kernel approach, the network program consists of a number of well-defined processes which interact with each other via semaphores. The low level input processes receive and group the incoming characters into packets. When the packets are complete, a semaphore is signalled to initiate a high level process. For example, the arrival of a data packet triggers off the code for processing data packets (as shown in figure 4.2). The processing of that packet usually requires some form of output. To achieve this, the high level process must signal another semaphore which initiates the low level output processes (see figure 4.2). These output processes are responsible for transmitting the data through the appropriate SIO port.

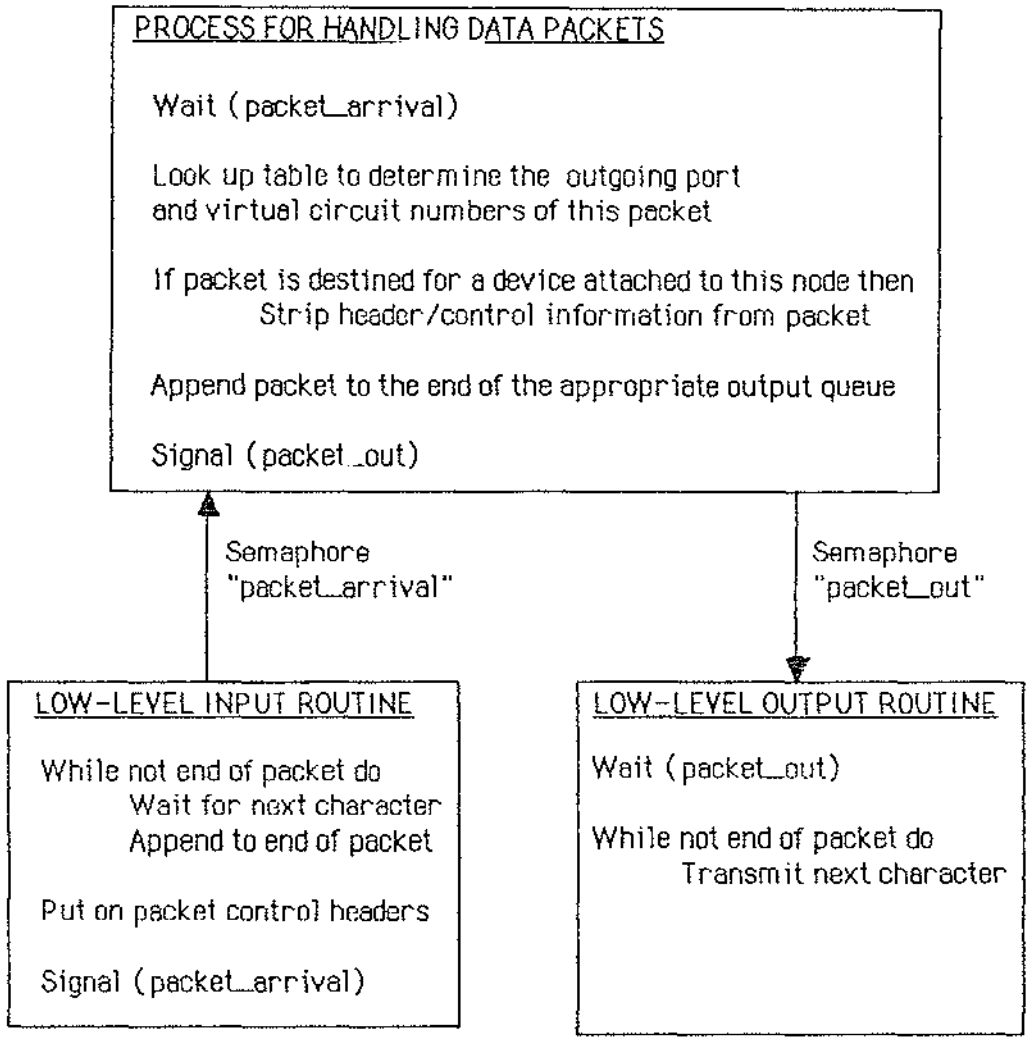


Figure 4.2 Multi-tasking Kernel Interface Between The Low-level Input/Output Routines And The High-level Network Protocol Routines.

This multi-tasking approach is well suited to the ISO OSI seven-layer model because each layer can be thought of as a separate process performing a well-defined task. The interface between different layers can be compared to the inter-process communication provided by semaphores. Ordering of these tasks (layers) is achieved by signalling the corresponding semaphores on which the various processes are suspended (waiting).

The second approach uses common data areas and interrupts to provide a suitable interface between the low level I/O routines and higher level network protocols. The Z80 provides fast interrupt response to character receipt and transmitter free states. E.g. the "character received" and "empty transmit buffer" interrupts from each channel result in calls to different input and output routines. The input code transfers incoming characters into a common input area where they are processed by the high level routines. Any resulting output is transferred to a common output storage area where it awaits transmission by the low level I/O routines (see figure 4.3). Since different routines require access to the same data, some form of mutual exclusion must be enforced.

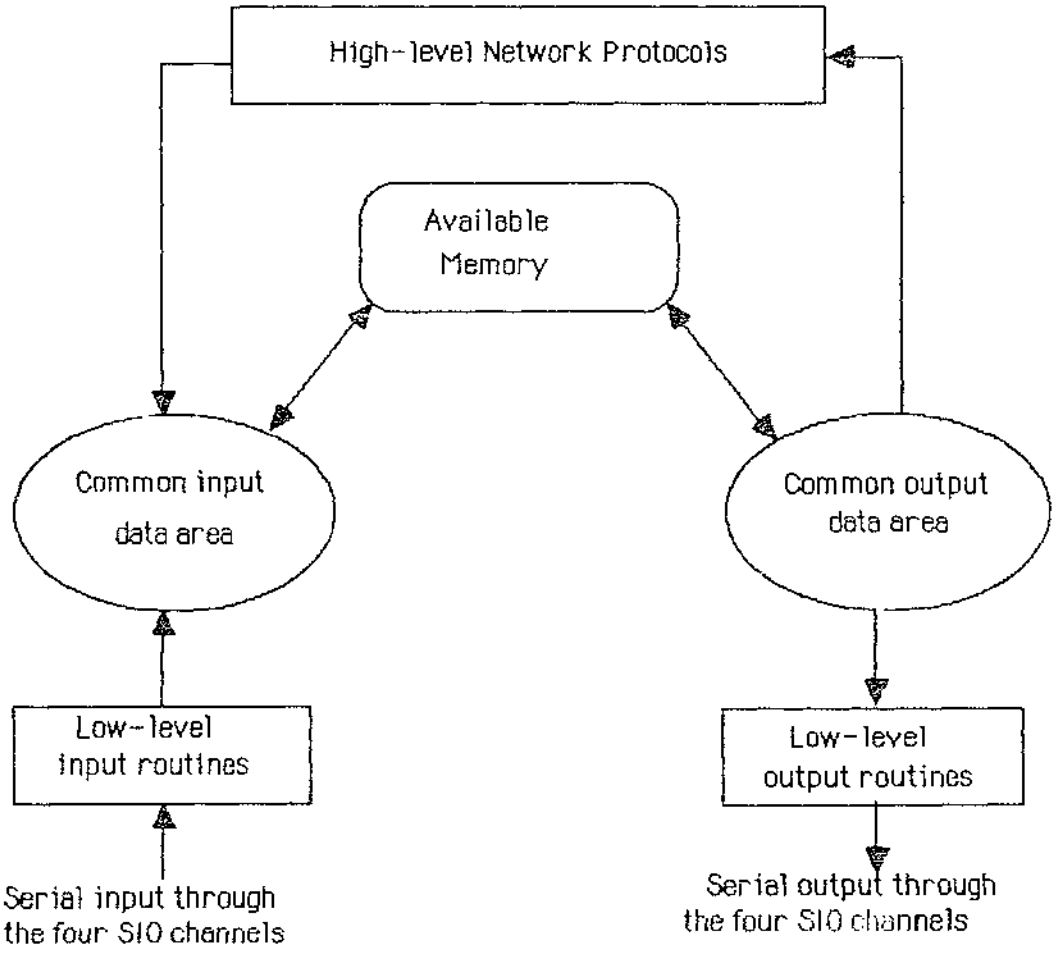


Figure 4.3 The Use Of Common Data Areas To Provide An Interface Between The Low-level (Interrupt-Driven) Input/Output Routines And The High-level Network Protocol Routines.

In implementing MASSEYNET, it was decided to use the second approach to providing inter-level communications for two main reasons. Firstly, the multi-tasking kernel under development elsewhere in the Department was not complete at the time and secondly, experience with the preliminary version of this kernel indicated that a Z80 processor using it would be unable to cope with processing overhead of a generalised multi-tasking kernel if heavy traffic was experienced. In using the second interrupt-driven approach, mutual exclusion is enforced by disabling interrupt processing around any routines which access either of the common data areas.

4.2.2 Programming Strategies

Since the Z80-based network nodes do not support hardware flow control, it was essential that the network-controlling program(s) should be I/O bound. If this is not the case and processing power is the limiting factor, then characters will be lost during periods of heavy traffic. Since transmission and receipt of each character is serviced by an interrupt routine, these routines must be fast enough to handle the worst case situation of data input and output.

In the Computer Science Department at Massey, all lines operate at 9600 baud. This means that 960 asynchronous characters can be received and transmitted per second through each of the four SIO ports on the Z80 communications nodes. This gives a total of 7680 calls to character-handling interrupt routines per second. Clock interrupts (which will be discussed later) increase this figure to 8130 interrupts per second. Since the Z80 nodes use a 4Mhz clock, this means that each interrupt routine can use at most 492 clock cycles if the program is to remain I/O bound. This is equivalent to approximately 140 Z80 instructions (assuming an average of 3.5 clock cycles per instruction).

To optimise the throughput of the program and ensure that it remained I/O bound, two programming strategies were followed.

Firstly, it was decided to develop four copies of the character-received and transmitter-free interrupt routines, each one specific to a particular port, rather than a single longer routine which was general to all ports. The general routine requires the use of offsets to access the variables associated with the interrupting SIO. These offsets increase the number of required instructions (by approximately 30%) and hence the processing time for that routine. While the four specific routines together take up more space than the single larger routine, this is justified by the significant reduction in processing time for each interrupt.

Secondly, the code for these interrupt routines avoids the use of the time-consuming IX and IY registers. Instructions using these registers can take up to 5.25 clock cycles. In addition, inline code is used to avoid the overhead of the CALL and RET sequences.

4.3 Communication Protocols

Packet switching networks require the use of different protocols for inter-node communication and for communication between external devices and network nodes. The device-to-node interaction involves the exchange of arbitrary-sized groups of characters while the node-to-node protocol controls the exchange of packets of data which may be variable in size but which have a rigidly defined structure.

MASSEYNET supports asynchronous communication between each device and the network node to which it is connected. The device-to-node interface is responsible for grouping the incoming characters into the required packet formats and extracting data from the packets destined for that device (Packet assembly and disassembly (PAD) functions).

Between adjacent network nodes, a protocol corresponding to the data link layer (layer 2) of the ISO model is needed to transfer packets as well-defined, separate entities. As described in Chapter 3, the protocol defined by the X25 recommendations is called the High Level Data Link Control (HDLC) [Blea.1982]. HDLC transmissions use frames (similar to packets) which are delimited by the special flag character 01111110. Transmission is bit oriented, which means that a frame can consist of any sequence of any number of bits. The

last 16 bits preceding the end of frame flag character are used for error detection. The Z80-SIO supports a variant of HDLC called Synchronous Data Link Control (SDLC). Unfortunately, the available Z80 communication nodes could not synchronise the receive logic with the transmitted data. Hence, some other protocol for packet exchange had to be developed.

Communication between adjacent network nodes involves the exchange of packets of data which may vary in length up to a fixed maximum size. To do this, MASSEYNET uses an asynchronous character-oriented protocol which marks the end of each packet with an ETX character (ASCII 3). Because ETX characters may appear within the data itself, some form of code transparency is also required. The MASSEYNET protocol provides this by transmitting a DLE character (ASCII 16) in front of every ETX or DLE character which appears as part of the data. On receiving a DLE character, the receiver knows that the next character should be considered as a part of the data regardless of its value (see figure 4.4). ETX and DLE characters are not often used in most environments and so the code transparency mechanism is rarely needed. The use of asynchronous communication introduces an overhead of 20% because each 8-bit character is preceded by a start bit and terminated with a stop bit. The end-of-packet (ETX) character introduces a further one character overhead per packet. MASSEYNET currently allows a maximum packet size of

12 characters which means that the node-to-node communication protocol introduces a total overhead of approximately 30% for full packets.

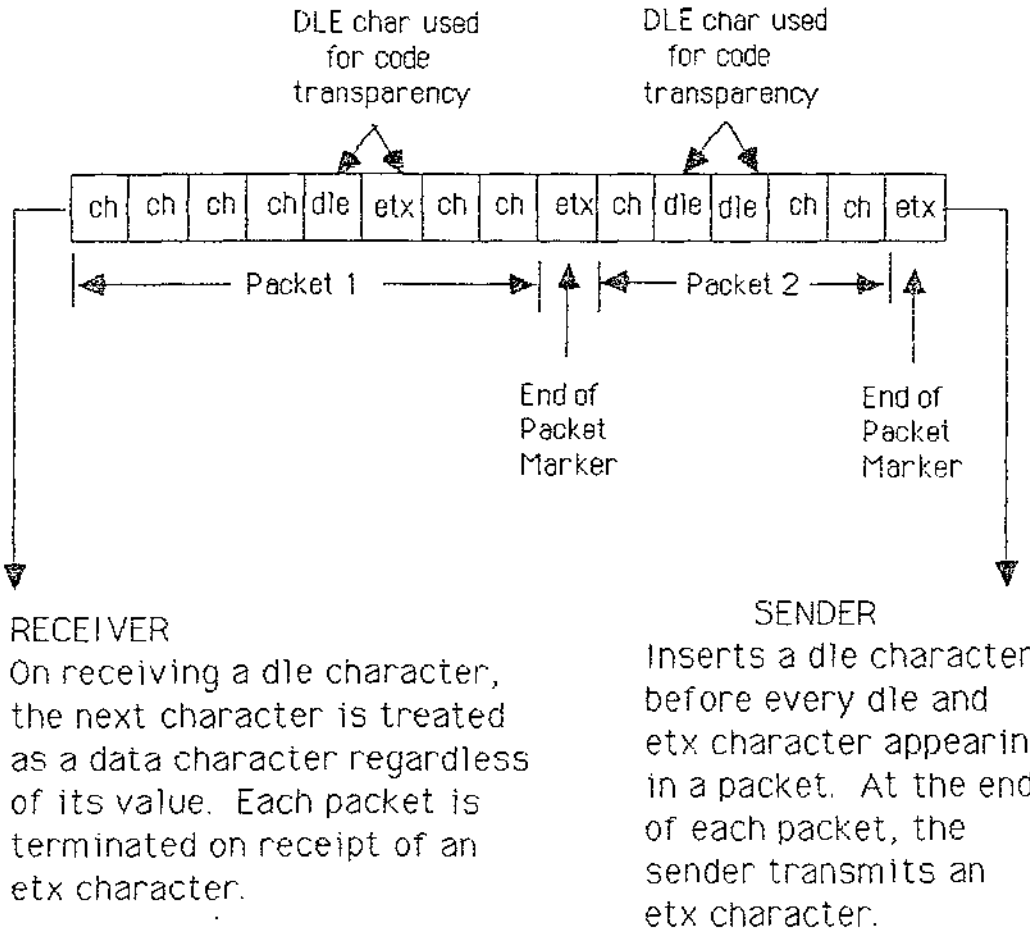


Figure 4.4 Inter-node (Data Link Layer) Communication Protocol

4.4 Packet Formation Criteria

As mentioned in the previous section, the device-to-node interface in MASSEYNET is responsible for grouping the asynchronous characters into well-defined packets and extracting the data from packets destined for the attached device. The packet disassembly is straight-forward because the packet's data is simply queued for output at the corresponding SIO port.

Packet assembly is more complex because of the variable sized packets which the network is capable of handling and the consequent need to optimise their size. If a packet is formed every time a character is received, then the control byte in each packet (Section 4.9) represents a 100% overhead. This means that the net data rate obtained from the network would be much less than the overall rate. Ideally, the node should wait a short period of time to determine if subsequent characters are following immediately behind the first. If they are, then packet formation should be delayed so as to fill the packet with its maximum quota of characters. It is thus necessary to determine the crossover point between increasing network delays for packet formation and decreasing network processing overheads with maximum sized packets.

An experiment was conducted to analyse the character exchange between a terminal operator and a computer over a 9600 baud line. The experiment analysed the traffic in both directions by keeping a frequency distribution of the elapsed time between successive characters. The terminal operator was a secretary with proficient typing skills. The results for traffic in both directions are plotted in figures 4.5(a) and 4.5(b).

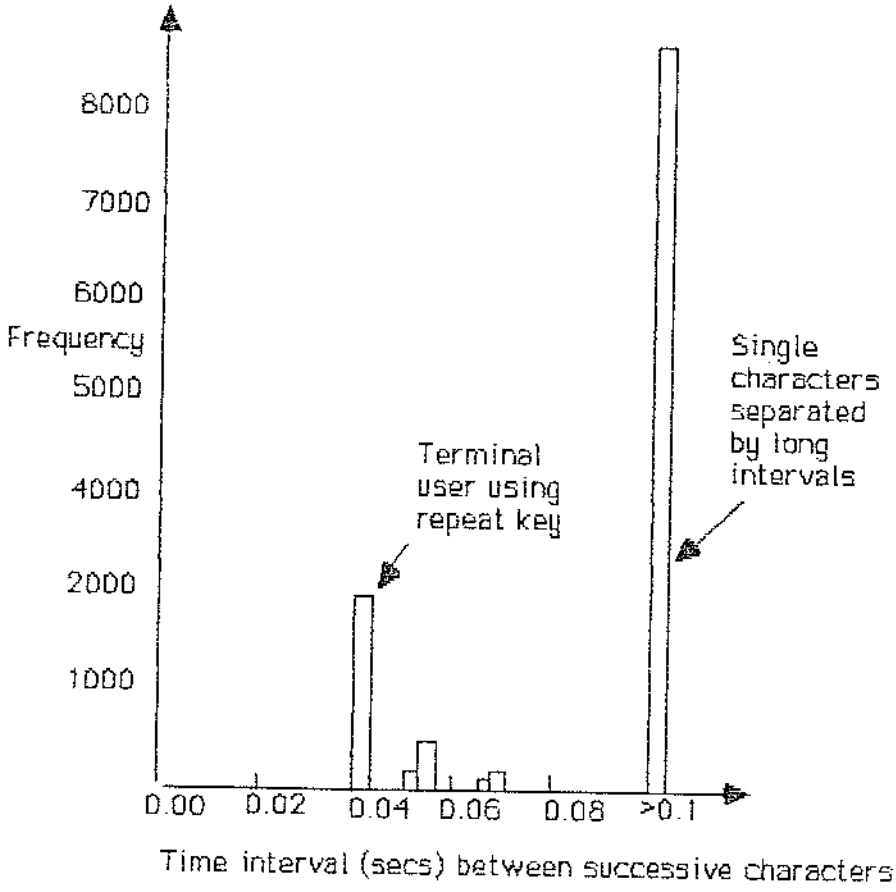


Figure 4.5a) Graph Of Inter-character Time Differences Versus Frequency For Terminal-to-Computer Traffic.

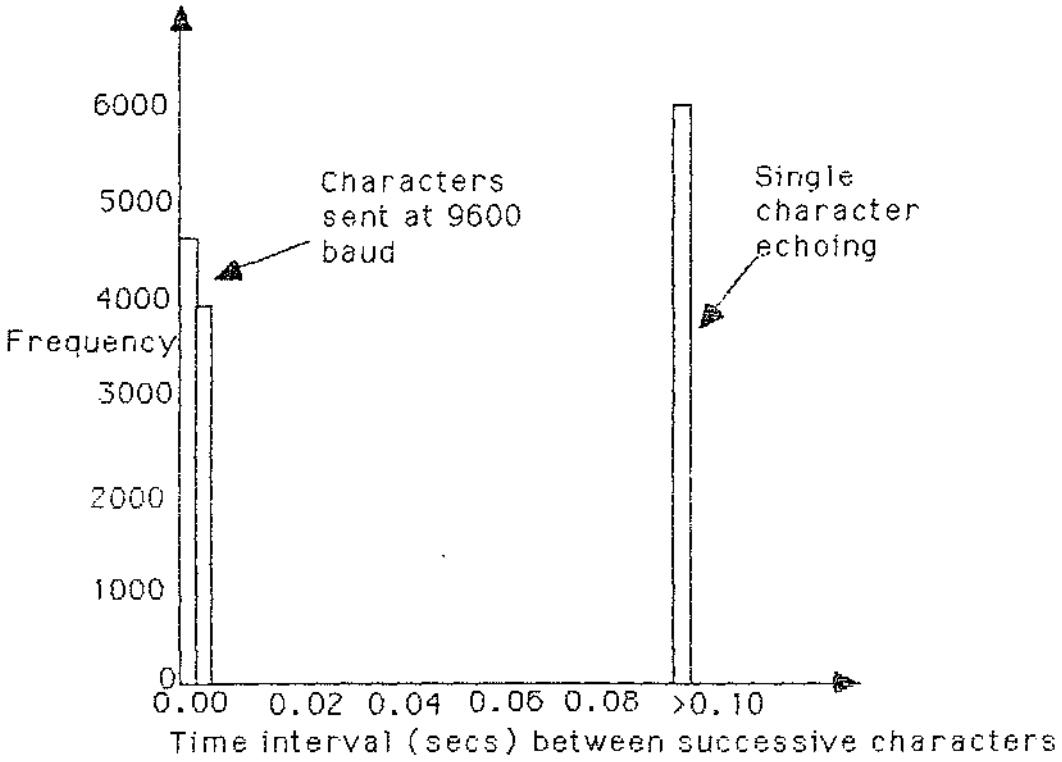


Figure 4.5b) Graph Of Inter-character Time Differences Versus Frequency For Computer-to Terminal Traffic.

The graph of terminal-to-computer traffic (figure 4.5(a)) shows that there were large gaps between successive characters entered by the human operator. In contrast, the computer-to-terminal graph (figure 4.5(b)) shows large peaks at elapsed times of 0.001 and 0.002 seconds as well as a smaller peak at high elapsed times (associated with the echoing of typed characters). The peaks at 0.001 and 0.002 seconds both correspond to multi-character messages transmitted at 9600 baud by the computer. The two peaks result from round-off error associated with the time counter used in the experimental program.

Woodson [Wood.1982] states that the response to a signal (e.g. feedback to a pressing key) should be set to less than 0.1 seconds to be acceptable to an interactive user. Looking at the graphs, it seems that successive characters from a terminal operator should never be grouped into the one packet, as this would require an excessively large inter-character timeout (e.g. 0.05 seconds) and would result in significant network delays. Instead, the graph of computer-to-terminal traffic suggests that the inter-character time delay should be slightly greater than 0.002 seconds. This timeout value ensures that multi-character messages from the computer (e.g. full screenfuls of data) are grouped and sent as maximum sized packets. This decreases the control overheads in each packet and results in a much higher net throughput of data.

4.5 Transport Mechanism

As discussed in Section 3.3, virtual circuits set up a logical channel for each call so that packets arrive at their destination in order. In contrast, datagrams treat each packet as a separate entity with the result that packets may follow different paths and arrive at their destination out of order. Datagrams require each packet to carry the full name of its destination address.

MASSEYNET uses the virtual circuit approach because it is better suited to the short message, session-oriented communication typical of interactive environments. Since the packets tend to be fairly short, a full destination address (say, up to five characters) in every packet represents a large overhead, and hence wasted bandwidth. Furthermore, interactive work requires packets to arrive in order and so datagrams would require extra information in each packet so that resequencing could be done at the destination node. With virtual circuits however, all packets follow the same route and so the original ordering of packets is maintained.

The price of using virtual circuits is the table space required and the time taken to set up the connection. In MASSEYNET, buffer space is not a critical resource and so virtual circuit tables can be easily accomodated. The cost of setting up the initial connection can also be justified by

the session-oriented nature of interactive computing. I.e. the length of the interaction is large compared with the time taken to establish the initial connection. For these reasons, MASSEYNET uses a virtual circuit mechanism for transporting packets between network users.

4.6 Buffer Allocation

The Z80 communication nodes have 60K of RAM which is used to hold the network program and to store packets as they traverse the network. The common input and output storage areas discussed in Section 4.2 use this memory to provide a suitable interface between the low level I/O routines and the higher level routines which implement the network protocol. The size of these two storage areas must expand and contract dynamically to meet the demands of fluctuating traffic loads through the network. One of the early implementation considerations was the memory management and allocation problem. I.e. what data structure should be used to flag unused memory and how should the data elements be allocated to the two storage areas? Two methods were considered.

The first method required memory to be segmented into fixed length blocks each of which could hold one maximum length packet. The unused blocks are linked together in a pointer queue to facilitate easy expansion and contraction. One of these blocks is allocated to each incoming and outgoing packet regardless of that packet's length. This method is simple but because most terminal-to-computer traffic consists of single-data-character packets, it can result in the wastage of quite large amounts of buffer space.

The second method is to segment memory dynamically according to the size of each particular packet (using a best-fit algorithm). This method can make more efficient use of buffer space but introduces the complication of garbage collection. In addition, the best fit algorithm requires more complex processing routines.

MASSEYNET uses the first approach because of its ease of implementation and because buffer space was not considered a critical resource while the processing power was. Hence, memory is allocated in fixed length blocks, each of which can hold a maximum size packet. Once packet processing and transmission is complete, the corresponding block of memory is returned to the list of unused memory.

4.7 Input/Output Queues

As mentioned in Section 4.2, MASSEYNET uses a common input data area to provide a suitable interface between the "character received" interrupt routines and the high level network protocol. This input data area consists of four linked lists of packets, one for each SIO port as shown in figure 4.6. After packet formation (Section 4.4), the resulting packet is appended to the end of the corresponding input queue where it waits to be processed by the main program. The main program polls each of the four queues in round-robin fashion checking for packets. If a queue is non-empty, the first packet is taken off it and processed by the main program. This processing involves establishing and terminating requested connections, and mapping various packets into the common output area to await transmission.

The common output data area in the initial version of MASSEYNET contained four output queues, one for each SIO port. This queueing mechanism worked satisfactorily but occasionally resulted in very high network delays. These delays were caused by the first-in-first-out queueing discipline used to share the single physical cable between several different virtual circuits. This discipline allows a single virtual circuit to monopolise the channel at the expense of the other circuits it is sharing with. For

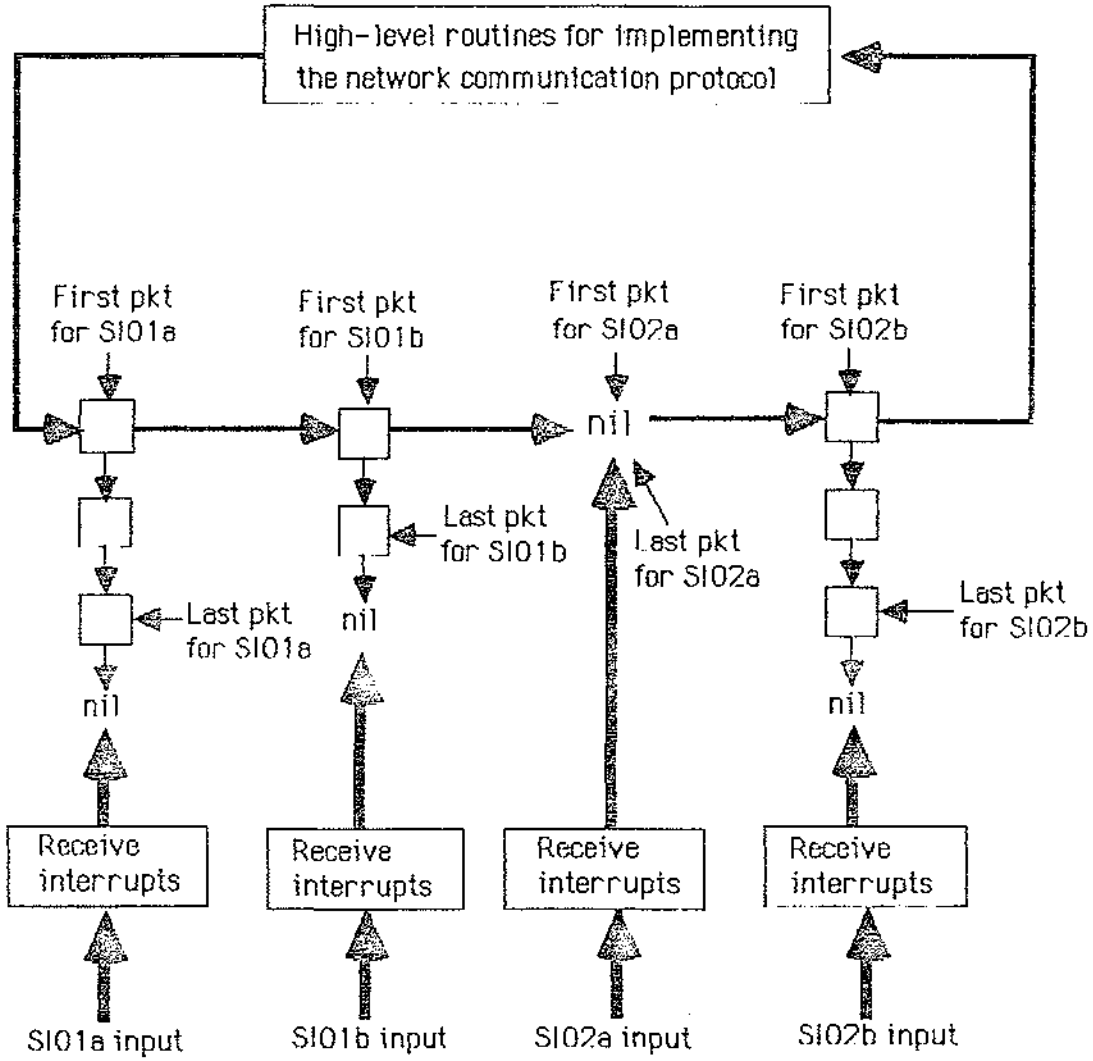


Figure 4.6 Input Queues For Handling Character Receipt Through Each Of The Four Sio Channels.

example, if one virtual circuit has 100 full packets queued for output at a particular SIO, then any subsequent single character packets (belonging to a different virtual circuit) would be appended to the end of that queue. Since the queue is processed sequentially, those single character packets are

delayed for (unacceptably) large periods of time.

As mentioned previously, Woodson [Wood.1982] states that the response to a signal (e.g. feedback to a pressed key) should be set to less than 0.1 seconds to be acceptable to an interactive user. While testing the initial version, typing feedback times of one second and more were occasionally experienced.

The subsequent version of MASSEYNET aimed to reduce queueing delays for short messages at the expense of longer messages. This was achieved by using a separate first-in-first-out queue for each virtual circuit as shown in figure 4.7. The low level output routines service these queues using a round-robin rotation scheme. On each pass, the output routines transmit the first packet in the queue out the required SIO port. This queueing discipline results in consistently short response times because no virtual circuit is permitted to "hog" the channel resources.

Fraser [Fras.1984] presents statistics comparing the single output queueing discipline with the multiple queue, round-robin mechanism. His graphs of mean delay vs. nominal utilisation (user data divided by raw channel rate) for both schemes are shown in figures 4.8(a) and 4.8(b). The single queueing mechanism subjects long and short messages to the

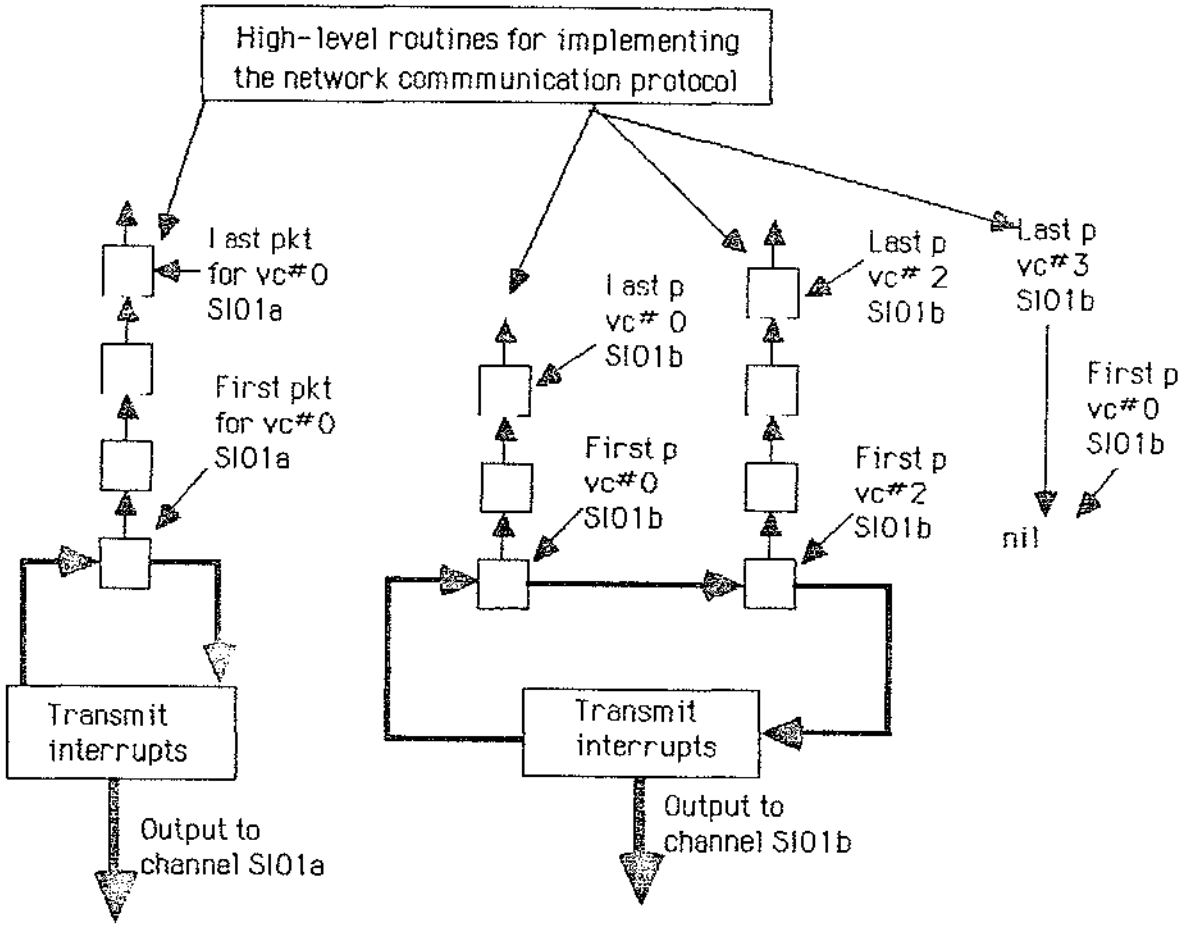


Figure 4.7 Example Of Output Pointer Queues For Sio1a And Sio1b.

same mean queueing delay while the latter technique treats longer messages worse in favour of short messages. Most importantly (particularly in an interactive environment), single character packets experience very low mean delays even as nominal utilisation approaches its limit. Hence, Fraser's statistics support MASSEYNET's progression from single output queues to multiple output queues with round-robin

multiplexing.

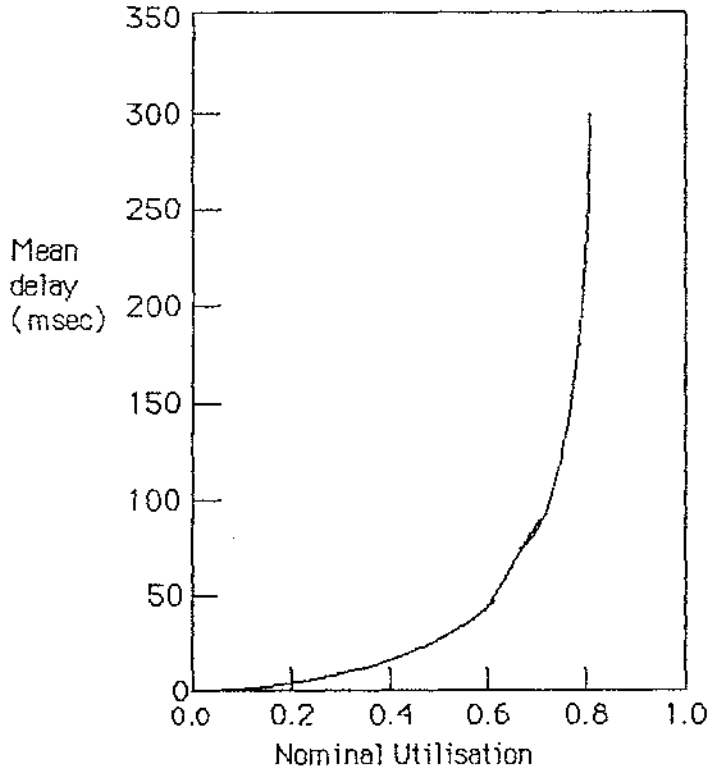


Figure 4.8 a) Mean Queueing Delay For Single Output Queues And A Packet Length Of 64 Bytes With No Overhead Bytes In Any Packet [Fras.1984].

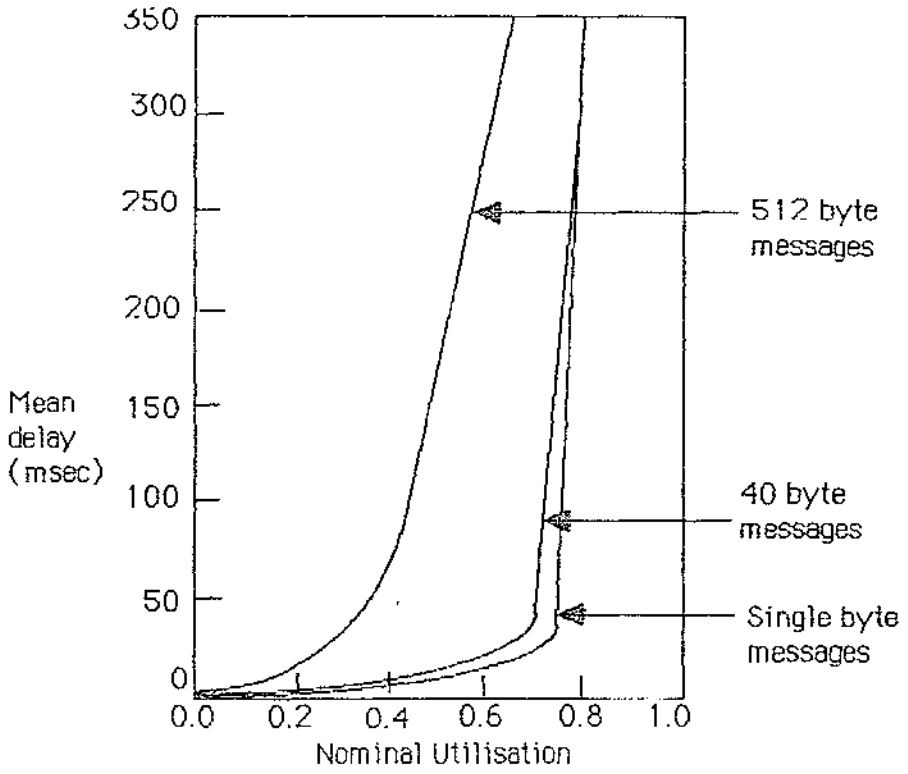


Figure 4.8 b) Mean Queueing Delay Using One Output Queue Per Virtual Circuit And Round-Robin Service. The Packetlength Is 64 Bytes With No Overhead Bytes In Any Packet [Fras.1984].

4.8 Routing Algorithm

The initial version of MASSEYNET used a fixed routing table scheme. As discussed in Section 3.4, each network node maintains a routing table which indicates the best path to each possible destination. This scheme performed quite satisfactorily but was very inflexible. Expansion and contraction were tedious because of the need to change the routing table in every node.

To overcome this lack of flexibility, the current version of MASSEYNET uses a routing algorithm similar to the Discovery process developed for Localnet (Section 2.5). This algorithm floods the network at the beginning of a session to establish a fixed route to the required destination. Subsequently, all communication in that session follows that initial virtual circuit. The resulting system is very flexible and expandable because each node is an autonomous unit and does not need to know anything about the overall network configuration.

The Discovery technique raises several questions as to how the initial flooding process should be controlled and when the resulting virtual circuit should be established. What is the best way to dampen the flooding process? Should a hop counter be used? Should a history of flood packets be kept and subsequent duplicates be discarded? How do we know when

a flood is unsuccessful? Should the flood and call request packets be two distinct packets? Should a recent history table be kept so that the route to commonly accessed addresses is easily obtainable? The solutions to these questions present several different approaches for implementing the Discovery algorithm.

4.8.1 Flood Approach 1

The first approach uses a combined flood/call request packet with a hop counter to dampen the flooding process. At each node, a copy of the flood/call request packet is transmitted out every network port except the one through which it entered. The hop counter is decremented at each node and, when it reaches zero, the call request is rejected. Temporary half-duplex virtual circuits are allocated on the outward flood route and cleared when a call reject packet (Section 4.9) retraces the original flood path. Once a node has allocated all its virtual circuits, all subsequent call requests must be rejected. If a node can satisfy the call request, it transmits a call accept packet back along the original flood route setting up the return half-duplex virtual circuit. This method is simple but does have a large overhead in terms of the number of packets in the network and also virtual circuit allocation. To compound these problems, decisions have to be made regarding the initial value of the hop counter and also the maximum number of virtual circuits

which may be allocated to any one node. Logically, the hop counter must be equal to or greater than the number of nodes in the shortest path between the two most distant nodes (otherwise devices on the network may be "out of reach"). Problems also occur when two or more floods occur simultaneously. If the first flood exhausts the virtual circuit allocation of a particular node, then the second flood will be unable to search that particular node (hence, the flood will fail when perhaps the required destination was available).

4.8.2 Flood Approach 2

The second approach separates the flood and call request packets and maintains a history table at each node. This history table keeps a list of recently accessed devices and the best path to them. To establish a connection, a flood packet emanates through the network requesting a connection to a particular destination. If a flood packet reaches a node which has historical information about that destination, the flooding process is stopped and a controlled search is adopted. If no such information exists, the flooding process continues and is dampened by nodes keeping a copy of recent flood packets and discarding any duplicates that arrive. If this search process finds the required device and it is available, a positive acknowledgement retraces the search path back to the origin node setting up an entry in the

history table of each transit node. On receiving a positive acknowledgement to the flood, the origin node transmits a call request packet along the fixed path defined by the history tables. This packet sets up a virtual circuit connection between the origin and destination devices. If the origin node does not receive a positive acknowledgement to the flood, the call is rejected. The separation of the flood and call request packets presents problems because there is no guarantee that the device which is available at flood time will be available when the call request packet arrives. Some other network user may have connected to it immediately after the flood packet detected it as being available. Recovery from this situation can be difficult because it involves clearing the temporary virtual circuit and re-flooding other parts of the network in search of the required device. Furthermore, if several floods occur simultaneously, information may be lost from the finite length history tables. If this happens then the call request packet will be unable to reach the destination. To prevent this situation, the history table must be large enough to cater for the estimated maximum number of floods occurring simultaneously. It seems therefore that the combined flood/call-request packet used in the first approach results in simpler algorithms than when the two are separated. Keeping copies of recent packets (and discarding subsequent duplicates) dampens the flooding process more quickly than the use of a hop counter because it stops cycling immediately

(instead of waiting for the hop counter to decrement to zero).

4.8.3 Flood Approach 3

The third approach (which is used in the current version of MASSEYNET), uses a combined flood/call request-packet in the initial flood. The flooding process is dampened by each node maintaining a copy of recent flood packets and discarding any duplicate packets which arrive. The flood/call request packet (Section 4.9) emanates from the originating node setting up the outward half-duplex virtual circuits. Each node in turn checks if it is connected to the required device. If it is and that device is available, a call accept packet (Section 4.9) retraces the flood path back to the originating node (setting up the return half-duplex virtual circuit). If the device is not attached to that node, the flood process continues to other parts of the network. If the whole network has been flooded without finding the required device or if the device is not available, then a call reject packet (Section 4.9) is sent back to the originating node and all the virtual circuits are cleared. The drawback with this method is that each connection request requires a complete flood. Ideally, network nodes should keep some history information so that routes to commonly used devices are remembered and the virtual circuit can be

established without flooding. This history table search could be added as a precondition to the flooding algorithm. It is not included in the current version of MASSEYNET but could be added as part of a future project.

The flooding process has been enhanced by two additional features. The first feature is the use of duplicate device names. The second feature is the use of a loading factor to determine the "easiest" path to the required destination.

4.8.4 Duplicate Device Names

Duplicate names are very useful when a particular computer has several lines attached to the network. In the case of Massey University, for example, the network may have several connections to the VAX research computer. A network user wanting to work on the VAX does not care which particular VAX line is used. If each line had a unique name (e.g. VAX1..VAX24), the user might enter 24 separate connection requests, one to each line, only to find that they are all busy. This connection procedure would be time-consuming and very frustrating to any user, particularly if an earlier line became free while later lines were being tested.

MASSEYNET overcomes this problem by allowing duplicate names. More than one device can have a name such as VAX. The network, on receiving a call request for the VAX, uses flooding to look for any available device with that name. Only if all the VAX lines are busy will the call be rejected. If more than one VAX line is available, MASSEYNET uses the loading factor (described below) to determine which one to connect to.

4.8.5 Loading Factor

The flooding process used in MASSEYNET may discover several devices which satisfy the connection request. Because of fluctuating traffic loads, the first available device may not necessarily be the best one to connect to. Some other quantitative measure is needed to determine which path is likely to give the best service. Several criteria were considered.

One of the simplest methods is to measure loading by the number of nodes in each path. This technique is fairly crude because it does not give an indication of the traffic loads along the paths.

Another method required each node to gather statistics on average traffic over a recent time interval. This method gives a clearer indication of network traffic if the traffic loads are fairly consistent. Collecting these statistics is more difficult than the previous method because it involves using timeouts to measure line usage over a fixed period of time.

The approach taken in MASSEYNET uses the number of established virtual circuits as a measure of loading. This method is based on the assumption that all virtual circuits are equally loaded. While this may not necessarily be true, this technique is simpler than the previous technique because it does not require each node to collect time-averaged statistics for each line.

The flooding process in MASSEYNET requires each flood packet to accumulate a loading measure for the path it follows. At every node, the loading factor in the flood packet is incremented by the number of established virtual circuits passing through that node plus one. The extra one is added on so that, if there are no established virtual circuits, the loading measure is equivalent to the number of nodes in each path. If a flood (connection) request is accepted by an available device, the accumulated loading factor is transferred into the resultant call accept packet. As the call accept packet(s) retrace their respective paths, each

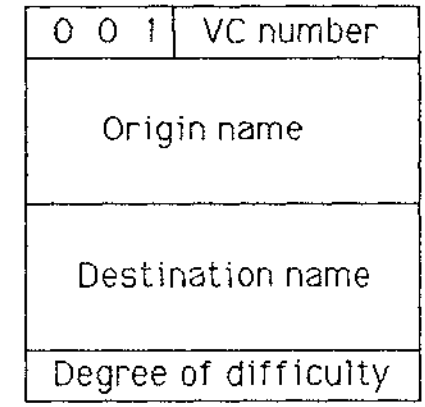
transit node determines the "best" route by comparing the loading factors associated with different paths. If more than one call accept packet arrives at a node, the "best" one is accepted while the other temporary acceptances are cleared. Ultimately, the "easiest" path is chosen and a virtual circuit is established for the duration of the session.

4.9 Packet Formats

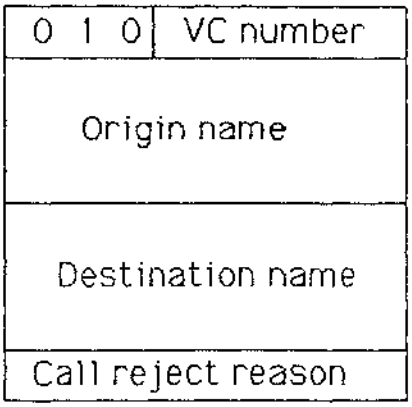
To allow the controlled exchange of information between network nodes, the packets of data must have well-defined structures. In designing the packet structures, it is important to keep the network overheads to a minimum. The packet formats used in MASSEYNET are shown in figure 4.9.

The leading byte of each packet is the control byte used for identifying and routing packets through the network. The first three bits specify the packet type while the last five bits identify the virtual circuit number of the channel through which the packet entered (giving a maximum of 64 virtual circuits between any two adjacent nodes).

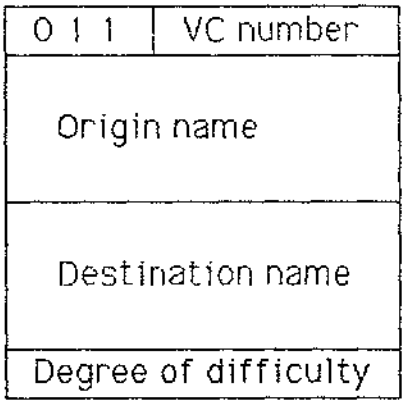
The flood/call request packet is used in the initial flooding process to request each node for the required destination device. The full names of the origin and destination addresses are included in the packet so that nodes can recognise duplicate or cycled packets and so dampen the flood process. As well as the leading byte, this packet also reserves the last byte for accumulating the loading factor for the path it takes.



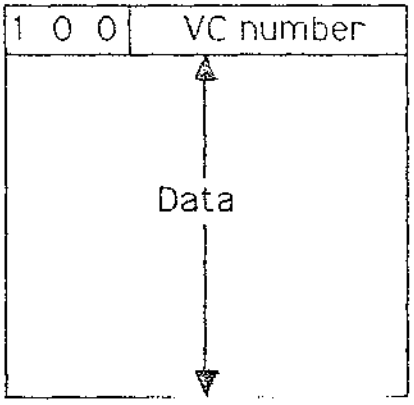
FLOOD / CALL REQUEST PACKET



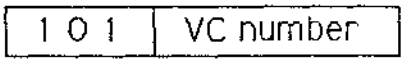
CALL REJECT PACKET



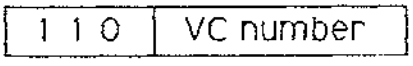
CALL ACCEPT PACKET



DATA PACKET



CLEAR REQUEST PACKET



CLEAR CONFIRM PACKET

Figure 4.9 Packet Formats Used in MASSEYNET

The call accept packet is identical to the flood/call request packet except for the packet type field. This packet is used in response to the flood/call request packet to indicate that the required destination is available.

The call reject packet is similar to the previous two packet types but is used to indicate that a connection cannot be established with the required device. The last byte in the packet contains the reason for rejecting the call. A value of 0 implies that the device does not exist while a value of 1 indicates that the device is not available (i.e. busy).

The data packet is the most frequently used structure because the other five packet types are only used when setting up and clearing a connection. Consequently, it was important to restrict its format to a single byte overhead. Its leading byte simply contains its type and the virtual circuit number of the channel along which it is to be sent. Data packets may be variable in length up to a fixed maximum size and, as with all packets, they are terminated with an ETX character (Section 4.3).

The clear request and clear confirm packets are used at the end of a session to clear the two half-duplex virtual circuits between the origin and destination. The clear request packet clears the origin-to-destination half-duplex virtual circuit while the clear confirm packet clears the

destination-origin half-duplex virtual circuit.

4.10 Maximum Packet Size

In MASSEYNET, packets may vary in size up to a fixed maximum length. This maximum length must be conducive to low delay to ensure quick response times for single character messages and high throughput so that large volumes of data can be transferred through the network as quickly as possible. To achieve low delay, the maximum packet size should be small so that output queueing delays at each node are minimised. To achieve high throughput, large packet sizes should be used so that very little channel bandwidth is wasted by the overheads of the packet control bytes.

Obviously, the requirements of the two message types conflict and so some compromise must be made. Since MASSEYNET places special emphasis on the demands of interactive computing, it uses a reasonably small maximum packet size of 12 bytes (characters). To satisfy Woodson's 0.1 second feedback criterion [Wood.1982], the queueing delay through the network (using 9600 baud lines) must not exceed the time taken to transmit 96 asynchronous characters (approximately eight 12-character packets). With the round-robin transmission scheme and the low utilisation of interactive lines, such a queueing delay is unlikely. Factors affecting these queueing delays include the path length and the number of established virtual circuits on each line. As well as providing low delay, the 12-character packet size means that the overhead

introduced by the control byte in a data packet is less than 10%. This allows high throughput for longer messages.

4.11 Error Control

Error control involves the elimination of transmission errors in the lines and recovery from lost or duplicate messages or parts of messages. The error rate in local area networks is much lower than in wide area networks because they use dedicated, reliable cables instead of using error-prone public telephone facilities.

The X25 recommendations include detailed control and error checking information for communication over the error-prone telecommunications cables used in wide area networks. The line transmission errors are detected by including redundant information in each packet. Using this information, the receiver can detect bit errors and request that the relevant packets be retransmitted. The problem of lost or duplicate packets is overcome by including sequence numbers in the control fields of each packet.

These error control procedures work well in wide area networks where packet sizes are large and response times are not critical. In a local interactive environment however, a lot of traffic consists of single character packets and so each extra error-control character increases the control overheads by 100%. Even with maximum sized packets (presently set at 12 characters), network throughput is reduced significantly by the inclusion of error detecting

codes. Retransmission of packets also requires quite complex processing routines which may result in the Z80 nodes being processor bound. Hence, instead of building protocols for handling the few errors that do occur, MASSEYNET leaves error detection and correction up to the users.

4.12 Flow Control

MASSEYNET does not currently support any form of flow control. The network has been developed and tested in the Computer Science Department at Massey University where all lines operate at 9600 baud. Consequently, it was decided that flow control could be omitted from the initial version of MASSEYNET. Subsequent versions of the network have led to refinements in the queueing and routing algorithms but flow control routines have yet to be implemented. A separate project involving the development of hardware using line-level flow control signals is under way. These signals can be used to prevent an SIO from transmitting data when the buffer space at the receiving node has been exhausted.

4.13 Congestion Control

Another early design decision was the assumption that congestion would not be a problem. Since the buffer space at each node is large compared to the expected volumes of traffic, this seemed to be a reasonable assumption. If congestion does occur, the excess packets are simply discarded and recovery is left to the network users. This crude technique avoids deadlock because the network nodes transmit data regardless of the resource availability at the adjacent node.

Obviously, this may not be a valid assumption in other environments. Before MASSEYNET can be considered as a general networking solution, some control mechanism must be included which prevents or recovers from lost packets due to congestion within the network. This would allow large files to be transfered from one computer to another across the network.

4.14 Network Initialisation

Initialisation of each network node is very quick and simple. The network program currently resides on the Prime computer and must be downloaded into each Z80 communications box. Having downloaded the program, the user is prompted to enter the name of the device connected to each of the four RS232 ports (as shown in figure 4.10). At present, device names are restricted to a maximum of five characters although this is a program constant which can be changed. After entering the maximum number of characters, subsequent characters are discarded and are not echoed to the screen. The interface allows the use of the backspace (ASCII 8) and delete (ASCII 127) keys to correct typing errors and the device names may be entered in any combination of upper and lower case characters. If no name is entered in response to a prompt, the program assumes that the corresponding port is connected to another network node. After entering the four device names, the various queues and tables are initialised. On completion, the message "Network initialisation is complete." appears on the screen as shown in figure 4.10.

```
Z80-BASED PACKET SWITCHING NETWORK
==================================

Enter lower right port name (no name implies a network port) ==> VAX
Enter lower left  port name (no name implies a network port) ==> PRimE
Enter upper right port name (no name implies a network port) ==> TONY
Enter upper left  port name (no name implies a network port) ==> Paul

Network initialisation complete.
```

Figure 4.10 Example Of Network Initialisation

4.15 Network User Interface

The network user interface is also very simple to use. Connections are established and terminated by pressing the break key. An example session is shown in figure 4.11.

In response to a break, the network prompts the user to "Enter device to connect to?". As with network initialisation, the name of the required device may be entered in any combination of upper and lower case characters with corrections being made with the backspace and delete keys. If the name exceeds the maximum length, an error message is printed as shown in figure 4.11.

After the user has entered the requested destination name, a call request packet is formed and flooded into the network. If the call is accepted, the prompt "You are connected." appears on the screen. If however all devices with the required name are busy, the prompt "Sorry, device is busy." is displayed. If the device does not exist, the network replies with "Error, cannot find device.".

```
<break>
Enter device to connect to? VAX
You are connected.

<interactive session with VAX>

<break>
Connection is terminated.


<break>
Enter device to connect to? Bill
Error cannot find device.

<break>
Enter device to connect to? PriME
Sorry device is busy.

<break>
Enter device to connect to? Margar
Error name is too long.

<break>
Enter device to connect to? Altos
You are connected.

<Interactive session with Altos>

<break>
Connection is terminated.
```

Figure 4.11 Example Of Network User Interface

Connections are terminated by sending a break to the attached network node. This break requests the network to clear the two half-duplex virtual circuits. When this has been done, the prompt "Connection is terminated." appears on the screen.

CHAPTER 5

RESULTS

5.1 Version 1

The initial version of MASSEYNET featured single output queues (Section 4.7) and used a fixed routing scheme with single character address names. The network topology was linear and gave five interactive users access to five computers via two internal patch panels (as shown in figure 5.1).

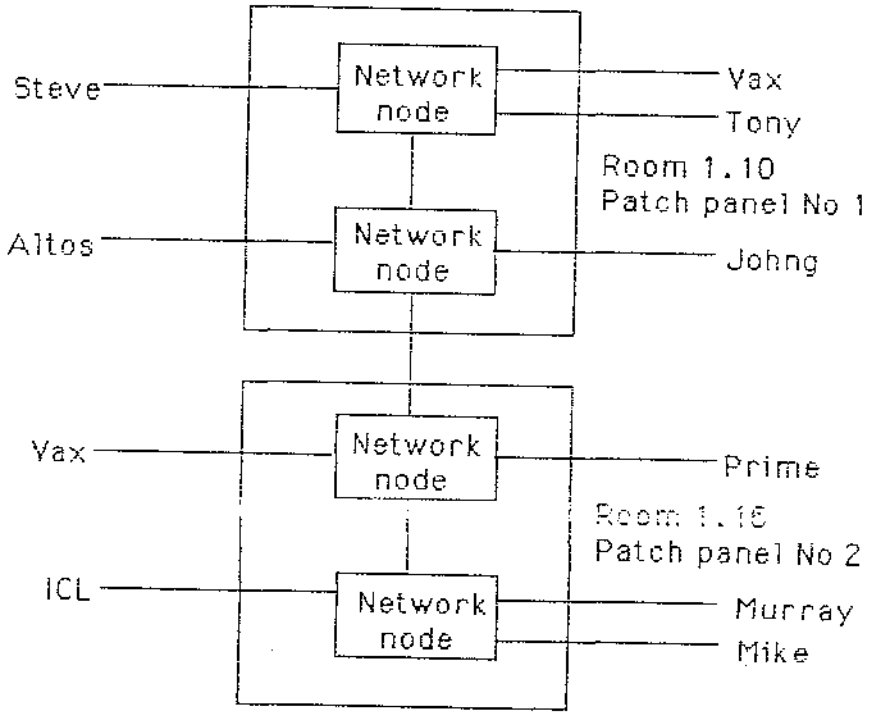


Figure 5.1 Initial Network Configuration

The network operated successfully but, occasionally, unacceptably high response times were experienced during high usage periods. This highlighted the need for a more efficient queueing mechanism for sharing a single channel amongst several virtual circuits. The test run also highlighted the disadvantages of the fixed routing scheme. Apart from its lack of flexibility, network initialisation was a tedious process because each node has a different relative view of the network topology; hence, each node must be loaded with a different routing table. The test run supported the omission of flow, congestion and error control because the network had no difficulty in handling the interactive workloads of the Computer Science Department. Information was only lost when devices transmitted at full speed (9600 baud) for an excessive period of time (2 to 3 minutes depending on the number of nodes in the pathway). The results of this first test run stimulated refinements to the queueing discipline and to the routing scheme.

5.2 Version 2

The current queueing discipline (described in Section 4.7) maintains a separate output queue for each virtual circuit and uses round-robin multiplexing to give each circuit its fair share of the channel capacity. This mechanism was implemented in the second version of MASSEYNET and tested in the same network configuration as the initial version. The observed results show a marked improvement in response times for single character packets with typing feedback appearing to be instantaneous.

These network delays were analysed quantitatively through two linear network configurations, one consisting of three nodes, and the other consisting of four nodes. Network delay was measured as the elapsed time between the transmission of a single character at one end of the network and its receipt at the other end. Samples of 100 single character transmissions were collected with

(a) the networks unloaded
(b) the networks supporting one heavy load (i.e. an end-to-end virtual circuit handling continuous 9600 baud traffic)
(c) the networks supporting two heavy loads (i.e. two end-to-end virtual circuits handling continuous 9600 baud traffic)

The average elapsed times and associated standard deviations for each sample are tabulated in figure 5.2.

	Networks unloaded	1 end-end 9600 baud load	2 end-end 9600 baud loads
Linear three node network	14.46 (.34)	30.09 (3.75)	39.50 (3.95)
Linear four node network	18.40 (.31)	44.87 (3.73)	59.81 (7.97)

Note : All times are in micro-seconds.
Standard deviations are shown in brackets.

Figure 5.2 Table Of Average Network Delay (And Standard Deviation) For Samples Of 100 Single Character Transmissions

To obtain single character feedback times, the elapsed times for the forward and reverse virtual circuits must be added together. Hence, for the tabulated cases, the 0.1 second maximum typing feedback recommended by Woodson [Wood.1982] is only exceeded in the case of a four node virtual circuit subject to two heavy loads in one direction, and at least one heavy load in the reverse direction. The likelihood of this occurring in practice is heavily dependent on the size and topology of the network, and the expected network traffic loads. Since interactive computing typically results in low utilisation

of the available channel bandwidth, mesh networks of 20 to 25 nodes (40 to 50 devices) could be expected to handle the demands of Massey's Computer Science Department, provided that devices are separated by no more than five or six nodes.

5.3 Version 3

The third and current version of MASSEYNET (presented on micro-fiche at the back of this thesis) replaced the fixed routing scheme with a routing algorithm based on flooding (Section 4.8). This move has resulted in a very flexible network which handles initialisation, expansion and contraction easily. The routing algorithm does not require each node to know the overall network topology and so initialising each node is very simple (as shown in Section 4.14). It takes 10 seconds to download the network program into a node and, after the four device names have been entered, network initialisation takes less than one second. The time taken to establish a connection varies according to the network size and topology, the network loading at that instant, and the location of the required device (e.g. if the destination device is attached to the same node as the user and it is available, then there is no need to flood the network). With the trial configuration shown in figure 5.3, the users observed that most connections were established within an extremely short time interval (i.e. less than 0.5 seconds) and terminated instantaneously (i.e. no detectable time delay).

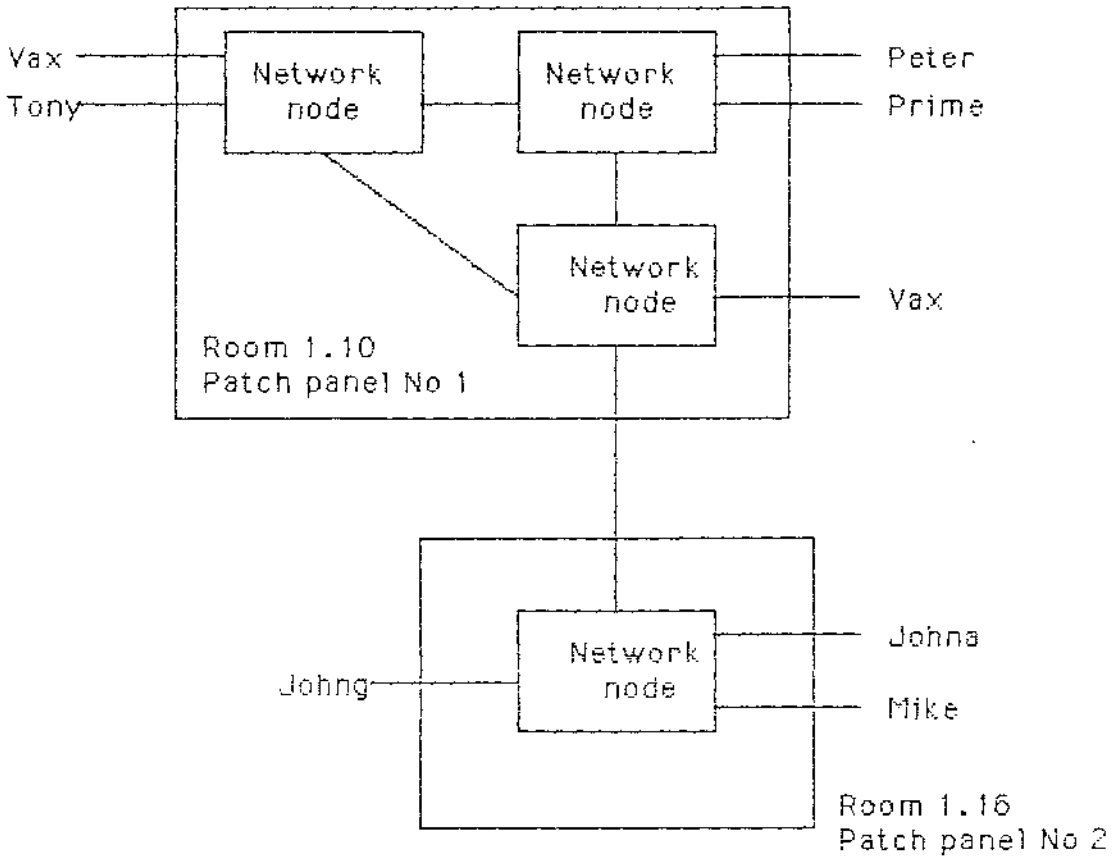


Figure 5.3 Second Trial Configuration

5.4 Drawbacks of Routing Algorithm

The major drawback of the current routing algorithm is its lack of robustness. The flooding process is suspended at every node until a reply to each subsequent flood is received. If a node goes down or a cable is disconnected, a reply might never arrive and so the flooding process is suspended indefinitely. Consequently, the routing algorithm will never work on a partially complete network. Even at network initialisation time, the network configuration must be complete before connections can be established. Hence, although network initialisation, expansion and contraction are performed easily, the rest of the network must refrain from establishing connections while the network is incomplete.

The use of duplicate device names in the network will cause the routing algorithm to fail if two users with the same name attempt to connect to the same device (e.g. if two users with the name "Helen" attempt to connect to any "Vax" line). The resulting flood/call request packets will be identical and network nodes will reject the second one as being a duplicate. Hence, the second connection request may fail. This situation is overcome by not allowing devices with a duplicate name to establish connections. This ensures that the requesting device

name is always unique and hence, the flood/call request packet will also be unique.

Furthermore, if a connection between one "Helen" and a "Vax" line has been established and the other "Helen" subsequently requests a Vax connection, the first connection may be "stolen" by the second "Helen". When the first connection is established, the destination node remembers the quantitative loading factor of the established path. During the session, if a subsequent Helen-Vax request of lower difficulty arrives, then the node clears the old path and establishes what it considers to be an easier path between the source and destination nodes. If this subsequent call request came from the second "Helen", then the connection is effectively "stolen" from the first "Helen". Once again, this problem can be overcome by not allowing devices with a duplicate name to establish connections. Hence, all flood/call requests will be unique to the initiating device.

5.5 Comparison with Network Objectives

Each version of MASSEYNET has been tested in a trial configuration within the Computer Science Department. The refinements to each version have brought it closer to the ideal objectives listed in Chapter 2. The current version of MASSEYNET satisfies the requirement for a simple network interface which provides network-transparent single-byte communication between any two devices attached to the network. It provides a significant improvement in security over the patch panel communication system and is flexible enough to allow easy initialisation, expansion and contraction. In most configurations, each network node supports on average two network devices. The cost of each node is approximately $NZ280 (including cabling) and so each device can be connected to the network for a cost of $NZ140. This is cheap when compared to the Ethernet connection cost ($US350 to $US500) and is also small in comparison with the purchase price of all the computer equipment used in the Department (the cheaper terminals cost between $NZ1000 and $NZ2000). The current version of MASSEYNET is not robust and the network does not degrade gracefully. However, the network could satisfy both these criteria with the addition of a timeout mechanism to the flooding algorithm (as discussed in the next chapter). Further developments could also extend the user interface to

facilitate queueing of connection requests and other application-oriented functions.

5.6 Conclusion

The three network versions resulting from the iterative design and implementation of MASSEYNET have been progressively tested over five months in the Computer Science Department. The live testing exposed several programming errors but, once these were corrected, the network catered successfully for the demands of the interactive users. More importantly, the design and implementation of MASSEYNET has shown that the packet switching techniques commonly associated with wide area networks are well suited to interactive traffic. In addition, the distributed topology of wide area networks overcomes the problems of flexibility, security and cost associated with the centralised approach of existing local area networks. Hence, the networking techniques incorporated in MASSEYNET form the basis of what promises to be a viable networking solution for a variety of local area environments.

CHAPTER 6

FUTURE DEVELOPMENTS AND CONCLUSIONS

The three versions of MASSEYNET have worked well in testing but there are several further developments which need to be completed before it could be successfully used as a practical networking solution. Suggestions for further developments are discussed below.

6.1 Routing Algorithm

Perhaps the biggest drawback of MASSEYNET is its lack of robustness. As described in the previous Chapter, the routing algorithm assumes that packets will never be lost and so waits indefinitely for a reply to each flood packet. Consequently, the routing algorithm will never work on a partially initialised network or on a network where nodes are down or lines are disconnected.

To make the network robust, the routing algorithm should incorporate a timeout mechanism at each node so that the flooding process is not suspended indefinitely. If, after a period of time, an adjacent node has not responded to a flood request, the node should time out and assume that the adjacent node is down or the packet has been

lost. Subsequently, it sends a reply back towards the originating node and the connection establishment continues as usual. Ideally, only those nodes at the end of each path should time out. To enforce this, the timeout value needs to decrease for nodes further away from the originating device. The size of the timeout value needs to be small enough to reduce network delay but large enough so that each reply packet has ample time to arrive at its appropriate node. The use of a timeout would make the network extremely flexible and robust. It would handle expansion and contraction without disrupting the existing network and the routing algorithm would work regardless of node and line failures.

6.2 Duplicate Device Names

As discussed in Chapter 5, the use of duplicate device names introduces problems if two separate but identical connections are requested. The problem arises when a previously established virtual circuit is cleared in response to a subsequent call request packet to make way for an easier path. This problem could be eliminated by setting the loading factor for established connections to zero. Since this is the lowest possible loading factor, the established virtual circuit will always be "easier" than circuits established by subsequent call requests. Another approach is to use this feature to provide a system which changes virtual circuits dynamically in

response to fluctuating network loads. The path between each source and destination could be periodically analysed (by flooding) to update the path according to recent traffic statistics.

6.3 Hardware

The Z80-based communication nodes used in MASSEYNET drive four RS232 ports. In most network configurations, each network node is connected to two other nodes leaving 2 ports to which terminals and computers can be attached. Hence, on average, one box is required for every two network devices. In a network of (say) 100 devices, the system of 50 boxes would be unwieldy to organise and set up. A more suitable solution would be to develop a box which supported six or even eight RS232 ports. In this way, far less boxes would be needed to interconnect the same number of devices and yet, the boxes would still facilitate incremental growth (unlike the nodes in Mininet).

6.4 Automatic Loading

The program for controlling network communication requires 6.4K of memory and so cannot be stored in the 4K of ROM in the Z80-based communication nodes. At present, the network program resides on the Prime minicomputer from which it is loaded into each box. This loading process is subject to the availability of a Prime line (of which the Department has only two).

A project is currently under way to develop an automatic loading program which transfers a program from one Z80 box to another. This loading program could be stored in ROM and executed each time a new node is added to the network. When reset, the new node will send a special control code through each of its ports requesting each of its neighbours for the program. If an adjacent network node receives this request, then it transmits its program into the new box. In this way, network initialisation is not dependent on the availability of a Prime line. Network expansion proceeds automatically by adding a new node to the appropriate network position and resetting it.

6.5 Flow, Congestion and Error Control

As discussed in Chapter 4, the current version of MASSEYNET does not include flow or error control and it simply discards packets during periods of congestion. The testing of MASSEYNET has supported the omission of these control mechanisms because the network has had no difficulty in handling typical workloads within the department. If the messages are lost or errors occur, recovery is left to the users. This system may be satisfactory in an interactive environment, but other environments (especially those involving direct computer-to-computer communication) require a network to provide an error-free, controlled transfer of data between any two devices. If MASSEYNET is to be adaptible to these environments, then future developments should involve the implementation of some or all of these mechanisms. The complexity of these control schemes may be restricted by the processing power of the Z80 nodes.

6.6 Conclusion

This project has involved the design, implementation and testing of a local area network using techniques previously associated only with wide area networks. This research was stimulated by the networking requirements of the interactive computing environment at Massey University and the failure of existing local area networks to satisfy the required list of objectives. The resulting system, called MASSEYNET, uses the wide area packet switching techniques to produce a flexible, distributed system which establishes, controls and terminates links between any two network users. MASSEYNET uses a variation of the pure flooding algorithm to discover the best route to a particular destination. Having found the best path, two half-duplex virtual circuits are established between the origin and destination nodes for the duration of the session (one virtual circuit in either direction). Each node maintains separate output queues for each virtual circuit and uses a round-robin multiplexing scheme to share the output line evenly amongst them. The resulting network has worked well during testing with response times being comparable to a dedicated connection. Further development is required however before MASSEYNET becomes a practical networking alternative. By introducing a timeout into the initial flood process, the network would be very robust because the routing scheme would work with

partial network configurations which may result from node failures, line disconnections, or incomplete initialisation. To be adaptible to other networking environments, MASSEYNET should also incorporate flow, congestion and error control mechanisms.

This project has shown that packet switching techniques are well suited to an interactive local area environment. Using small packet sizes, response times through the network are comparable to those of a dedicated terminal-computer connection. In addition, the distributed mesh topology of wide area networks can be successfully adopted in a local area situation to provide a very flexible, robust network, with the capacity for incremental growth and indefinite expandability.

BIBLIOGRAPHY

Ahuj.1982 - Ahuja,Vijay

Design and Analysis of Computer Communication Networks

McGraw-Hill Inc, 1982

Arth.1983 - Arthurs,E., Chesson,G.L., Stuck,B.W.

Theoretical Performance of Virtual Circuit LAN Sliding Window Flow Control

Bell Laboratories, Murray Hill, New Jersey, 1983

Bass.1983 - Bass, Charlie

Local Area Networks - A Merger of Computer and Communications Technologies

Systems Magazine, March 1983

Blea.1979 - Bleazard,G.B.

Why Packet Switching?

NCC Publications, 1979

Blea.1982 - Bleazard,G.B.

Handbook of Data Communications

NCC Publications, 1982

Cars.1982 - Carson,George S.
Message-Based Distributed Computing
GSC Associates, Hawthorne, California

CCIT.1981 - The International Telegraph and Telephone
Consultative Committee (CCITT)
Data Communication Networks
Services and Facilities,
Terminal Equipment and Interfaces
Recommendations X.1-X.29

Deas.1982 - Deasington,Richard
A Practical Guide to Computer Communications
and Networking
"Computers and Their Applications",
Ellis Horwood Publishers, 1982

DeMi.1983 - DeMillo,Richard and Merritt,Michael
Protocols for Data Security
Georgia Institute of Technology, February 1983

Detr.1983 - DeTreville,John and Sincoskie,W.David
A Distributed Experimental Communications
System
IEEE Journal on Selected Areas in
Communications, Vol.SAC-1, No.6,
December 1983

Fras.1983 - Fraser,Alexander G.
Towards a Universal Data Transport System
IEEE Journal on Selected Areas in Data Communications, Vol.SAC-1, No.5, November 1983

Fras.1984 - Fraser,A.G., and Morgan,Samuel P.
Queueing and Framing Disciplines for a Mixture of Data Structures
A.T.&T. Bell Laboratories, Murray Hill, New Jersey, U.S.A.

Gee.1982 - Gee,K.C.E.
Local Area Networks
NCC Publications, 1982

Harb - Harbitter,Alan, and Tripathi,Satish K.
A Model of Transport Level Flow Control
Department of Computer Science, University of Maryland

Hals.1982 - Halsall,F., and Ruela,J.A.
Prototype X25 Exchange for Use in Local Area Networks
Software and Microsystems, Vol.1, No.3, April 1982

Hayw.1985 - Hayward,R.G., and Kay,P
Report on Visit to U.S.A.
Massey University, September 1985

Hutt.1981 - Hutty,Roger
Z80 Assembly Language Programming for Students
Macmillan Press Ltd, 1981

Kali.1984 - Kaliszewski,Jean Michel
The New Routing Algorithm on the SITA Data Transport Network
S.I.T.A., France, 1984

Kak.1983 - Kak,Subhash
Data Security in Computer Networks
Guest Editor's Introduction, Februrary 1983,
Louisa State University

Logi - Logica Limited
X25

Lyon.1985 - Lyons,P.J., and McGregor,A.J.
Preliminary Specifications for a Data Communications Exchange
Massey Computer Science Report 85/7,
October 1985

Micr.1981 - Micro-Professor
MPF-I Experiment Manual (Software/Hardware)
Experiment 15, Clock 2 (With CTC Interrupt Mode 2)
Multitech Industrial Corporation, 1981, Fifth Edition

Mitc - Mitchell,Lionel C.
Optimization of Maximum Packet Size in a Local Area Network
Control Information Systems, Fairfax, VA

Morl - Morling,R.C.S., and Cain,G.D.
A Routing Protocol That Maintains Packet Sequency
Polytechnic of London

Neri.1984 - Neri,G., Morling,R.C.S., Cain,G.D., Faldella,E., Natali,P., Longhi-Gelatti,M., Salmon-Cinotti,T.
MININET: A Local Area Network for Real-Time Instrumentation Purposes
Computer Networks 8, North-Holland Publishing Company, 1984

Peti - Petitpierre,C.
Contention-Free Local Area Network Technique With High Channel Utilisation
Laboratoire d'Informatique Technique,
Ecole Polytechnique, Federale de Lausarine,
Lausanne, Switzerland

Pulf.1984 - Pulford,W.C.A., and Lawrence,R.T.
A Simple Multiplexor for a Cambridge Ring
Journal of Microcomputer Applications (1984),
Vol.7, pp 139-147, Academic Press Inc.
(London) 1984

Pung.1984 - Pung.H.K., and Davies,P.A.
Data/Voice Integration for Fibre Optic LANs with Arbitrary Topology
Computer Communications, Vol.7, No.5, 1984

Radi.1978 - Radio Shack TRS-80
Editor Assembler User Instruction Manual
Tandy Corporation, 1978

Reag.1984 - Reagan,Philip H.
Is it the PBX or the LAN?
Datamation, March 1984

Rosn.1976 - Rosner,Roy D., and Springer,Ben
Circuit and Packet Switching
A Cost and Performance Tradeoff Study
Computer Networks 1, North-Holland Publishing Company, 1976

Soln.1980 - Solntseff,N.
A Distributed Operating System for an Educational Micro-computer Network
Unit for Computer Science,
McMaster University, Hamilton,
Ontario L8S 4K1, July 15 1980

Suns.1983 - Sunshine,C., Kaufman,D., Ennis,G., Biba,K.
Interconnection of Broadband Local Area Networks
Sytek Inc., Los Angeles Operations,
Culver City, CA

Tane.1981 - Tanenbaum, Andrew S.
Computer Networks
Prentice-Hall International Inc., 1981

Tele.1982 - Telecoms
Packet Switching, Facts and Facilities
Post Office Headquarters, Wellington, 1982

Thar.1982 - Thareja,Ashok K.
Buffer Allocation Policies For Message Switching Networks
Systems Design and Analysis Group,
Department of Computer Science,
University of Maryland, Maryland

Wood.1982 - Woodson
Human Factors Design Handbook
1982

Zilo - Zilog Inc
Product Specification Z80/Z80A CTC
Zilog Data Book

Zilo - Zilog Inc
Product Specification Z80/Z80A SIO
Zilog Data Book