A Gateway between XLNET and TMS-IBM Token Ring

A Thesis

Submitted to the University of New South Wales

Kensington, New South Wales, Australia

as

Requirement for the Degree of

Master of Engineering

by

Yi-Chun Wang

June, 1990

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of a university or other institute of higher learning, except where due acknowledgement is made in the text of the thesis.

Signed

ACKNOWLEDGEMENTS

The research program reported here was carried out in the Department of Communication of University of New South Wales under the supervision of Professor A.E. Karbowiak and Associate Professor P.L. Chu. I would like to express my appreciation and thanks for their guidance and encouragement throughout the project.

I am also grateful to the pleasant working atmosphere created by my friends, Paulos Nyirenda, Weiming Li and Rhonda Chuang. Special mention goes to Hassan Mehrpour, Frank Lui and Thomas Fong for their valuable assistance and advice in this project.

Finally, I would like to express my profound thanks to my parents for their loving support and understanding over the past two years; without them this work would never have been possible.

ABSTRACT

The difficulty of interconnecting networks when they obey different architectures is an important inhibitor to computer communication growth and flexibility. Therefore, a suitable link device must be required to solve interconnection problems of dissimilar networks. The aim of this thesis is to perform a theoretical and experimental investigation on a linking device (gateway) employed in the two heterogeneous local area networks (LANs), XLNET and TMS-IBM Token Ring.

To do this, we first review some important aspects of the two LANs considered to be connected. These two LANs have dissimilar layer protocols we find up to the Session Layer. In our research, neither a bridge nor X.75 protocol would be a suitable communicator, except that a gateway provided with the function of protocol conversion, is able to achieve the peer-to-peer communication in this interconnecting system. Therefore, the gateway is employed in our project. Moreover, by the shared memory scheme installed at the Network Layer of a gateway, we can limit the protocol conversion just to the Transport Layer and Session Layer. By this means, the cost on the conversion in the rest of layers is saved and the speed of the internet packets transmission can be increased.

The design of the gateway including hardware and software is presented in this thesis. Furthermore, the performance of the gateway are analyzed. A simplified queueing model is adopted to yield approximate expressions for the mean delay time and throughput. The laboratory results are compared with the computed ones, and close agreement is observed.

CONTENTS

APPENDICES

REFERENCES

CHAPTER ONE

INTRODUCTION

1.1 The Need for a Gateway

Over the past few years, there has been an increasing interest in the design and construction of local area networks (LANs). Some networks are incredibly fast and other are comparatively slow; some provide powerful functions in some fields, and some do not provide them at all. With the increasing use of LANs, the requirement for LAN interconnection arises to overcome limitations in distance, and capability of supporting a huge number of hosts. Hence the need for LANs interconnection.

An analogue of interconnecting device is offered the Harbour Bridge in Sydney shown in Fig.1.1. The type of a proper linking device depends very much on the protocol system of each LAN considered. A gateway is a communications link between two dissimilar networks (here XLNET and TMS-IBM Ring). Among others, a gateway translates protocols of source LAN to that of the destination LAN. This thesis is concerned with details of protocol translation, as well as addressing and routing of the messages between two laboratory LANs: XLNET and TMS-IBM Token Ring. The basic model necessitates buffering of messages for the two direction of the system. (Fig.1.2). The architecture, operating principles and performance of the gateway are also presented.

Cunard's 'Saga Rose' passing beneath Sydney Harbour Bridge

Fig. 1.1 The Symbol of a Gateway

The Sydney Harbour Bridge is used here as an example illustrating the gateway function of linking the north part of the Sydney metroppolitan area with the south.

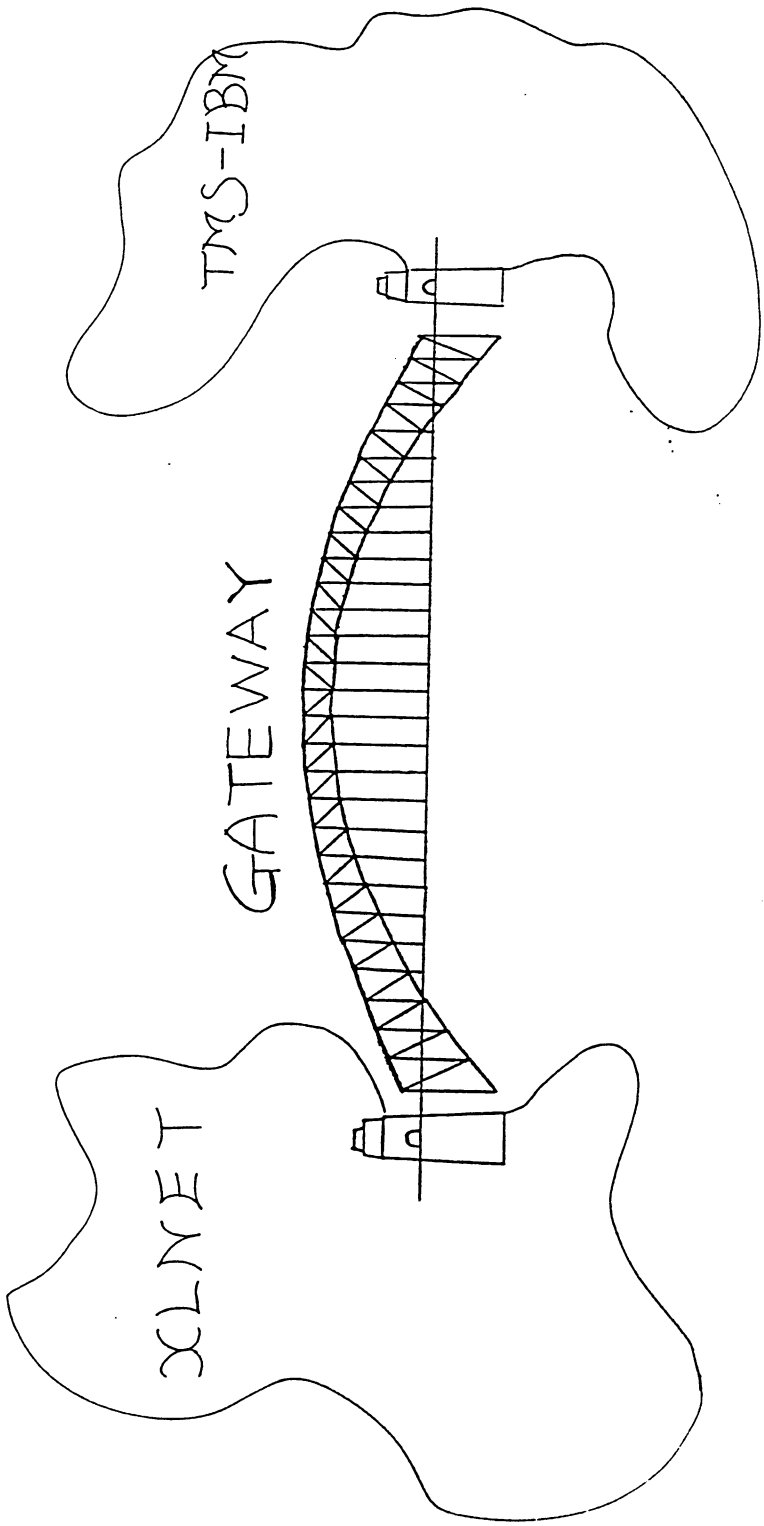


Fig. 1.2 The Gateway between XLNET and TMS-IBM

In that follows, we discuss, briefly, the various necessary functions in a gateway by reference to the Open Systems Interconnection (OSI) model.

1.2 The OSI Model

The OSI Reference Model[1], developed jointly by the International Organization for Standardization (ISO) and the International Telegraph and Telephone Consultative Committee (CCITT), specifies the architecture for open communications. It also assigns functionality to each layer, and identifies how they interrelate. The model consists of seven layers as shown in Fig.1.3.

These seven layers begin at the Application layer, where end users interact with a system or where application software is executed. The user information is eventually passed to the Physical Layer, where logical information is converted into signals that are transported through physical media.

Actually, in the reference model, each layer in the system is contained in a subsystem. Each subsystem contains entities that provide services for a subsystem of higher rank in the model. The network supports entities in a subsystem which together form a layer. Peer-to-peer communications between the entities in a given layer are done in accordance with specific rules defined by ISO as protocols. Entities in one layer communicate with entities in another layer through well defined service access points using specific interface protocols.

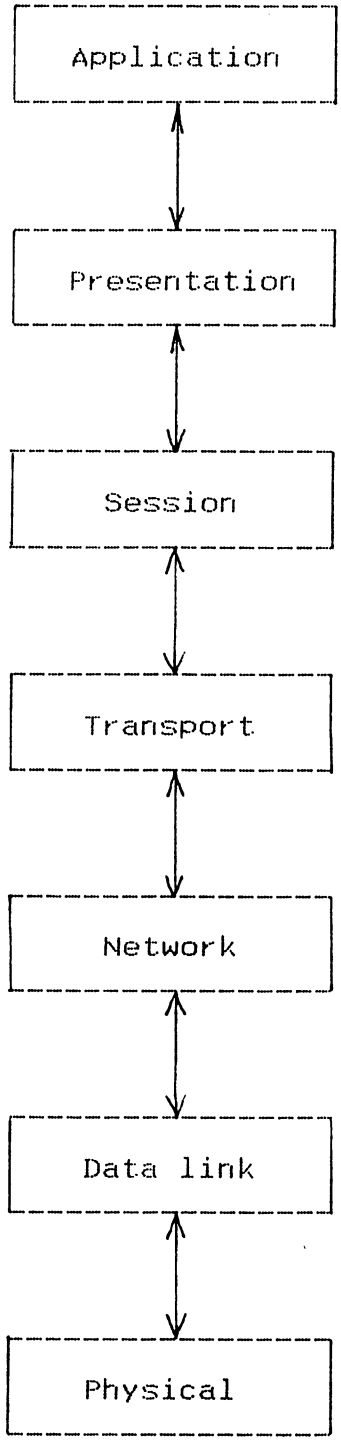


Fig. 1.3 The OSI layered network model

A brief description of each layer is given below:

1. Physical layer: the physical layer, the lowest layer in the model provides for transmitting raw bits across a communication channel. The responsibilities of this layer are for activating, maintaining and deactivating the physical path. The simplest example is the RS-232-C standard.

2. Data link layer: the data link layer is responsible for error free transfer of data between the nodes and network. It provides for the creation and recognization of frames, their detection and possible correction on account of transmission error. A well known example of a standard for this layer is the High Level Data Link Control (HDLC) protocol.

3. Network layer: the network layer handles the routing functions for data transferred between two open systems. It ensures that packets are correctly received at their destinations and in the proper order. The layer is the highest OSI level supported by some communication networks. The best known standard for this layer is CCITT X.25 providing a packet switched network interface.

4. Transport layer: the transport layer provides the interface between the data communications network and upper three layers. It is responsible for establishing and maintaining transport connections between session entities. The transport layer standards have been developed to provide reliable data transfer, flow control required on an individual basis. The ISO standard ISO8073 specifies the transport protocol designed to perform these functions.

5. Session layer: the session layer provides the functionality to establish and manage a dialogue between communicating end systems. It is dependent upon the transport layer in the sense that each session connection is handled by one and only one transport connection. The ISO standard ISO 8327 specifies the basic session protocol and related options.

6. Presentation layer: the presentation layer provides for the syntax of data in the model, that is, the representation of data. It is to accept data types (character, integer) from the application layer and then negotiate with its peer layer as to the syntax representation. The ISO standard ISO 8823 specifies the basic protocol.

7. Application layer: the application layer is concerned with support of an end user application process. Unlike the presentation layer, this layer provides for the semantics of data. The layer provides service to the users of OSI, not to a next high layer. This layer is seen as containing various protocol, aimed at supporting different type of applications. The best well-known examples are ISO standard IS 8571 File Transfer, Access and Management (FTAM), and IS 8832 Job Transfer and Manipulation Protocol (basic).

1.3 Structure of the Thesis

The first step in this project is to understand the architecture of the two networks to be interconnected, XLNET and TMS-IBM Ring. Chapters 2 and 3 describe some of the important aspects of the two rings, respectively. After the literature overview, Chapter 4 describes the proposed gateway architecture. A description of the "protocol translator" is also given in this chapter. Chapter 5 describes the laboratory implementation of the internetworking system. In this chapter, the hardware aspects of the gateway and the relevant software will presented.

The performance of the gateway implemented in the internetworking system is analyzed in Chapter 6. In Chapter 7, the shortcomings and possible improvements of this project are discussed. Finally, Chapter 8 summarizes the conclusions. The basic structure of this thesis is outlined in Fig.1.4.

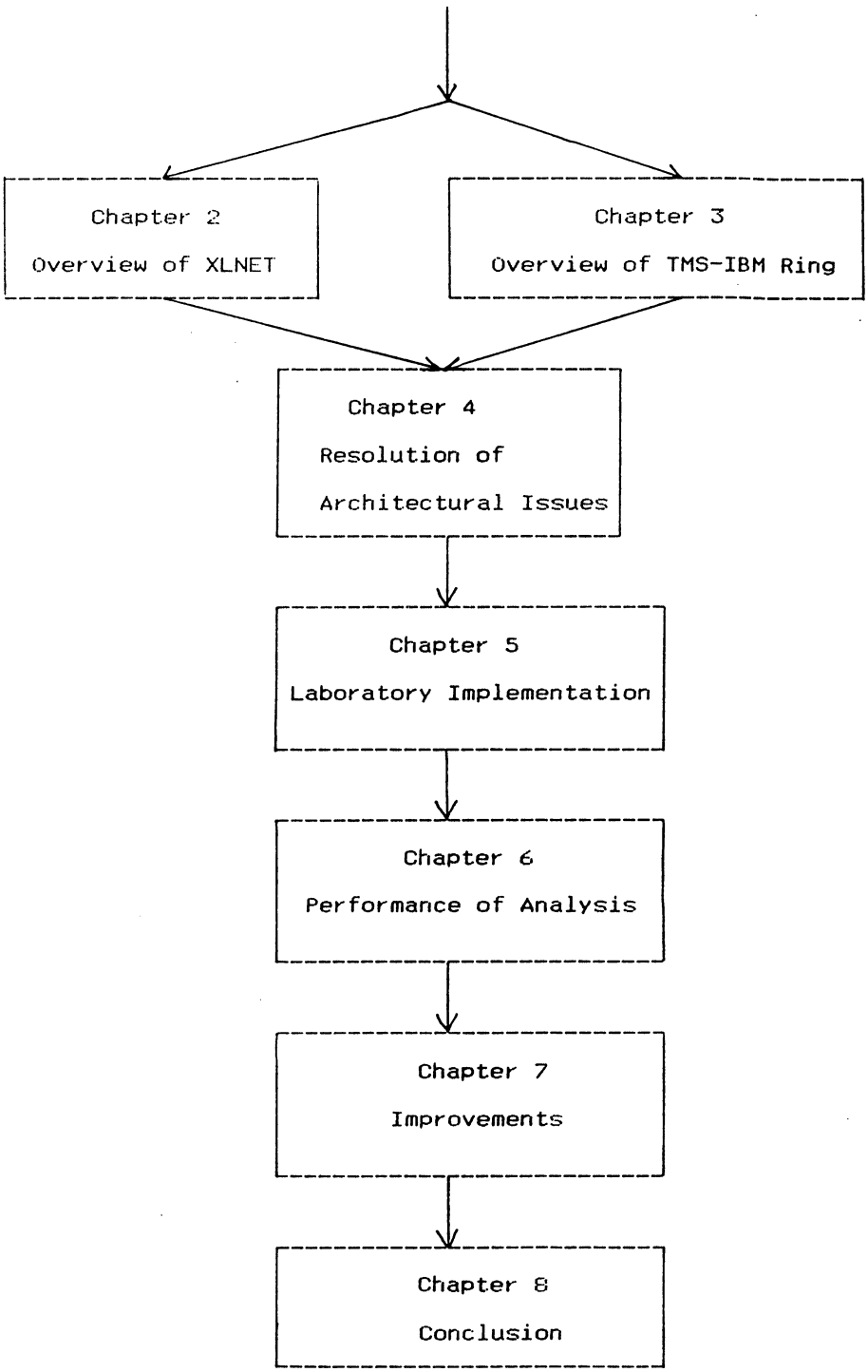


Fig. 1.4 The Structure of this Thesis.

CHAPTER TWO

OVERVIEW OF XLNET

2.1 Introduction

XLNET is a novel local area network with fully distributed architecture, which was originally developed by A.E. Karbowiak and G.J. Anido in the University of New South Wales. XLNET is a LAN based on the Open System Interconnection (OSI) Reference Model to integrate the transmission of voice and data communications. Since, the integration occurs at all levels of the architecture and packet switching is used as the vehicle for integration, XLNET allows the system capacity to be dynamically shared among the users.

This chapter is intended to describe some important aspects of XLNET. In section 2.2, the hardware aspects of XLNET will be presented. Then the XLNET software structure will be briefly introduced in section 2.3. Finally, the main characteristics of the system will be reviewed in section 2.4.

2.2 Hardware Aspects

The integrated switching facility of XLNET is shown in Fig. 2.1. The switching entities are called nodes. Each node contains three units. The name and functionalities of each unit are below:

1. Link Interface Unit (LIU): LIU provides interfacing between the node and XLNET ring, enabling logical and physical access to the network. The architecture of the LIU is shown in Fig. 2.2. Details of operating these functions are given in [18].

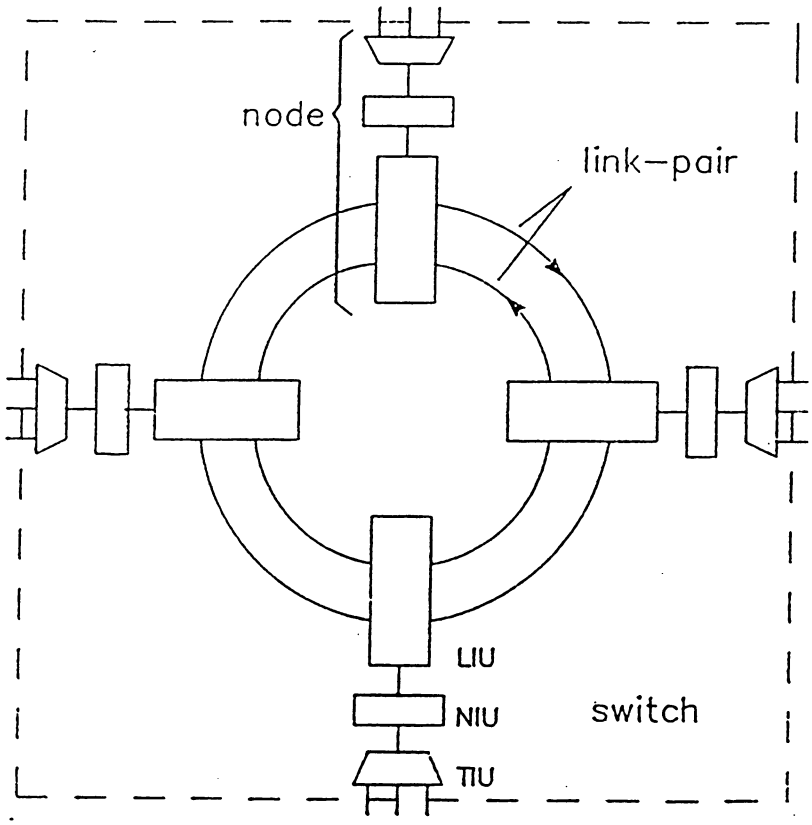


Fig. 2.1 Switch Architecture

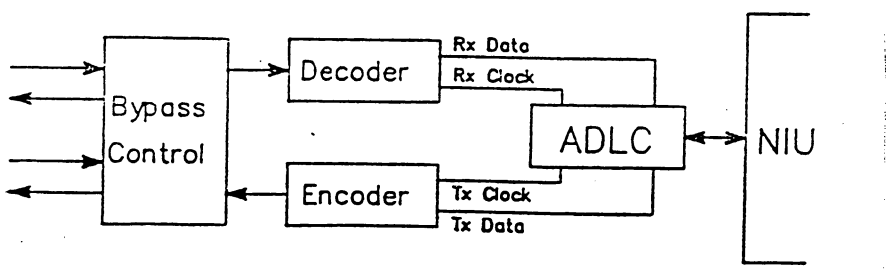


Fig. 2.2 LIU Architecture

2. Network Interface Unit (NIU): NIU provides the protocol handling capabilities and data processing of the XLNET node. The architecture of NIU is shown in Fig. 2.3. Details of operating these functions are given in [18].

3. Terminal Interface Unit (TIU): TIU provides the physical ports through which the user communicates with the node and with other users on the XLNET. The architecture of TIU is shown in Fig. 2.4. Details of operating these functions are given in [18].

The system prototype specifications are listed in table 2.1 [2].

2.3 Software Aspects

The architecture of network software protocol is based directly on the OSI seven-layered Reference Model. The architecture has the structure shown in Fig. 2.5. The Data Link Layer (layer 2) is further divided into Logical Link Control and Medium Access Control Sub-layers as recommended in the IEEE 802 documents. The Medium Access Control Sub-layer uses many of the recommendations of 802.4 [3, 4], while the 802.2 Class 1 protocol (connection-less operation) [5, 6] is adopted for the Logical Link Control Sub-layer. Further, the OSI Connection-less Network protocol [7, 8] is selected for the Network Layer. The resulting network service offers an efficient and reliable datagram service. For those applications requiring a connection-oriented service, the appropriate choice of Transport Layer software is the OSI Connection-oriented Transport Protocol Class 4 [9, 10] which supports the necessary sequence and flow control.

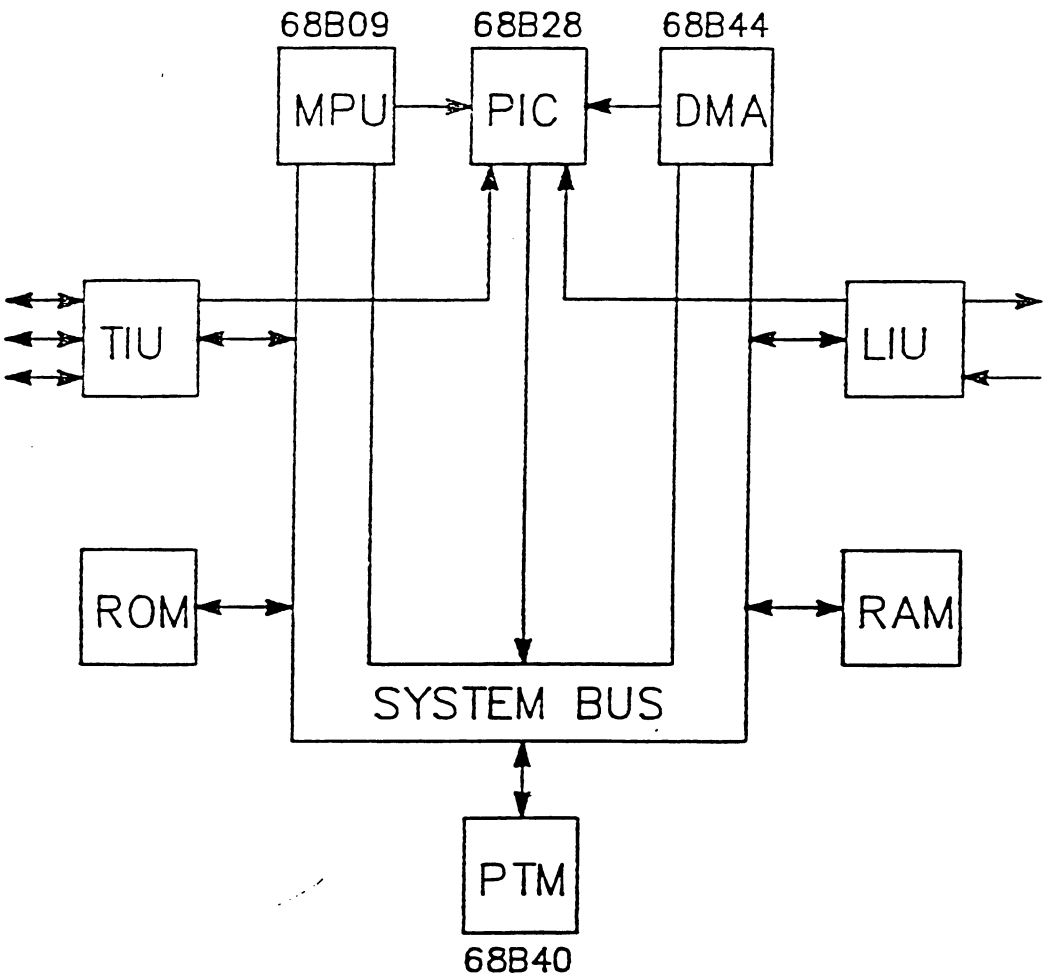


Fig. 2.3 NIU Architecture

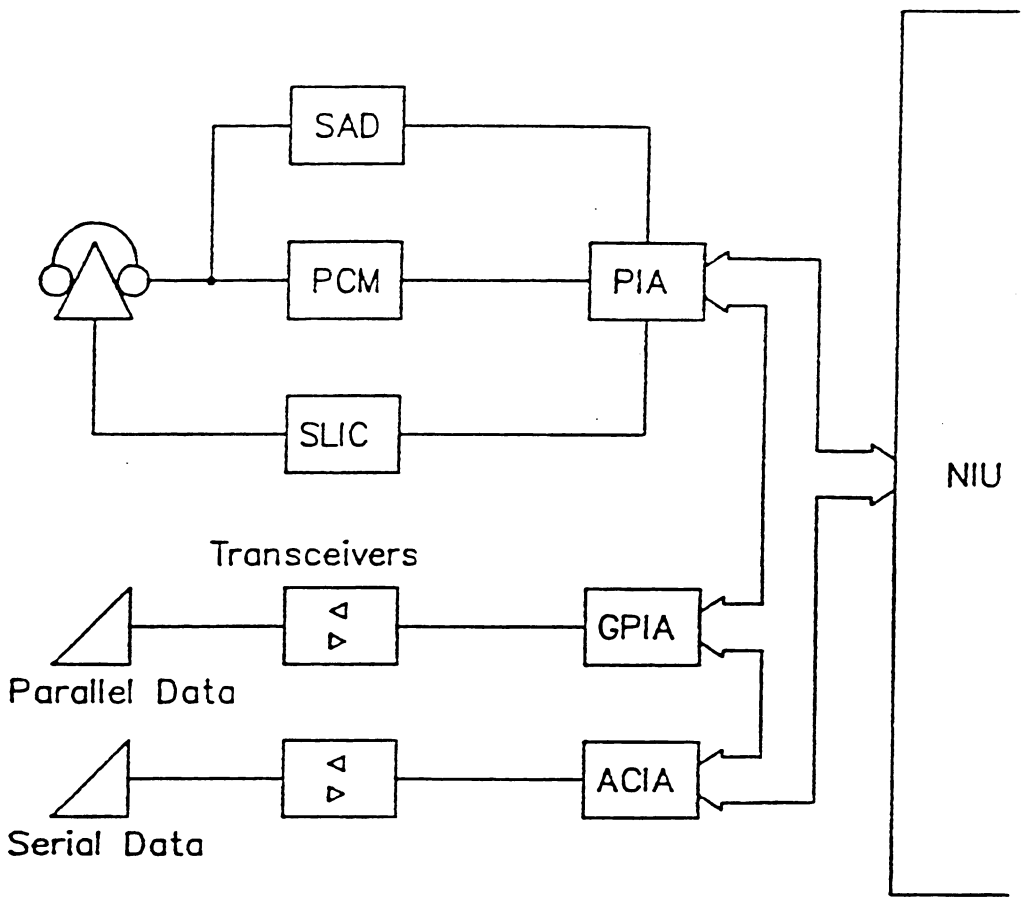


Fig. 2.4 TIU Architecture

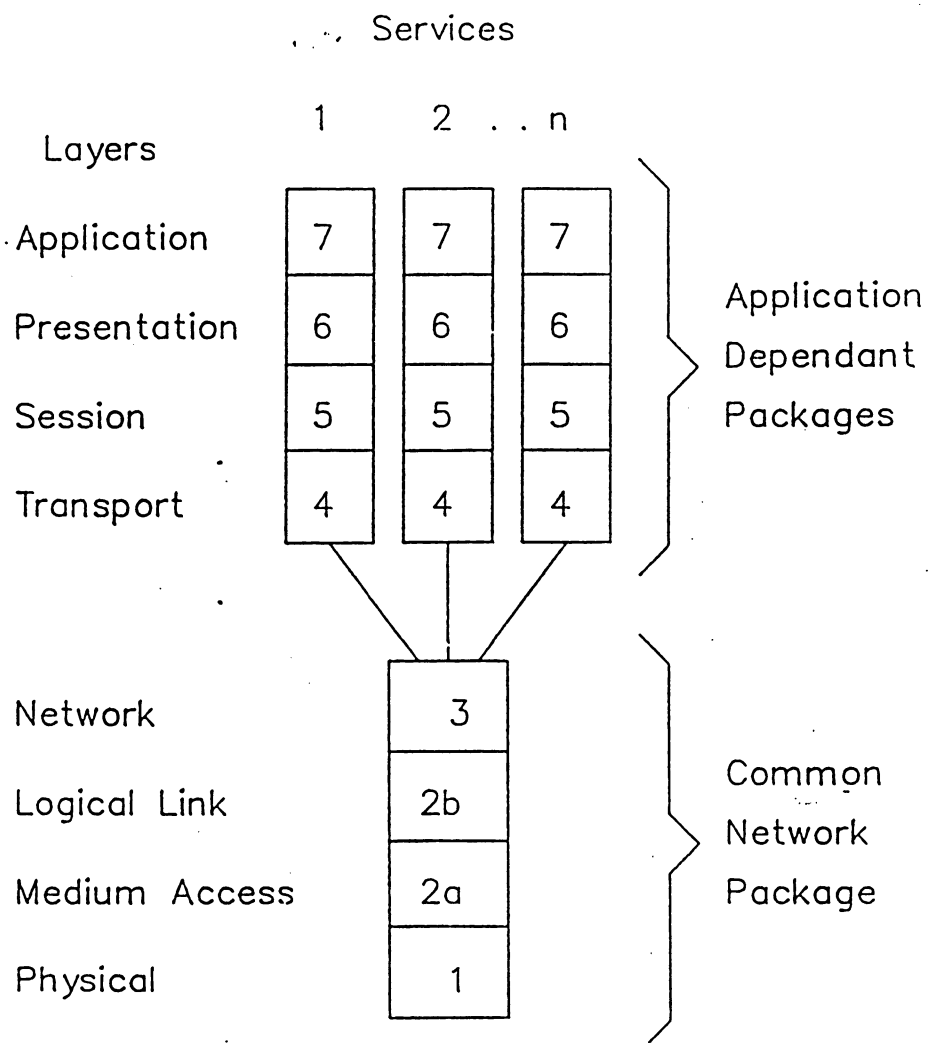


Fig. 2.5 Protocol Architecture

Medium	Twisted Pair
Line Code	Biphase-S
Signalling Rate	2 M Baud
Access Address space	255 nodes
Data Aspects:	
Bandwidth	1 M bit/second
Delay	3.5 ms (1 K bit packets)
Interfaces	IEEE-488; v.24
Voice Aspects:	
Capacity	12 simultaneous calls
Loss	< 1%
Delay	8 ms (to PSTN)
Attack Time	< 0.4 ms
Hangover	150 - 250 ms

Table 2.1: Prototype specifications

In essence, the software used in XLNET is composed of a number of routines called processes. These processes are divided into two distinct groups: low-level processes, and higher-level processes.

The low-level processes provide the functions that require critical responses. The functions performed by these processes are associated with the user ports of the TIU such as the handling of voice communications and the functionalities of LIU such as frame reception and transmission. In general, the low-level processes are invoked by the use of hardware and software interrupts.

The higher-level processes perform functions associated with node and network management, and with user-oriented communications. These processes are driven by the operating system XLNCE.

The correspondence between the hardware units, software processes, and protocol layers is shown by the process flow chart in Fig. 2.6. The executive, called XLNCE, provides inter-process communication capabilities for use between the NIU processes by using massage-passing, and between NIU and low-level processes by using mail-boxes. Details of the executive are given in [19].

2.4 Other Significant Features

As previously mentioned, XLNET, is able to integrate the transmission and switching of synchronous and non-synchronous communications (for instance, voice and data). Several advanced schemes, such as efficient integration of data and voice communications within a limited geographical local area, have been employed to provide improved communication capabilities.

The key characteristics of the system are as below:

(1) Using the distributed cyclic service protocol to transmit voice packet with high efficiency and acceptable delay;

(2) Using speech interpolation to increase voice carrying capacity.

(3) Using variable time-slot to improve data carrying efficiency.

(4) Using a decentralised architecture to enhance reliability.

In the following sub-sections, these functions will be examined in detail and some special features of XLNET will be discussed as well.

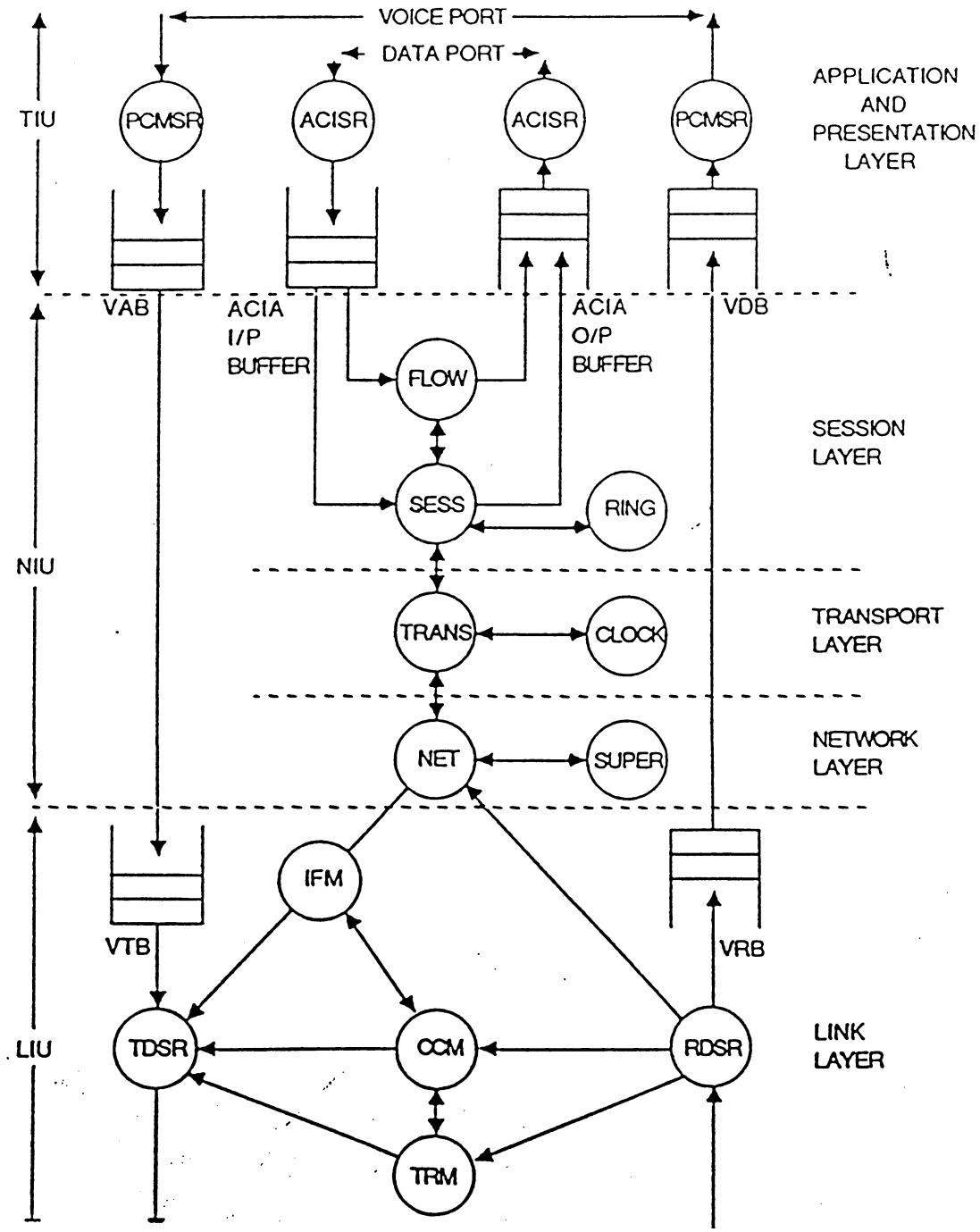


Fig. 2.6 Process Flow Graph showing the main Interrup and XLNCE processes and the main interprocess communiction paths

2.4.1 Strategy of Token Generation

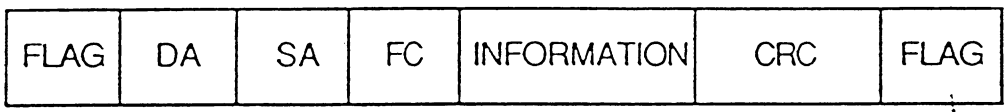
The medium access mechanism of XLNET is based on the token ring passing protocol IEEE 802.4 [3]. In a ring topology, network nodes are linked by a form of closed ring. Message or information, in the form of frames or packets (the frame format of XLNET shown Fig. 2.7), normally flows unidirectionally around the ring. In the token passing mechanism, a token is a unique symbol representing the right to access the network. The token format of XLNET is shown in Fig. 2.8. Initially, a free token generated by a designated node (monitor) travels around the ring until a node ready to transmit converts it into the opening flag of the frame. The node proceeds to transmit the remainder of the frame, following which the token is released.

The strategy employed for token generation plays an important role in the token ring passing protocol. There are three possible strategies:

(1) Single packet mode: the transmitting node has to wait until the frame completely circulates the ring before releasing the token into the ring.

(2) Single token mode: its operation is different from the first in that, the transmitting node need only wait until the head of the frame rotates around the ring.

(3) Multiple token mode: the token is released at the end of the frame transmission.



FLAG = 01111110 (1 OCTET)

DA = DESTINATION ADDRESS (1 OCTET)

SA = SOURCE ADDRESS (1 OCTET)

FC = FRAME CONTROL (1 OCTET)

INFORMATION = INFORMATION FIELD (n OCTETS)

CRC = CYCLIC REDUNDANCY CODED CHECK SEQUENCE (2 OCTETS)

Fig. 2.7 Frame format

TOKEN = 01111111 (1 OCTET)

(b) Token Format

Fig. 2.8 XLNET token

In the consideration of frame service time, a choice of suitable strategy employed in the token ring depends on the nature of the ring. On small rings with a roundtrip delay smaller than the frame transmission time, under the operation of single token mode, the service time is simply equal to the transmission time; otherwise, it is equal to the roundtrip delay. In contrast, on large rings with a roundtrip delay larger than the frame transmission time, the multiple token mode is a better choice than other modes since the service time always equals the transmission time.

In our interconnected network system, the multiple token mode is employed in XLNET, while the single token mode is used in TMS-IBM Ring (see Chapter 3).

2.4.2 Distributed Cycle Service Scheme

In this sub-section, the distributed nature of the cyclic service scheme will be considered.

In common with other members of the cyclic service networks such as the Zurich Ring [11, 12] and Welnet [13], XLNET operates by dividing the time scale into fixed length intervals called cycles shown in Fig. 2.9. The first part of each cycle, called voice service sub-cycle, is given to voice traffic, while the remainder of the cycle, called data service sub-cycle, is offered to data traffic. The cycle duration is chosen to equal the voice packet generation time, 16 milliseconds.

XLNET employs a control packet, called the poll packet (whose format shown in Fig. 2.10), corresponding to a timestamped packet [14] as the basis for a voice synchronisation mechanism. The poll packet is generated at the start of a given cycle by the controller of that cycle. The controller, called the polling node, maintains a cycle timer.

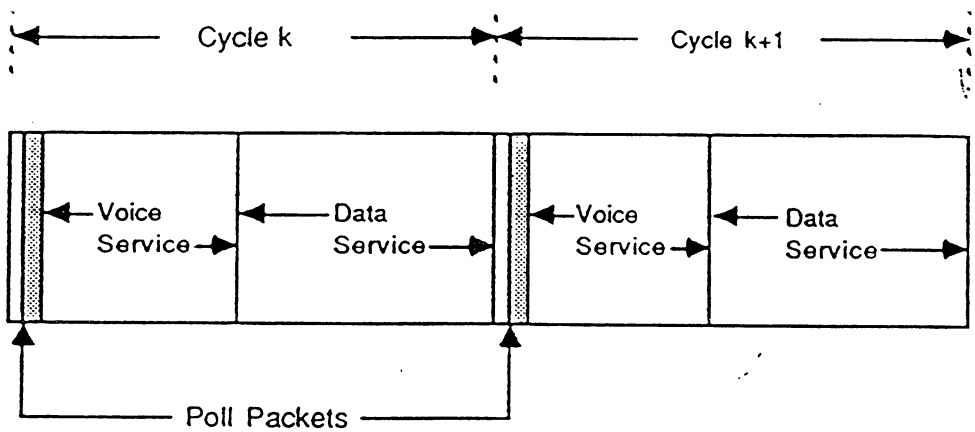


Fig. 2.9 Cycle Structure showing voice service, which starts following the transmission of the Poll packet, followed by data service.

HEAD	DA	SA	CONTROL	TIMESTAMP	TAIL

Fig. 2.10 Poll packet format showing the Timestamp field which, upon reception, contains the time since the current cycle started.

(DA: Destination Address, SA: Source Address)

As a result of the poll packets, the cycle timers of all nodes are synchronised. When the end of each cycle is reached, the cycle timer in each node will timeout, and the first node to receive the token following the end of each cycle creates and issues a poll packet, the first token arrival at each node following the poll packet is used for voice service, with subsequent arrivals for data service.

If the number of voice packets requiring service during any given cycle exceeds the capacity of the system, then the token will not make it back to the polling node prior to the end of the cycle. Rather than the fixed location of the cyclic service controller, the location of that in XLNET is distributed. In other words, it is not bound to any particular location on the ring, but in response to the applied load. A number of advantages are achieved by using such a distributed controller:

(1) The performance penalties [14] by the use of a centralised control of service cycle are eliminated. Since the location of the controller will vary from cycle to cycle, no node will be favourably served during any service cycle. In other words, the desire of the fair access to the ring medium is reached.

(2) The voice synchronisation technique can be applied. Since the cyclic service controller periodically issues a timestamped packet which is always generated at the start of a cycle, it is able to meet the requirement to reach the voice synchronisation. Sub-section 2.4.3 will discuss it in greater detail.

(3) Reliability is promoted since control of the cyclic service and voice synchronisation is not based on any particular node.

The performance, associated with probability of packet discard and loss of voice capacity by means of the distributed cycle service scheme, was analyzed by Gary Anido [14]. The comparison of voice service utilisation for XLNET, Zurich Ring and Welnet is shown in Fig. 2.11. The latter two networks employ centralised cycle service controllers.

2.4.3 Efficient Improvement On Voice Service

In general, there are two problems encountered in networks which employ packet switching for voice. First, the excessive voice delay causes echo problems on PSTN calls. Secondly, the delay variance results in glitching in the receive voice signal.

XLNET has successfully solved this problem by the use of a "Cyclic Service Switching Scheme", discussed in section 2.4.2, for the integration of voice and data. The scheme allows the echo delay on calls to the PSTN to be arbitrarily small. Furthermore, the scheme can be applied for an elegant variance smoothing technique to overcome the glitching in the receiving signal.

How can the cycle service scheme be a powerful weapon for shooting the two troubles above? The explanations will be presented in the following paragraphs.

In XLNET, all the nodes in the ring are synchronised at the beginning of a cycle by a polling frame, the voice packets assembled in the previous cycle are serviced in the current cycle. When the voice packets arrive at the destination node, they will not be played out until the beginning of the next cycle. Hence a voice delay is the constant end-to-end delay which is equal to two cycle durations.

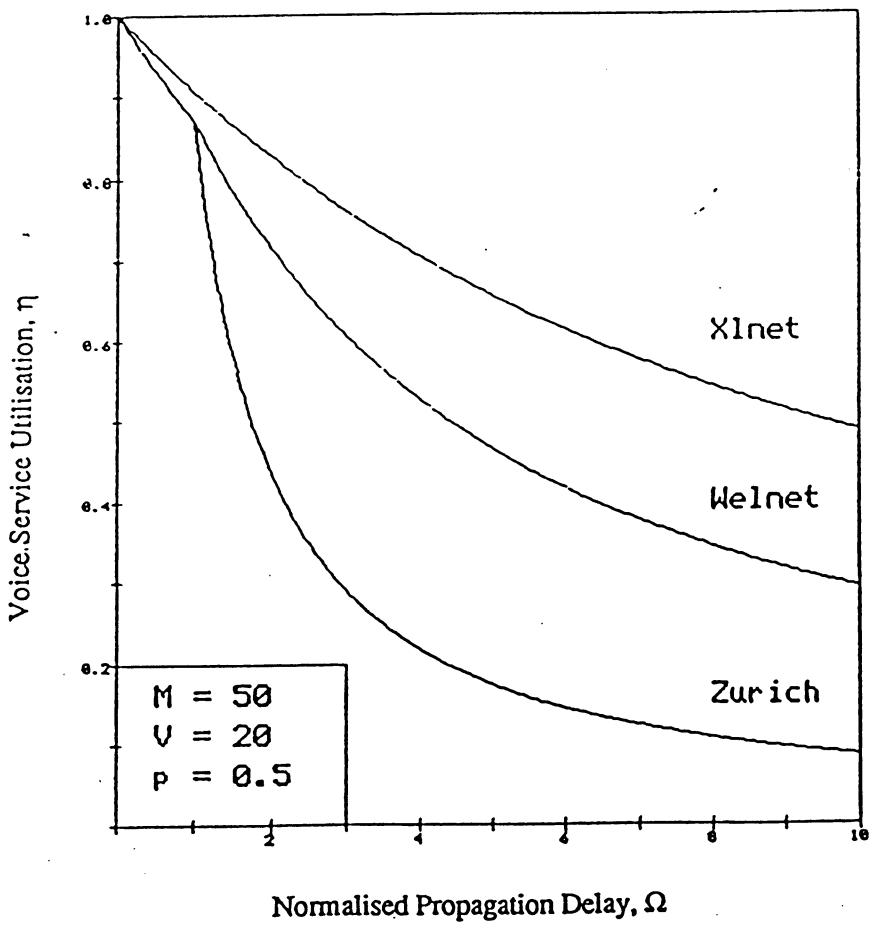


Fig. 2.11 Comparison of voice service utilisation for the three systems (Zurich ring, Welnet, and XLNET).

The end-to-end delay for a voice packet is defined as the sum of the packetisation delay and the transfer time, that is

$$D = T + Tf$$

where D is the end-to-end delay, T is the cycle duration (equal to the packetisation delay), and Tf is the transfer time which is defined as the time from when a packet is assembled to the time it is disassembled at the destination node.

If the transfer time is allowed to vary, the end-to-end delay also varies. It results in discontinuous voice output forward to the receiver, called "glitching" [15]. The implementation of Minoli's "Receive End Buffering" with "Limited Waiting for Late Packet" [16] employed one method to overcome this problem. This method succeeds by inserting a delay 'Dy' to smooth the delay variance. This introduced delay is employed to augment the transmission delay "Dt' such that the transfer delay is a constant. the formula is below:

$$Tf = Dy + Dt = T \text{ [seconds]}$$

The introduced delay Dy is bound by T (equal to one cycle time).

In XLNET, by means of the evolutionary cycle service scheme, the end-to-end delay is a constant equal to twice the cycle duration. The voice signal reproduced by the destination node is essentially continuous. Therefore, the problem of "glitching" is solved.

2.4.4 Speech Interpolation and Its Advantage

Since XLNET uses the fair access and distributed cyclic service scheme, there are no pre-assigned locations within the voice service for specific voice sources. Under this circumstance, the use of speech interpolation is readily accommodated. If a voice user is currently silent,

then no packet is transmitted during the voice cycle. In this case a token is passed without serving a packet when it arrives at the node during the voice service, the duration of the voice service is reduced and effective data capacity is increased. When the user returns to talkspurt and is generating voice packets, the increased voice service is accommodated by borrowing back the capacity previously made available to data. The flexibility is contributed by the "Dynamic Boundary" shown in Fig. 2.12 [16] in the cycle service scheme.

By means of this technique, under heavy voice traffic, much of the nominal capacity will be used for most of the time. Furthermore, under light voice load, data sources are able to use nearly the entire system capacity. In a word, the speech interpolation, together with the dynamic boundary is a key scheme used to improve the utilisation of the system capacity.

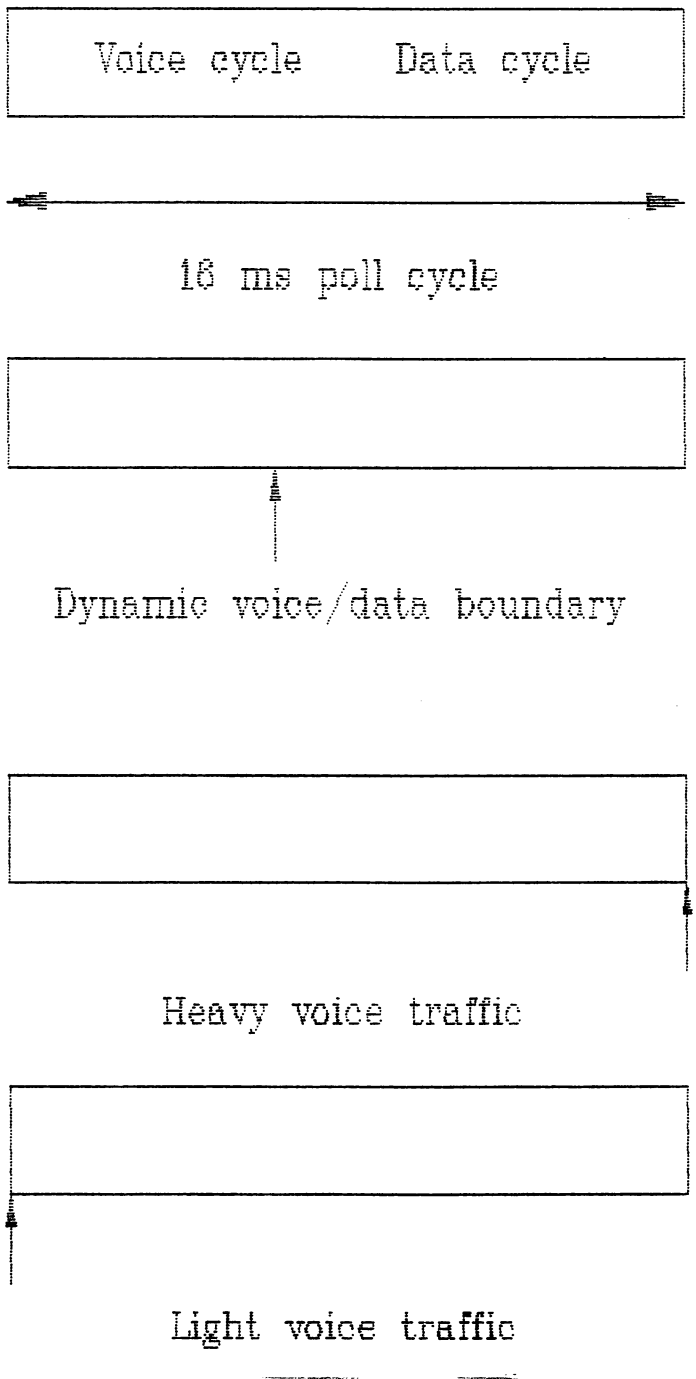


Fig. 2.12 Dynamic voice/data boundary in XLNET

CHAPTER THREE

OVERVIEW OF TMS-IBM TOKEN RING

3.1 Introduction

The TMS-IBM Token Ring, based on the token ring architecture with a star wired configuration, is designed to carry computer traffic. It uses the TMS-380 chipset [20]. The integration of voice and data in the IBM ring was successfully achieved by T.L. Ng at the University of New South Wales in 1987 [21]. The operating system of TMS-IBM Ring, NET-OS is not a feature of the TMS commercial product, but is designed by T.L. Ng to be used into the network system with the TMS-380 adapter chipset and IBM XT/PC machine. The gateway we have realised bridges this implementation of the IBM Ring on one side and XLNET (see Chapter 2) on the other.

In section 3.2, the key architecture of the hardware, TMS-380 adapter chipset will be discussed. The software, NET-OS, and the operation of the TMS-IBM Token Ring will be introduced in section 3.3. Finally, in the last section of this chapter, section 3.4, we present an overview of the main characteristics of the system.

3.2 Hardware Aspects--TMS-380 Adapter Chipset

The implementations of the Physical Layer and the Medium Access Control Sub-layer of the TMS-IBM token ring network are realised by the TMS-380 adapter chipset, which was developed by the Texas Instruments and IBM, in the spring of 1985. The main feature of the adapter architecture is shown in Figure 3.1 [20]. The adapter essentially consists of five VLSI chips which together enable a transmission of 4 megabit per second signal through the twisted pair wire.

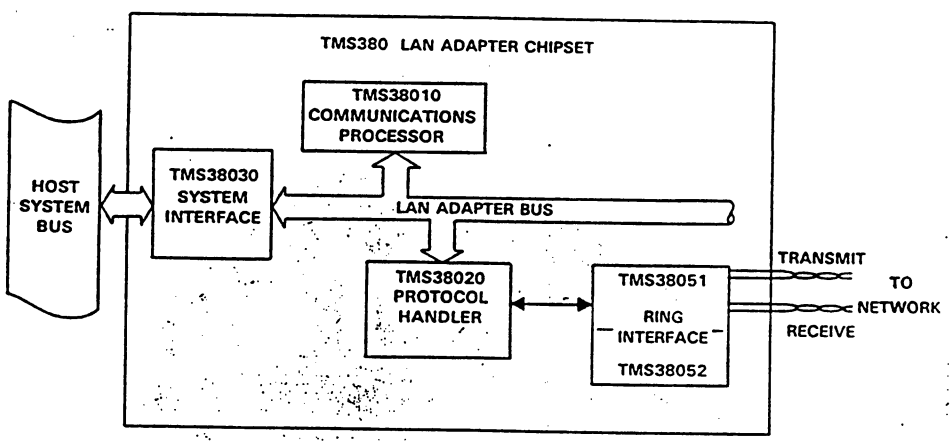


Fig. 3.1 TMS380 Integrated Adapter Architecture

Furthermore, as the adapter works as a front-end processor to execute the host-independent operation in the LAN, it does not only reduce the burden of the host system in the LAN, but also provides a reasonable degree of reliability for data transmission.

The function of the main components of the adapter chipset are summarized in the following sub-sections.

3.2.1 TMS38010 Communications Processor (CP)

The Communication Processor contains a dedicated 16 bit CPU with 2.75k bytes of on-chip RAM. The CP executes the adapter software resident within the TMS38020 (Protocol Handler). Besides, it maintains on-chip RAM buffers the frame being received and transmitted. The chip provides single cycle arbitration of the 3 MHz LAN adapter bus for maximum data throughput in the adapter chipset. A more detailed functional block diagram is given in the Appendix A.1.

3.2.2 TMS38020 Protocol Handler (PH)

The Protocol Handler performs hardware based protocol functions for a 4-megabit per second token ring LAN compatible with the IEEE 802.5 standard. The adapter software, contained within the 16k byte on-chip ROM and executed by the Communications Processor, supports LAN management services, diagnostic coverage and ring operation. The PH implements frame address recognition and Differential Manchester encoding/decoding. To insure high-speed frame transfer between the ring and the adapter's buffer RAM, the PH provides 4 DMA channels, two for receive and two for transmit. The detailed functional block diagram is collected in Appendix A.2.

3.2.3 TMS38030 System Interface (SIF)

The System Interface (SIF) offers up to 40 megabits per second of data to the host system via its own built-in DMA transfers. as its name implies, the SIF acts as the interface between the host system and the adapter chipset card through the System Command Block, the System Status Block and the interrup registers. A more detailed functional block diagram is given in Appendix A.3.

3.2.4 TMS38051 Ring Interface Transceiver (RIT) and TMS38052 Ring Interface Controller (RIC)

The Ring Interface Transceiver and Controller provide the functions for the ring interface, such as the clock for the ring when in active monitor mode, a phase locked loop for clock recovery, phase alignment, data detection, error detection of wire faults, a loop-back path for diagnostic testing and so on. The two chips, in conjunction with the above three chips, form a highly integrated token ring LAN adapter compatible with IEEE standard 802.5-1985 Token Access Method and Physical Layer Specifications. A more detailed functional block diagram is given in Appendix A.4.

3.2.5 The Operation of the TMS-380 Adapter

The adapter presents a full duplex interface to the IBM XT-PC host system with separate receive and transmit channel between the adapter and the LAN, and a 48-megabit per second LAN adapter bus with single-cycle arbitration for internal adapter transfer. Furthermore, a 40-megabit per second DMA controller connects the LAN adapter bus to the host

system bus. Figure 3.2 [20] shows data flow through the TMS-380 adapter chipset.

In particular, the PC host system and the adapter pass information through a shared memory which contains the System Command Block (SCB) and System Status Block (SSB), to each other. The adapter can be initialized to meet specific host system bus requirements. The initialization parameters include:

(1) Interrupt routines.

(2) Buffer sizes.

(3) Allocation of these buffers to transmit and receive channels.

(4) Expansion memory.

(5) The setting of addresses.

As mentioned earlier, the communication between the adapter and the host system is via the SCB and SSB. A procedure of a typical transmit operation is shown in Figure 3.3 [20]. Before transmitting a frame, the PC host first sets up the SCB which contains the transmit command and the starting address of the transmit list in the host system memory. Then, the host interrupts the adapter. After receiving an interruption from the host, the adapter DMA reads the SCB, and according to the address of the transmitted frame in the SCB, the adapter DMA reads the whole frame to the adapter RAM. Having the validation on the transmit list and frame format, the adapter captures a free token and transmits the frame onto the LAN. The frame will circulate until it returns to the sender adapter, where it is removed from the ring and a free token is released to the LAN. The SSB is then updated by the adapter and sent to the host system via the adapter DMA.

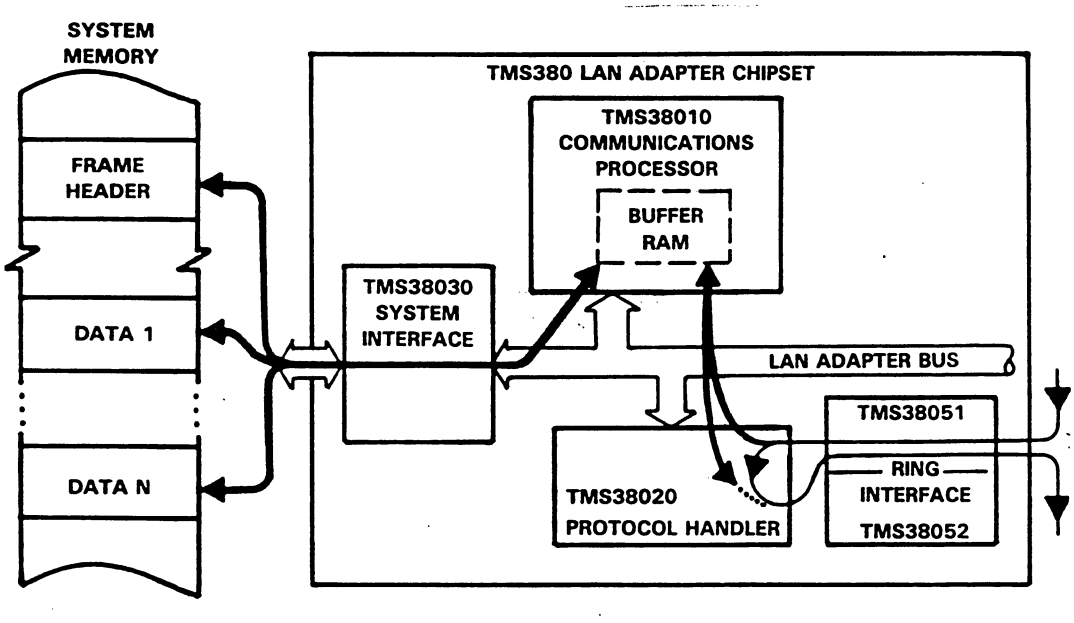


Fig. 3.2 Adapter Data Flow

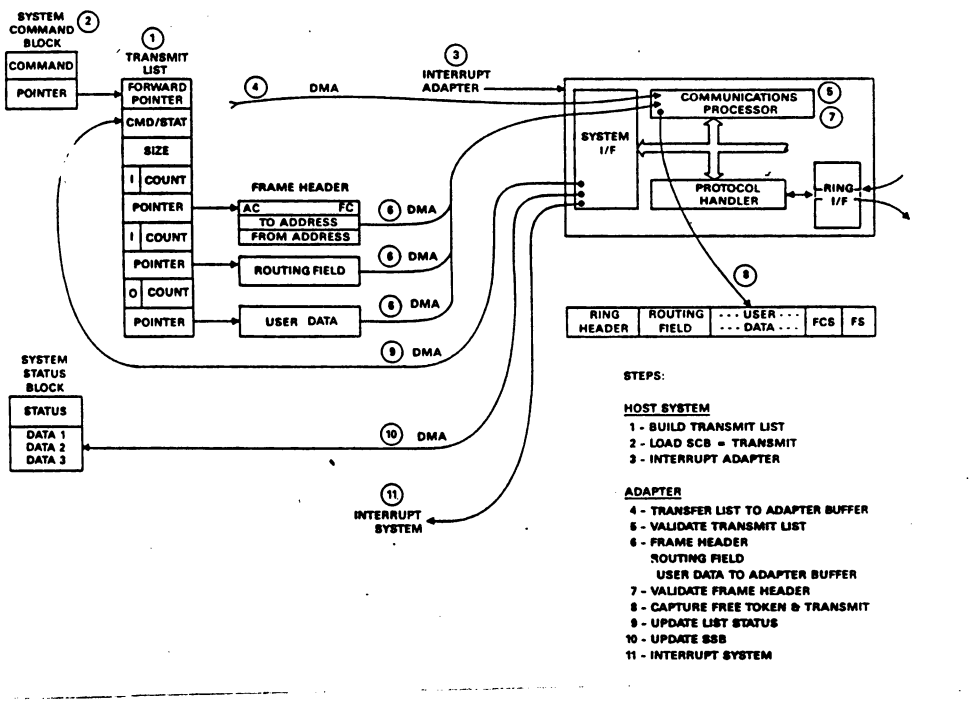


Fig. 3.3 Transmitting A Frame

In a similar fashion, the host issues a Receive Command via SCB to the adapter pointing to the receive list in host system memory. Upon reception of a frame, the adapter will transfer the receive list from the host system to the adapter RAM via DMA, then transfer the received data to the appropriate location in the host system memory via DMA as well, and finally update the SSB.

3.3 Description of Software

Taking the advantages of the modularity of design of open systems, the network software architecture, as in XLNET, is based on the OSI seven-layered Reference Model. The architecture has the structure shown in Fig. 3.4. The Physical Layer and Medium Access Control Sub-Layer (IEEE 802.5 Token Passing Ring [22]) are provided by the hardware and firmware in the TMS-380 adapter chipset, which has just been discussed in the last section. The Logical Link Control Layer, Network Layer and Transport Layer employ the protocols similar to those of XLNET, the IEEE 802.2 Class 1 Protocol, the OSI Connection-less Network Protocol and OSI connection- oriented Transport Protocol Class 4, respectively. The protocol of the Session Layer is designed by T.L. Ng [21].

Under this protocol structure, the system does not only offer datagram service to implement the integrated communications (data and voice), but also provides functions such as flow control, error recovery, and the ability to mutiplex multiple transport connections on to one network connection.

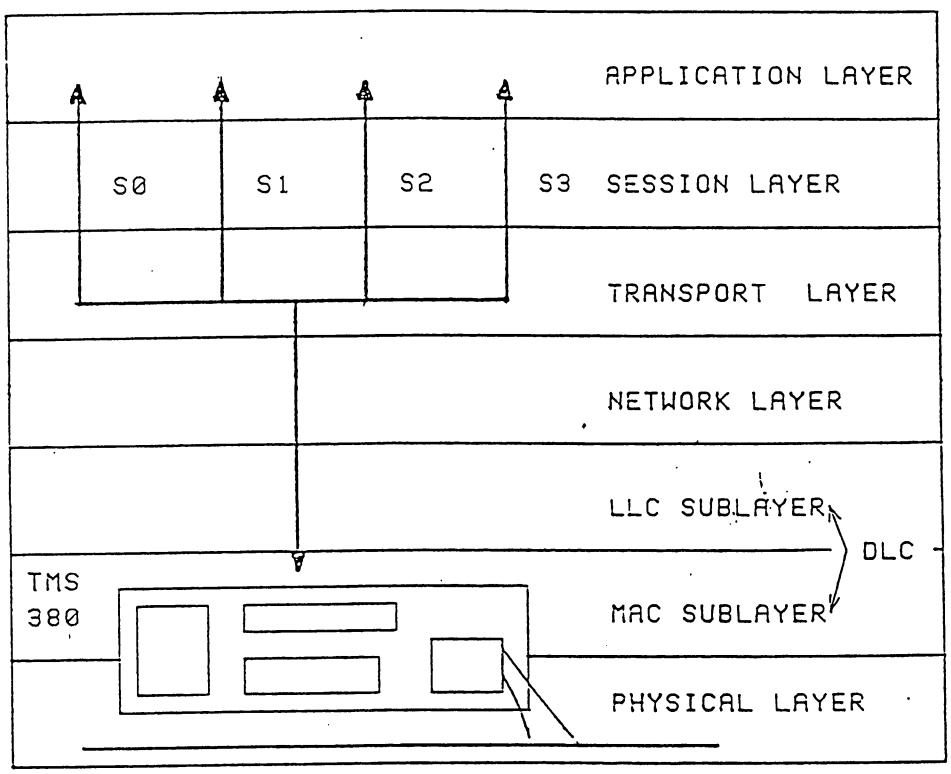


Fig. 3.4 Layered Architecture

As with XLNET, the software used in TMS-IBM Token Ring is composed of several routines, called processes. The processes are divided into two distinct groups: synchronous processes and non-synchronous processes. Indeed, the functions provided by the two kinds of processes are similar to those offered by the low-level processes and the higher-level processes in XLNET as well. The synchronous processes are driven by the operating system NET-OS [21] which takes advantages of the multitasking environment [23] to process tasks concurrently. At the heart of the NET-OS is a table of process Command Blocks. Each process command block contains information pertaining to the status of each process (shown in Figure 3.5). Non-synchronous processes, on the other hand, are interrupt driven.

3.4 Special Features

The contribution of NG Lee is to implement an efficient integration of voice and data communications in the TMS-IBM Token ring. Although some of the protocols employed here, as mentioned earlier, are quite similar to those in XLNET, there are several techniques significantly different from those in XLNET. In the following sub-sections, those dissimilar but important schemes will be examined.

3.4.1 The Integration service

The switching scheme for the integrated communications, in TMS-IBM Ring, is based on the nature of multiple level priority packets, in lieu of the distributed cycle service technique in XLNET.

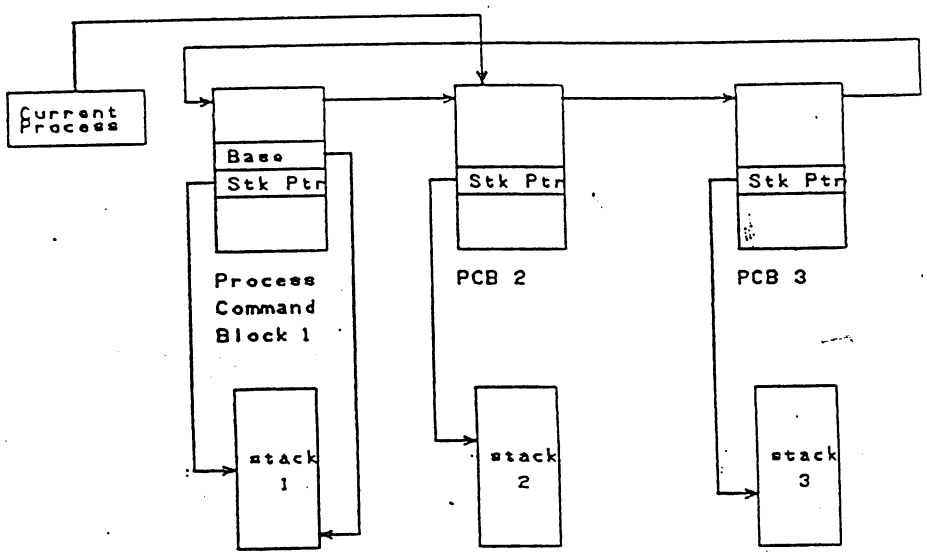


Fig. 3.5 NET-OS Environment

According to Bux [24], the performance of the integration achieved at the Medium Access Control Sub-layer such as the case of XLNET should be better than that at any other level of the OSI reference model. However, the protocol of the Medium Access Control Sub-layer provided by the firmware of TMS-380 adapter chipset is invisible and inaccessible for commercial reasons. Therefore, the integration is impossible to be implemented in this level, although it can be done at a higher level, namely the Logical Link Control (LLC) Sub-layer. In TMS-IBM Ring, the LLC sub-layer assigns voice packets a higher prority over data packets, and maintains two queues for these two kinds of packets. Whenever a free token arrives at a node requiring service, the voice queue is served first, following that, the data queue is served until the token-holding timer times out [21].

3.4.2 The Scheme of Speech Interpolation

The scheme used to overcome the problem generally associated with voice communications in packet switched network, such as the delay of voice, is Minoli's "receive end buffering" with "limited waiting for late packets". The comparative principle and operation of the scheme for smoothing the voice packets' delay between XLNET and TMS-IBM Token Ring have been presented in section 2.4.3.

3.4.3 Token Access Techology

As mentioned previously, the TMS-IBM Token Ring employs a token passing access method with a star-wired ring topology as described by IEEE 802.5. Figure 3.6 [20] shows the topology of token passing access with 4 nodes A,B,C,D connected in the ring. When there is no data being transmitted, a free token will circulate unidirectionally around the ring.

The token format is show in Figure 3.7 [20] which consists of three bytes: STARTING DELIMITER, ACCESS CONTROL, ENDING DELIMITER. If any of the nodes, such as node A, desires to transmit a frame through the ring to a destination such as node C, the steps it must follow are listed below:

(1) Wait for the free token.

(2) Seize it on its arrival.

(3) Mark it 'busy' by setting a specific bit in the access control field of the token format.

(4) Convert it into a data frame with the format shown in Figure 3.8. [20].

Upon recognising the destination address of the transmitted frame the receiving node C copies the data frame as it passes through the interface. Node A that originated the frame has the responsibility for removing the frame when it circulates back, and releases a free token onto the ring for the other nodes to use. It is worthy to mention that the difference between a token and a data frame is not only as regards their lengths but also the status of the specific bit in the access control field. If the specific bit is set, the token is occupied and becomes part of the header of a data frame; otherwise the token is free. Token passing access provides priority levels to be assigned to tokens, thus providing support for synchronous traffic. As there is always one token on the ring and possession of the token gives that node exclusive use of the ring, for the purpose of fair token access to the ring for all the nodes, a token-holding timer restricts the maximum time a node can use the token before passing the token.

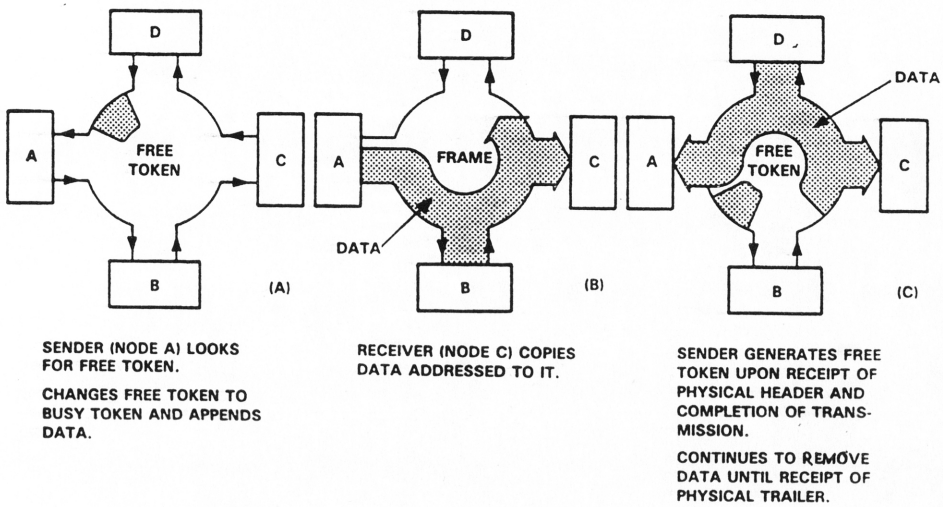


Fig. 3.6 Token Passing Example

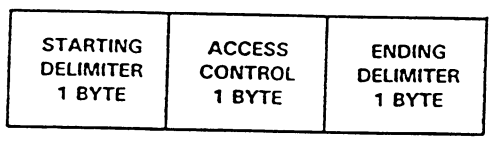


Fig. 3.7 Free Token Format

Fig. 3.8 Frame Format

CHAPTER FOUR

RESOLUTION OF ARCHITECTURAL ISSUES

4.1 Introduction

An appropriate resolution of internetwork architectural issues plays a very important role in the internetwork design. The key to the resolution is dependent on the applications of the internetwork system and the technologies used in its constituent networks. This chapter discusses the solutions of the key architectural issues in the internet work system considered here.

An interconnected set of networks is referred to as an internet, and a constituent network in an internet is referred to as a subnet. In addition, subnets are connected by devices which are referred to as gateways. A gateway provides a communication path to exchange messages between subnets. Under this internetworking architecture, the first issue which is discussed in section 4.2 is the strategy taken to achieve the subnet independence. Secondly, the four possible approaches, "bridge, X.75, Internet protocol and Protocol translator", applied to internetworking will be briefly described in section 4.3. The last one, Protocol translator, was chosen as the approach for the internet . The reasons for this choice are given as well. Section 4.4 discusses the connection-oriented and connectionless communication services. The next issue to be discussed in section 4.5 is the addressing methodology. Then, the shortcomings of the implementation on the segmentation and reassembly in the considered internet will be given in section 4.6. Finally, section 4.7 explains why routing is not necessary in the particular implementation.

4.2 Subnet Independence

There is a large investment in the hardware and software of the two existing networks, XLNET and TMS-IBM Token Ring. As far as possible, the internet protocols should be transparent to the two existing subnets. Obviously, some changes to the software of the two systems need to be made, but we shall follow the principle: the fewer changes, the better.

Two possible strategies are considered:

(1) Internet messages are embedded in the subnet messages as user information (see Fig.4.1.), just as Network layer messages are embedded in the Data Link layer messages. The internet header contains the destination subnet and station addresses. The subnet header contains the address of the next gateway. The gateway removes the first subnet header and trailer and adds new ones before transmitting the message to the next gateway and so on. The internet message is thus completely transparent to the local network, but this scheme has the shortcoming of increased overhead of the local as well as internet headers and trailers. Furthermore, as the technique is designed for mutiple-gateway internetworking systems, it is not employed in our one-gateway two-subnet internetworking system.

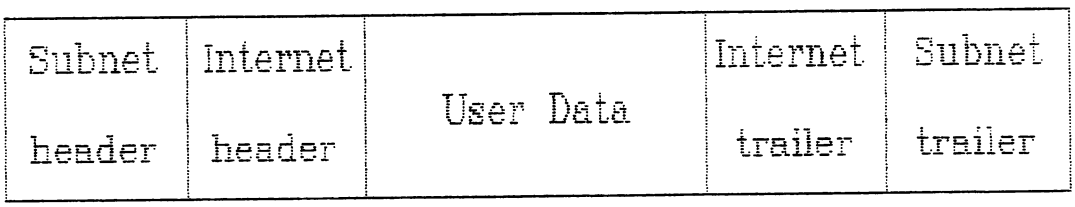


Fig. 4.1 Imbedded Internet Messages

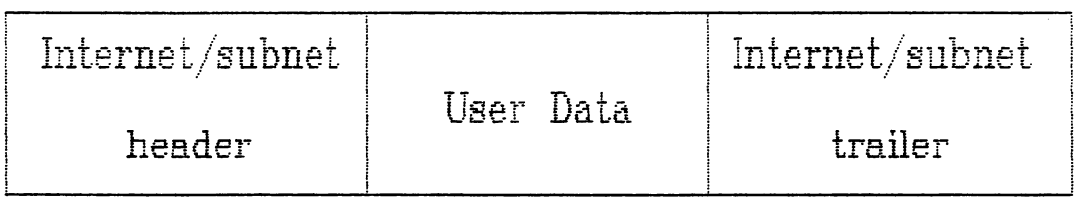


Fig. 4.2 Compatible inter/subnet Network layer

(2) If the Network layer of the subnet is designed with internetworking in mind, then the network header can be considered as the internet header. This is achieved by adding additional fields to existing protocols and formats (see Fig.4.2). This minimizes changes but allows internet and local message to be differentiated. Each subnet is able to interpret an internet address and route the internet message to an appropriate gateway. This method is employed in the implementation of the internet communications. The Network layer of each subnet is modified into internetworking functionalities by adding the source and destination network addresses to the existing formats.

4.3 Internetwork Architectures

The solution to the interconnection of different types of LANs is by the use of appropriate gateways and bridges. The type of gateway or bridge used varies depending on the extent of homogeneity between the constituent networks at the various layer protocols.

To assist in choosing an appropriate approach for implementing the internetworking functions, the similarities and differences between XLNET and TMS-IBM Token Ring at their respective OSI layers are listed in Table 4.1.

OSI layer	XLNET	TMS-IBM Token Ring
Application	Undefined	Undefined
Presentation	Undefined	Undefined
* Session	Designed by Gary Anido	Designed by T.L. Ng
* Transport	OSI Connection-oriented Transport protocol Class 4	Similar to XLNET
Network	OSI Connection-less Network protocol	Same as XLNET
Logical Link Control (LLC)	IEEE 802.2 Class 1 Connection-less operation	Same as XLNET
* Medium Access Control (MAC)	IEEE 802.4 Token Bus	IEEE 802.5 Token Ring
* Physical 1. Bit level synchronisation	Mutual method	Master clock method

2. Line code	Biphase-S code	Differential Manchester code
3. Transmission medium	Ribbon cable or optical fibre	Twisted pair or optical fibre
4. Electrical Interface	RS-422 standard	TMS 38051 & 38052

Table 4.1 The OSI layer protocol comparison between XLNET and TMS-IBM Ring

---

* The operating system of XLNET is written in 6809 assembly language and runs in the 6809 microprocessor and related chips (see in Chapter 2).

* The operating system of TMS-IBM Token Ring is written in Turbo Pascal Ver.3 and runs in the IBM XT/PC. (see in Chapter 3)

* The two systems use their own version of Session layer protocol. The comparison between them is made in Appendix B.1.

* Even though OSI Connection-oriented Transport protocol class 4 is selected for the Transport layers of both networks, some differences existing in the protocols between the two networks in this layer are explained in Appendix B.2.

* The MAC and Physical layers of TMS-IBM Token Ring are provided by the TMS-380 adapter chipset.

* More detailed information about the Physical layer in the two networks can be found in [25] and [20].

Of the possible approaches for internet gateway design, four are commonly encountered in the internet communication systems:

bridge, X.75, IP and protocol translator. Whether the above four approaches can be satisfied with the requirements of the proposed internet will be examined in the following sub-sections.

4.3.1 Bridge

A bridge [26,27,28] is a simplified gateway: it can be used to connect homogeneous networks. It operates within the Data Link layer, and higher Network layers are not involved. For LANs, the Data Link layer is further subdivided into the Logical Link Control (LLC) sub-layer and the Medium Access Control (MAC) sub-layer. Bridges are referred to as MAC layer bridges. As shown in Fig. 4.3, whenever user data is provided to LLC by an LLC user (normally it is an entity in the Network layer), the LLC appends its header and passes the composited data unit to MAC, which then appends its header and trailer to constitute a MAC frame. This frame is captured by a bridge, which then checks the destination address field of the frames on the LAN. If the MAC layer protocol is different from that associated with the destination LAN, the bridge replaces the MAC fields with the new MAC fields which are mapped from the source MAC fields to the destination MAC fields. Otherwise the bridge relays the MAC fields intact to the destination LAN.

However, on account of the dissimilarity between the two networks (XLNET and TMS-IBM Token Ring) of higher layers, a bridge implementation is not a viable solution.

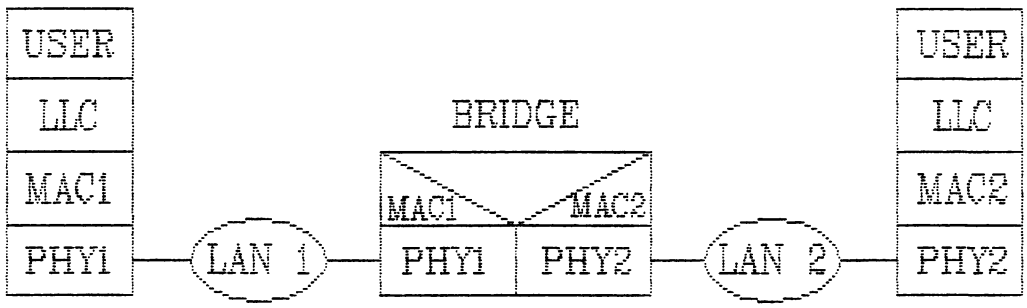


Fig. 4.3 Bridge Architecture

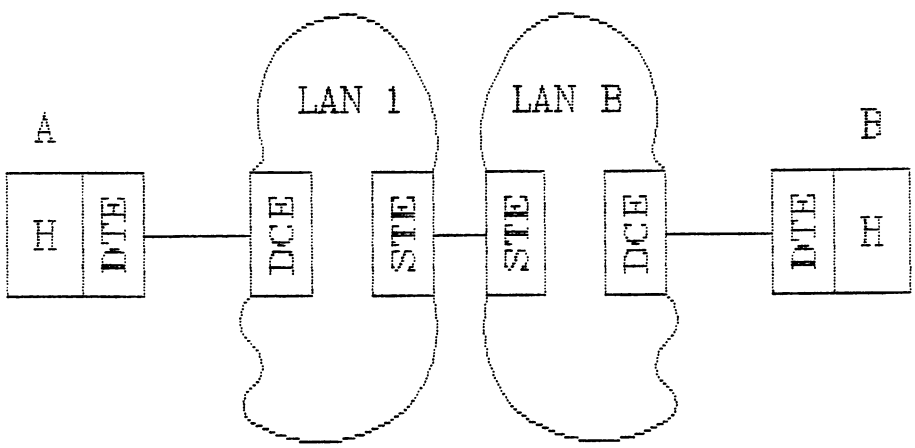


Fig. 4.4 Interconnection of X.25 Network via X.75

4.3.2 X.75

The X.75 standard was developed by CCITT as a supplement to X.25 [29,30,31]. It is designed for use between public X.25 networks and is not likely to be used as an interface between public and private networks. However, it could also be used to connect a collection of private X.25 networks in an internet that does not include public networks. It provides a connection-oriented service.

The transmission of a packet between two stations located in different LANs can be explained with reference to Fig.4.4. As shown, X.25 specifies an interface between a host equipment (data terminal equipment, DTE) and a network equipment (data circuit-terminating equipment, DCE), and X.75 uses a specific term for the network interface--signalling terminal exchange (STE) that acts as DCE-level gateways to connect two X.25 networks.

Station A sends an X.25 data packet to its DCE with the virtual circuit number that it associates with a connection to B. This packet is transmitted via LAN 1 to an STE. The STE uses the same format (Fig. 4.5), but with a modified virtual circuit number and flow control information for the appropriate STE-STE virtual circuit. The receiving STE then sends the packet to B's DCE, which presents a packet to B with the virtual circuit number that B associates with a connection to A.

If the higher layer (above Network layer) protocols are common in the subnets of an internet system, it is possible to use the X.25 protocol to implement the internet communication, even though all the subnets do not belong to the class of an X.25 network. At the sending

end, the local protocol is translated to X.25 protocol. This in turn is converted, in the second half of the gateway, from X.25 to its protocol.

For example, as shown in Fig.4.6, whenever gateway A receives an internet packet from LAN 1, it translates the local protocols and modifies the format to those of X.25 and then forwards the reformatted packet to another gateway B. Once gateway B receives the reformatted packet, it has to translate the packet into a protocol acceptable for transmission on LAN 2. As a result, a double translation is necessary to transmit an internet packet. In this case, each gateway has to contain two sets of translators, one for transmitting internet packets; the other for receiving.

If the X.25 protocol were to be implemented here, the necessary process, the double translation, would not be an efficient way to solve the problem. Further, the present higher layer protocols are not quite identical (see Table 4.1). Therefore, the X.25-X.75 Network layer gateway protocol is not adopted in the present implementation.

4.3.3 Internet Protocol (IP)

The IP is a protocol standard developed by DOD [34] as part of the DARPA Internet Project. It provides a connectionless or datagram service in the internet communications. The conceptual layer structure is shown in Fig. 4.7 [35].

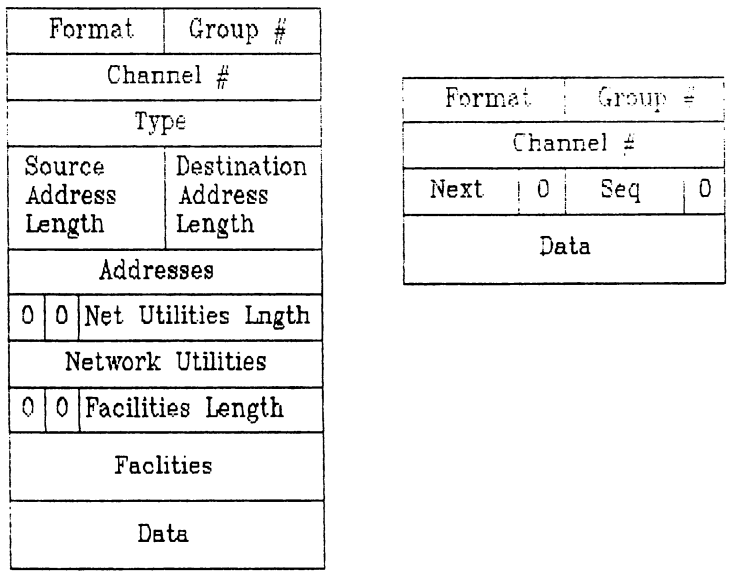


Fig. 4.5 X.75 package formats

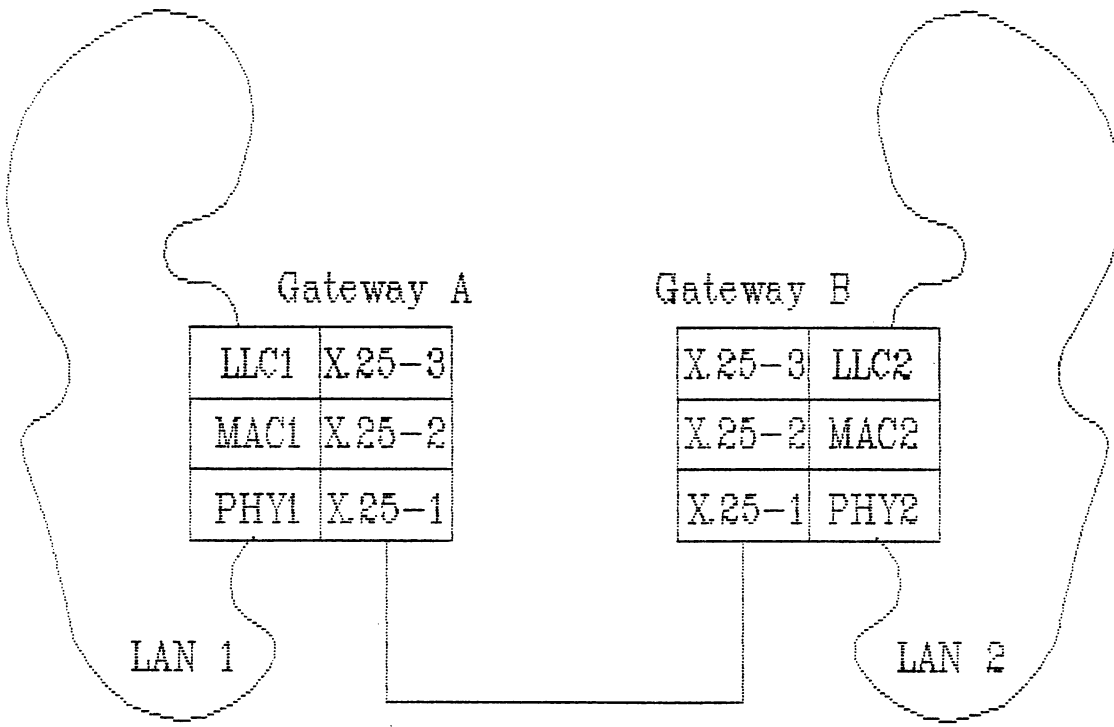


Fig. 4.6 X.25 protocol architecture

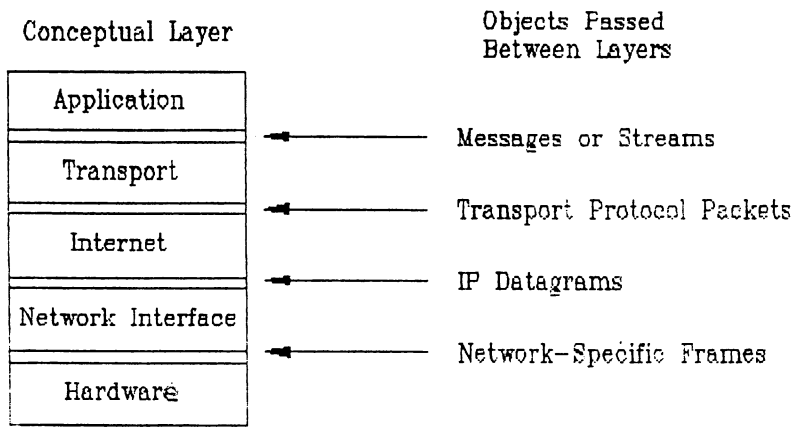


Fig. 4.7 The Conceptual Layer of IP

As an example, Fig. 4.8 depicts the operation of IP for data communications between station A on a LAN 1 and station B on a LAN 2. These two stations share a common transport protocol. The data to be sent by station A are encapsulated in a datagram (the datagram format shown in Fig. 4.9 [36]) in the Internet layer. Then the IP module in the station A specifies the global network address (station B) in the IP header and recognizes that the destination of the datagram is on another network. So the IP module appends to the IP datagram a LAN 1 header that contains the address of a gateway. For example, for XLNET or TMS-IBM Ring, a Network layer packet encapsulates the IP datagram to be sent to a gateway. When the packet is received by the gateway, because the packet contains an IP datagram, the LAN 1 header is stripped off, and the IP header is checked for validity. The destination IP address of a valid datagram is examined to determine whether the datagram contains control information intended for the gateway, or data intended for a station on LAN 2. Since the datagram is for station B on LAN 2 to which the gateway is directly connected, the gateway builds a new LAN 2 header for the packet and sends it to the destination, station B on LAN 2.

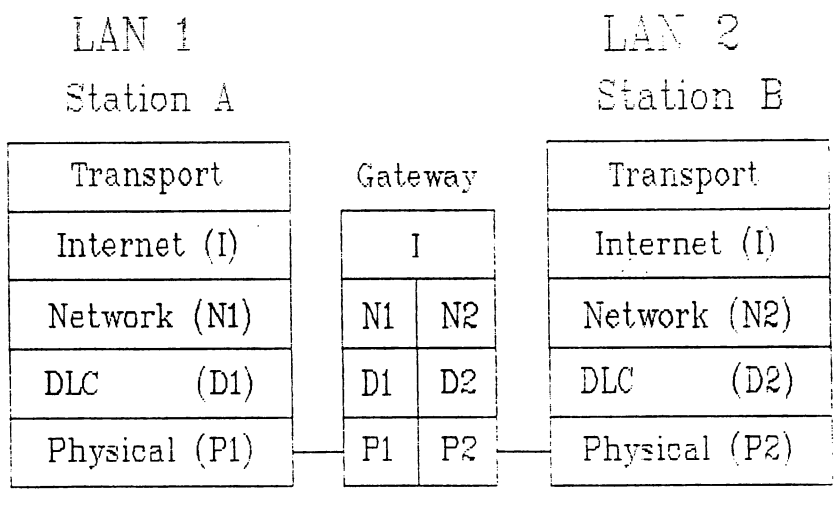


Fig. 4.8 Internet Prorocol

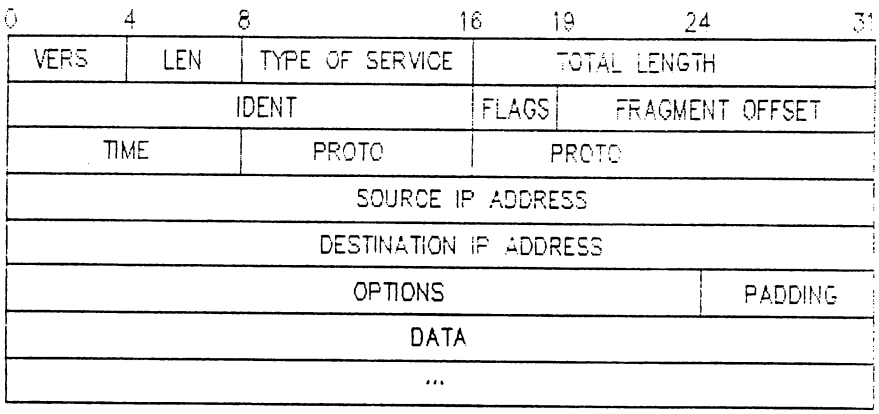


Fig. 4.9 **Format of an Internet datagram, the basic unit of transfer on the Internet.**

To provide reliable internet services, the vehicle also provides an internet control message protocol (ICMP) which is a required companion to IP. Basically, ICMP is used by gateways to report error and control information to internet stations and gateways. For example, when a datagram can not reach its destination, or when the lifetime of the datagram in a gateway expires, an appropriate message will be returned to the sender or gateways. Then the sender or gateways can respond appropriately.

As mentioned earlier, the fundamental requirement of using this approach is that the Transport protocols in the communicating stations must be common. Since the Transport protocols in the present subnets are not the same (see Appendix B.2), this approach is not selected for use in ~~the~~ our gateway.

4.3.4 Protocol Translator

From the preceding sub-sections, we know that the limitation in using the above approaches in the proposed internet is the degree of heterogeneity between the XLNET and TMS-IBM Ring. The vehicle, Protocol translator [37, 38, 39], is able to be used to interconnect heterogeneous networks that have dissimilar layer protocols up to the Application layer (OSI, layer 7), and has been employed in some major network architectures such as interconnection between IBM's SNA and Xerox's XNS [40].

As an example, Fig. 4.10 depicts the operation of the protocol translation for communications between station A on a LAN 1 and station B on a LAN 2. In a reference model of OSI, the various layers 1,2..., N, N+1, ..., 7 are identified for each layer of architecture of station

A, and likewise the layer 1,2,..., M, M+1, ..., 7 of architecture of station B.

Whenever the gateway receives a packet sent by station A, it first of all converts from the protocols of layer N to 1 in A to those of layer M to 1 of B respectively as shown in Fig. 4.10, then forwards the translated packet to the destination station B on a LAN 2, and vice versa.

It is arguable that the most natual situation for all the cases would be M=N. However, it is more often than not the case that M > N. For example, Deaton and Hippert [43] discussed the use of an X.25 virtual circuit (M=3) instead of an SDLC link (N = 2). Although it is a bit wastful to duplicate the layer 3 function, it has the benefits of low X.25 tariffs and the ability to treat end nodes (half-gateways) adjacent to one another.

From Table 4.1, it is obvious that in our case, the protocols from the Physical Layer to the Session Layer in the XLNET are different from those in the TMS-IBM Token Ring. Therefore, it is necessary to use the approach for our internet. The number N is given to represent the Session Layer in LAN 1, and the number M is given to represent the Session Layer in LAN 2. From the OSI protocol, N is equal to M. It is seen that the layers involved in the protocol conversion should be from the Session layer to the Physical Layer. However, as a part of the implementation, the shared memory scheme (see later) is installed at the level of the Network Layer of an intermediary or a gateway for the internet communication. Then, an internet packet is not physically transmitted and received from the Physical Layer of the gateway but

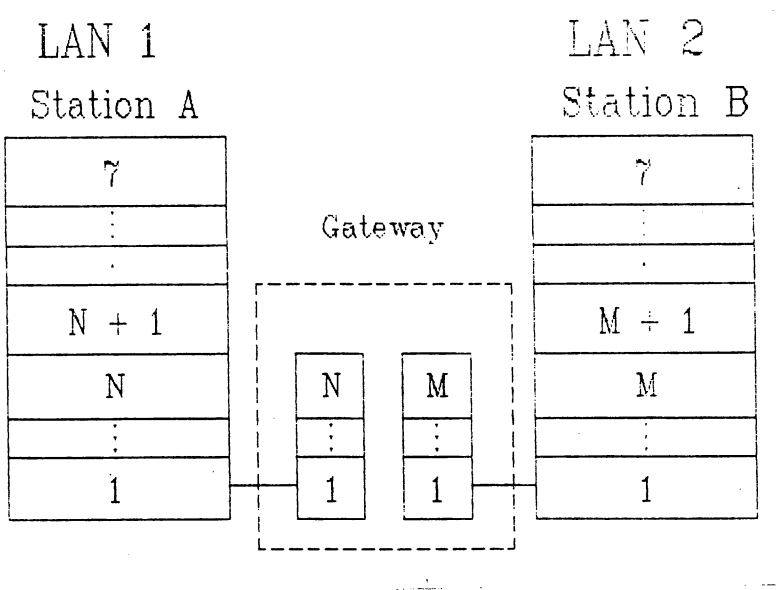


Fig. 4.10 Protocol Translator

from the Network Layer instead. Therefore, the layers involved in the protocol conversion are just the Transport Layer and Session Layer as shown in Fig. 4.11. Details of the scheme are given in section 4.3.4.1 and chapter 5.

4.3.4.1 Shared Memory Scheme

By the use of the characteristics of the dual-port RAM, the two separate I/O ports that each allows independent access to read or write to any location in the memory, this kind of memory can be employed as a physical path in the internet communication.

Once a half-gateway in a LAN recognizes a packet as an internet packet in its Network Layer, it puts the packet into the proper location of the memory through one I/O port. Then, the half-gateway sends an advice by an interrupt signal to its partner, the other half-gateway in the other LAN. At the other side, once the partner recognizes the advice, it is ready to pick the packet up from the memory through the other I/O port.

This approach employed in the internet system has the following merits:

1. The transmission delay in the medium can be considered to be equal to the memory access time. What is the value of the memory access time? The answer depends on what kind of the memory we choose. In our case, the memory access time of the dual-port RAM (see Appendix C) we used is just 45 nanoseconds. Meanwhile, the form of a packet transmission in the parallel 8 line ribbon cable is byte

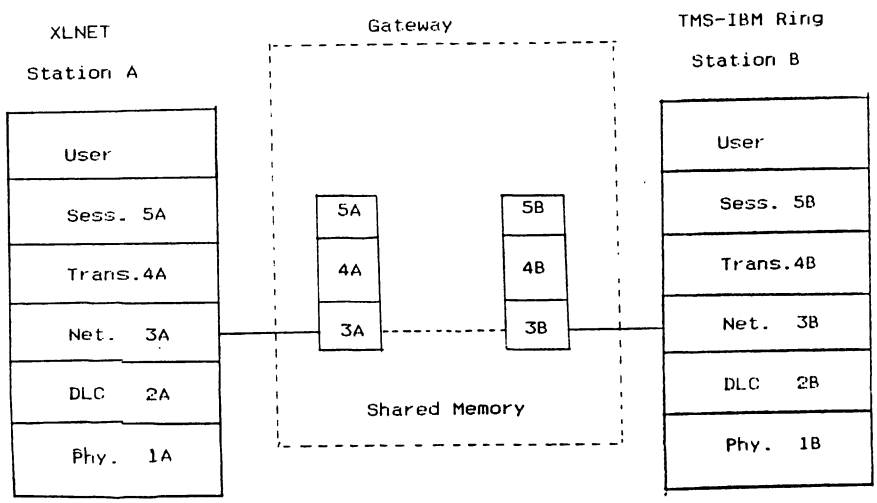


Fig. 4.11 Protocol translation by using the shared memory scheme.

by byte. The maximum transmission speed can be up to 8 bits/45 nanoseconds which is about 178 Mbits/sec. Of course, the speed is significantly faster than RS-232C 9.6 Kbits/sec. in the traditinoal way, that is, in the form of bit by bit transmission.

2. Since the main functionality of the Network Layer is to route a packet through a proper path to its destination, if the shared memory scheme is employed in the layer, an internet packet is not necessary to be served by the DLC Layer and Physical Layer of a half-gateway to the other half-gateway then its destination, but directly by the Network Layer through the dual-port RAM instead. By this way, the protocol conversions for the DLC Layer and Physical Layer are delimited. Meanwhile, the time delay spent on processing an internet packet in the DLC Layer and Physical Layer is saved. So the transmission speed for the internet packet can be increased. Concurrently, the cost of the physical equipment for the protocol conversion in the Physical Layer (e.g. the line code transformer might be needed for the translation of line code from the Biphase-S code to the Differential Manchester code, and vice versa.) is saved as well.

The drawback is that a short physical distance between the two subnets is required. Since an internet packet is transmitted byte by byte in a parallel form, the signal by parallel transmission cannot suffer large distortion incurred in the long distance transmission. However, the extension of the distance between the two subnets is still possible by using the signal repeaters in the appropriate location to supply the signal in the path.

4.4 Communication Services

The communication service in a network or an internetwork is chosen from one of two services. The first service is connection-oriented; the second is connectionless. A brief description of each service will be presented in the following paragraphs.

As shown in Fig. 4.12, a connection-oriented service provides a substantial amount of protection for the user data. Most connection services have three phases which are "Connection establishment", "Data transfer" and "Connection termination".

The connection must be established before data transfer takes place. Typically, to achieve the agreement to exchange data between two entities, one entity issues a connection request to the other, then the receiving entity either accepts or rejects the request. The establishment phase can be used to negotiate quality of service or operations, such as timing restriction or the length of a packet in the data transfer.

Following connection establishment, the data transfer phase is entered. During this phase, both control and data information is exchanged. Finally, a termination request is sent by one of the two entities which wants to terminate the connection.

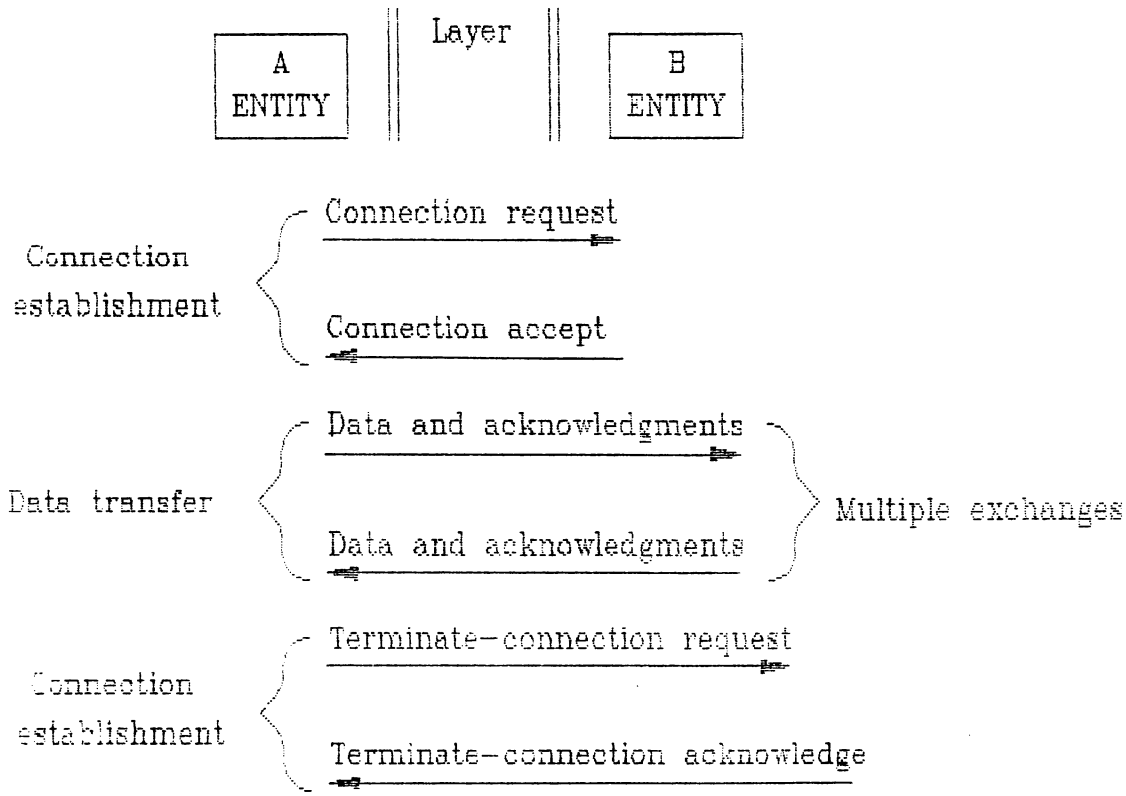


Fig. 4.12 Connection-oriented Services

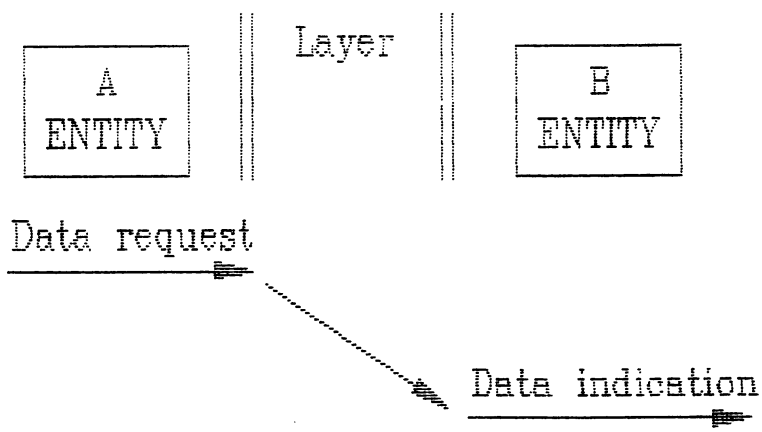


Fig. 4.13 Connectionless services

In contrast, connectionless services (sometimes called datagram services) are simple to implement in that the user is not provided with any form of response to a transaction, each transaction is independent of previous transactions and the user transmits data without prior coordination. As shown in Fig. 4.13, by using the connection-less service, the variety of service protocol is reduced; only one protocol over all subnetworks. However, duplicate messages can easily result from Data-link or Network layer retransmissions. Messages may also be discarded due to corruption of data or lack of buffer space. Therefore, frequently connection-oriented services are preferred.

From the above, it is clear that each service has its merits and demerits. But, based on the considerations given below we settle for a connectionless service. The following are the reasons:

(1) In our task, the integrated communications have critical real-time requirements for synchronous services, such as voice. Since packets must be delivered within a strictly bounded time, they can not tolerate the delay penalties associated with the sequence and flow control which might be incurred by the use of the connection-oriented services.

(2) If the connection-oriented technique were employed, the gateway would be more complex. For meeting the requirements of the connection-oriented service, the gateway would not only have to preallocate the space for the table but also process it for maintaining connection state information. Furthermore, the connection-oriented services incur more protocol overhead traffic than connectionless ones. These would result in significant process delay in the data transfer

through the gateway. Therefore, the connectionless services are more suitable for real-time communication (e.g. voice).

(3) The reliability of a connection-oriented service is obtained at the expense of performance. According to Meister [41], the throughput provided by connection-oriented services can be expected to be lower than by connectionless ones, since the connection-oriented services are more complex and need more protocol overhead traffic than connectionless ones.

(4) Furthermore, in both subnets, the network service (layer 1-3), which is common to all applications, is designed to provide an efficient and reliable connectionless network service to the connection-oriented Transport Layer which has functions, such as error control and flow control, to provide connection-oriented services. Due to the considerable amount of protocol architecture involved in both subnets, it is not really necessary to use the connection-oriented services in the internetworking communications.

4.5 Addressing

In both XLNET and TMS-IBM Token Ring, the Network layers are not designed with internetworking in mind. For interconnecting them and transmitting messages across each network boundary, it is necessary to add this function to the Network layer. Hence the addressing must be re-considered.

In a datagram network, every message contains source and destination station addresses. The station address is 8 bits long in XLNET, while it is 48 bits long in TMS-IBM Token Ring. Whenever a station is connected to its own LAN, it is assigned a unique and

specific length (8 bits in XLNET; 48 bits in TMS-IBM) address. Before an XLNET station is connected to the ring, its address must be assigned by the station user, otherwise it is considered as a repeater in the link. An XLNET station's address can only be changed by the user while it is in the operation mode of a repeater. For TMS-IBM Ring, a station's address must be assigned by the station user, otherwise it is not able to be connected to the link.

For communicating in an internetworking environment, there are two choices for addressing stations:

(1) Unique Global Address

That is, there is a unique identifier for each station in the internet. This implies that a gateway would need to derive subnet addresses from station addresses. The technique has been proposed by the developers of Ethernet [42]. The principal advantage of this approach is that it is able to make stations portable. That means it permits stations to move physically from one network to another without changing stations addresses. The primary disadvantage is that very large routing tables are needed, indexed by station address to determine the subnet and gateways to be used to reach the station.

If this technique were to be chosen, it would be necessary to change the present and simpler subnet addressing scheme and substantially effect the subnet independence. For this reason, it will not be adopted for our application.

(2) Network Specific Address

A gateway receives an internet packet with a reference in the form "subnet.station", where "subnet" is a subnet address. The

"station" component needs only be unique within the subnet, although the combined address is globally unique. The "subnet" is used for routing to the final subnet, thereafter the "station" is used for identifying the destination station in the subnet. The Darpa IP [34] uses this addressing scheme. The primary shortcoming [42] of this addressing mode is that additional administrative procedures are needed to assign station addresses within a subnet, and possibly translate different length station addresses within a gateway.

4.6 Segmentation and Reassembly

In general, subnets (no matter how heterogeneous) within an internet have different maximum (and sometimes minimum) packet sizes. Thus gateways may need to segment incoming datagrams into smaller pieces before transmitting into the next network. This process is called segmentation, or fragmentation.

With the proposed internet, although the packet size defined in the two non-homogeneous subnets is not the same (160 bytes in TMS-IBM Token Ring; 137 bytes in XLNET), the size of data fields is identical (128 bytes). Although the fragmentation could be implemented in this case, as explained above, it would result in an inefficient and unnecessarily complex procedure. By using the shared memory scheme mentioned in section 4.3.4.1 and chapter 5 to implement the protocol translation and frame reformation, the transfer of the user message in the data fields from an XLNET packet to an TMS-IBM Ring packet is executed as the exchange of the memory location, and vice versa. Then, the segmentation and the later reassembly are not necessary.

The shortcomings of the segmentation and reassembly are:

(1) As the following fields are required in the protocol control header to support segmentation and reassembly, the smaller is the block, the greater is the percentage overhead.

a. More Bit.

This indicates whether the packet is part of a segmented service data unit or not. The last packet of the data unit has the bit cleared.

b. Data Unit Identifier.

This identifies to which datagram a segment belongs.

c. Offset.

This provides the position of a segment in the original datagram to enable reassembly and detect lost segments.

d. Total Length

This indicates the length of the segmented datagram and allows reservation of buffer space.

(2) Doing the process of segmentation will increase the gateway process time, and , therefore, delay.

(3) Segmentation implies reassembly processing. This adds further to the delay.

a. The simpler solution is to make the Reassembly to be performed at the destination. The principal drawback of this approach is that the incoming packet is split into two packets in the transmission as data moves through the internet, and vice

versa in the opposite direction. This may decrease the overall internet throughput in the high speed internet.

b. The alternative is to have the reassembly processed at the gateway. By the use of this scheme, large buffers are required at a gateway, and there is a potential for reassembly congestion.

(4) One packet arrival will generate an interrupt that must be served. Smaller blocks result in more interrupts.

4.7 Routing

The routing refers to the process of choosing a path over which packets are sent.

The routing algorithm as used in our case of two interconnected rings is simple and straightforward, since one part of the destination address can be used to determine the route. Whenever the gateway receives a packet, it extracts the subnet address from the destination address and compares it to its own subnet address. A match means the destination of the packet is located within its own network, otherwise within the other network.

Here, we see one of the advantages of the network specific address scheme, namely that because the addresses of all stations on a single network include a common network address, and because extracting that address can be done in a few instructions, the process of routing is extremely efficient. It is this addressing scheme that has been adopted for the implementation of the gateway between XLNET and TMS-IBM Ring.

CHAPTER FIVE

LABORATORY IMPLEMENTATION

5.1 Introduction

The aim of the laboratory implementation in this project is to give a solid verification of results obtained in Chapter four. Section 5.2 introduces the operation of the gateway in our case. To demonstrate the functionalities of the gateway in the considered internet system described in section 5.2, the hardware aspects of the gateway and the relevant software will be discussed in section 5.3 and section 5.4, respectively.

5.2 Overview of The Experimental Gateway

With reference to the topology of the proposed internet as shown in Fig.5.1, we describe below the operation of the proposed gateway. We assume the following conditions apply:

1. The nodes B and C are designed to perform two functions: (1) each node acts as a local node in its own LAN and (2) each node can act as a half-gateway in the internet system.

2. Assume node A in XLNET wants to transmit a data/voice packet to node D in TMS-IBM Ring.

3. The priority of each class of packet in the internet system is listed in Table 5.1.

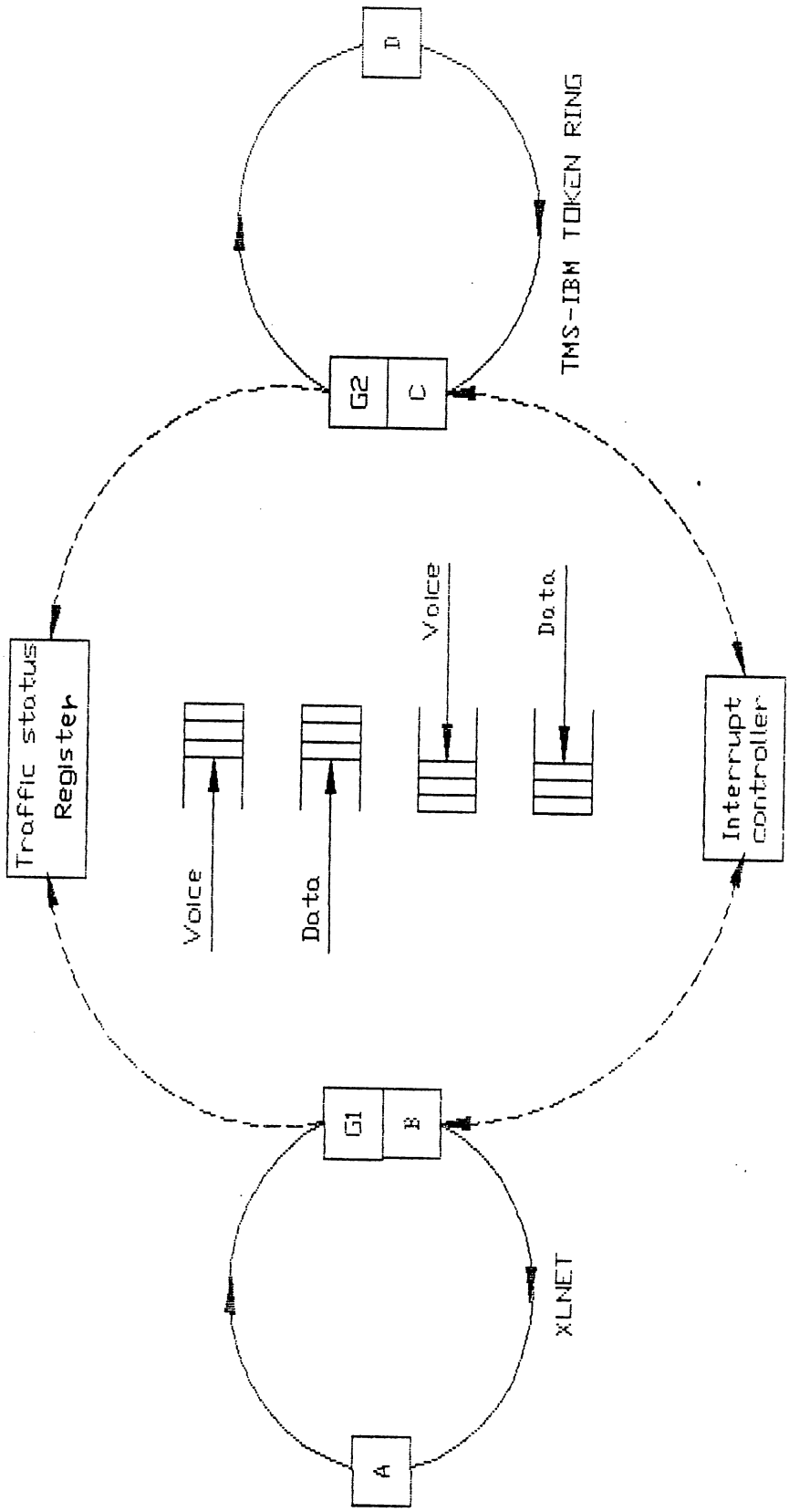


Fig. 5.1 The topology of the internet

4. In operation, if the packet is a voice packet, node A will capture the token in the first sub-cycle and transmit the packet into its ring. Otherwise the packet must be a data packet, then node A will capture the token in the second sub-cycle and transmit the packet into its ring.

5. The packet will be copied by half-gateway 1 (G1) into an appropriate buffer.

6. Gl reads the address of the packet received and if the packet is intended for TMS-IBM Ring, Gl will forward it to half-gateway 2 (G2). Otherwise, Gl handles it as a local packet for its subnet.

7. Each half-gateway maintains two queues: one for voice and the other for data. Depending on the type of packet received, a half-gateway will store it in the appropriate queue.

8. The gateway uses the Priority Status Register to determine the priority of the packet received. The action of G2 depends on the priority of the current packet on its subnet (TMS-IBM Ring) and the priority of the packet received from G1. The packet priority is given in Table 5.1. From the table we see that an internet voice packet is provided with the highest priority to be processed in the gateway. The control of action by G2 is by interrupts generated by G1 in line with the priority status. Should it happen that G2 is unable to accept the packet from G1 (e.g., because of lower priority of the packet or overload), G1 will try again after a random delay, up to 100 times, when it will issue an error message.

9. G2 on receiving a packet performs a frame reformat and a protocol translation from XLNET to TMS-IBM Ring.

Class of packet		Priority
Internet	Voice	4
	Data	2
Intranet	Voice	3
	Data	1

* The higher figure, the higher priority.

* The priority of an internet voice packet is the highest, and that of one of an intranet data packet is the lowest.

* A voice packet is always provided with higher priority than a data packet.

Table 5.1 Packet priority list in the internet system

10. The arriving packet is routed to the proper destination, node D in TMS-IBM Ring.

11. The error control is carried out by the communicating partners (node A and D), that is G1 and G2 are not involved in the ACK and NAK protocol.

12. In acknowledging the received packet, node D sends an ACK packet to node C but with the information field containing the destination address of the XLNET node. G2 on reading this address will forward the ACK packet to G1 with the corresponding priority on the reverse procedure above and therefore it will be sent to node A.

The schematic block of the gateway is given in Fig. 5.2. The gateway software is executed by the hardware of each half-gateway.

5.3 Hardware of the Gateway

A gateway of the type considered here is functionally composed of three parts as shown in Fig. 5.2: an XLNET's nodal hardware (6800 family), a TMS-IBM Ring's nodal hardware (IBM XT/PC and TMS-380 adapter chipset) and the gateway interface card. The architectures and functions of the nodal hardware of each of the half-gateway have been described in detail in Chapter Two and Chapter Three.

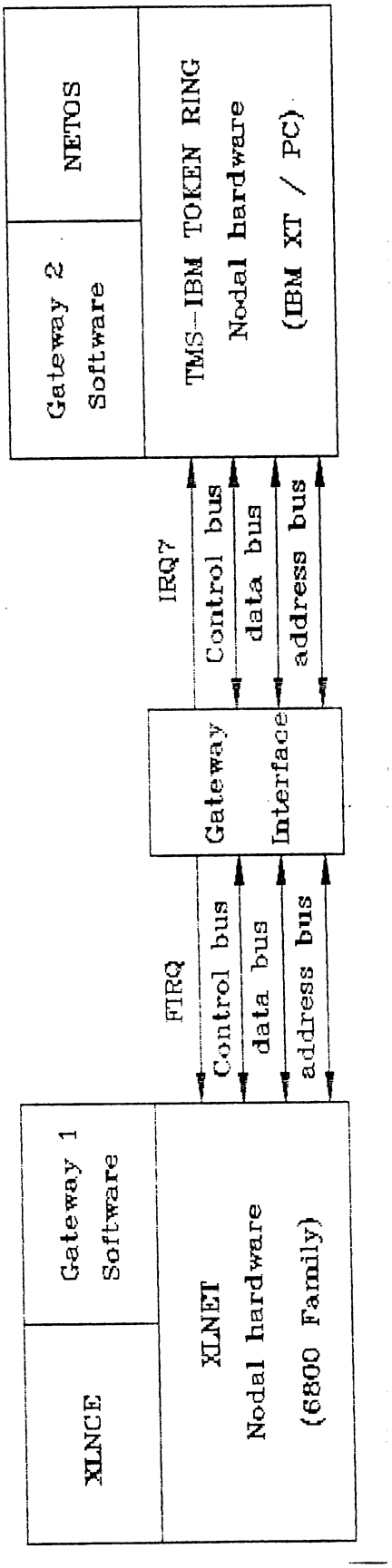


Fig. 5.2 The Proposed Gateway

Each nodal hardware, acting as a half-gateway in the internet system, is used to transmit and receive internet packets, and is supplemented by a number of appropriate software processes (such as protocol translation for transmitting various internet packets) so as to maintain the internet communication. In addition, since the 4 Mbits/sec processing rate of the TMS-IBM Ring processor is faster than the 1.5 Mbits/sec rate of the XLNET's nodal processor, the IBM model hardware is used to handle the frame reformatting and the protocol translation for the internet communication of the two non-homogenious subnets, XLNET and TMS-IBM Ring.

The gateway interface card is designed to provide a path for the internet communication. It consists of the following hardware parts as shown in Fig. 5.3:

1. I/O buffers.
2. Packet queues.
3. Priority status register.
4. Interrupt controller.

The I/O buffers are designed to provide an interface between the nodal hardware and the gateway interface card. Meanwhile, the I/O buffers are employed to improve noise rejection and high fanout for the asynchronous two-way communication between the nodal hardware system bus (including address bus and data bus) and the packet queues and priority status register.

The packet queues are used to temporarily accommodate the internet packet received from the gateway. In this thesis, the packet queues are designed to buffer the two different kinds of packets, i.e., data and voice packets. To accommodate and distinguish the internet

packets from different subnets, the packet queue is arranged into two groups. Each group for one direction (e.g., from XLNET to TMS-IBM Ring or vice versa) contains two queues; one for data and one for voice. In this implementation, the packet queues and the priority status register are designed by means of the VLSI VT7132A 45pc 2k byte dual-port RAM chip, which features two separate I/O ports. Each port allows independent access to read or write to any location in the memory. Details of this chip are given in Appendix C.

In the packet queues, one I/O port is accessed by a half-gateway in XLNET, the other by that in TMS-IBM Ring. The addresses of the I/O port accessed by one XLNET node acting as a half-gateway in XLNET, are in the range from $8000 to $87FF (described in hexadecimal code) within the addresses for 48 Kbytes of RAM of the XLNET nodal hardware. However, in order to avoid changing the commercial 640 Kbyte memory system of the IBM XT/PC, the address of the other I/O port accessed by one TMS-IBM Ring node acting as a half-gateway, are in the range from $DF800 to $DFFFF (described in hexadecimal code). These addresses are not defined within the addresses of the 640 Kbytes of RAM of the TMS-IBM nodal hardware. In other words, the extension address area in the TMS-IBM Ring nodal hardware is specially provided for the 2 Kbyte dual-port RAM.

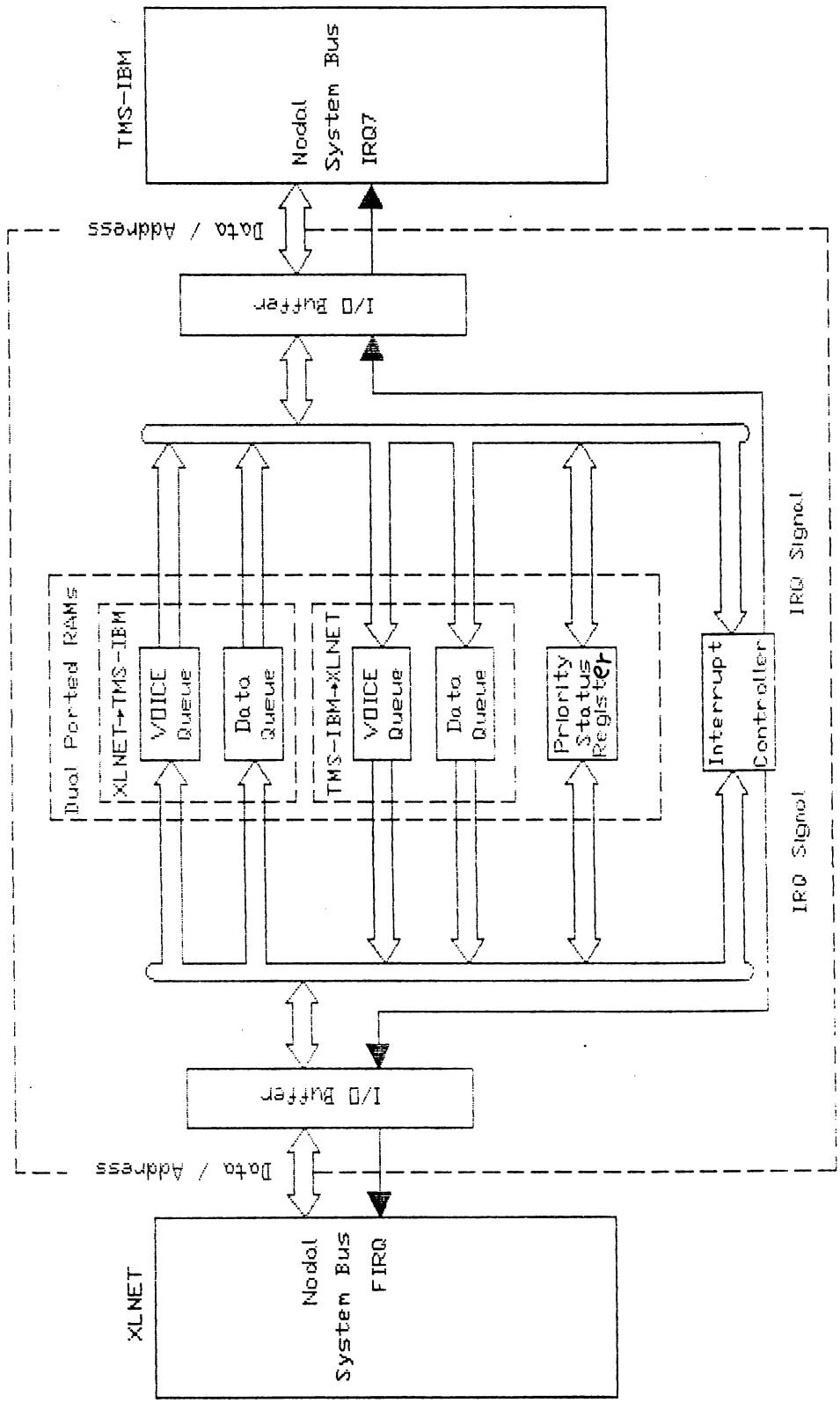


Fig. 5.3 The architecture of the gateway interface card

In order to avoid the collision incurred by an internet packet and an intranet packet arriving simultaneously in a half-gateway, the priority status register is designed with the related software to provide the priority of the current packet of the half-gateway in the destination subnet to the other half-gateway which forwards the internet packet. Two bytes of the 2 Kbyte dual-port RAM are employed in the register. Each byte is used to contain the priority status of the current packet in each half-gateway. Whenever a half-gateway begins to process a packet which may come from internet or intranet, it places the priority of the packet into the register, and replaces the content by zero when it finishes processing the packet.

Consequently, whenever a half-gateway is interrupted to process a packet (which may come from internet or intranet), it compares the priority of the intended packet with that of the priority status register. If the priority of the packet is higher than that of the register, the priority of the register will be replaced to be the higher priority. In this way, the half-gateway will be correctly interrupted to process the higher priority packet. If the priority of the packet is lower than that of the register, the next interruption for the packet will be invoked after a random delay, until the packet is processed or discarded. This will be carried out for a limited period of time (here 100 retries) after which an error message will be sent. If an internet packet and an intranet packet, both having the same priority, arrive in a half-gateway, the intranet packet will be given the preference.

The interrupt controller is designed to generate an interrupt signal to inform the half-gateway in the destination subnet to handle the internet packet from an appropriate packet queue.

The method to generate an interrupt signal by the interrupt controller is quite simple. Whenever a half-gateway in line with the priority status wants to ask the other half-gateway to handle an internet packet, the former half-gateway is asked to read the content of a specific memory location of the dual-port RAM. What is read is not important, but the location accessed is the point to generate an interrupt signal, by a logical decoding circuit in the interrupt controller. Details of the interrupt controller are given in Appendix D.

5.4 Software Aspects of the Gateway

To achieve the internet communication with the hardware architecture of the gateway mentioned in section 5.3, the following four software parts are used in the implementation:

1. Addressing.
2. Routing.
3. Frame reformation and mapping.
4. Protocol translation.

The parts (1) and (2) of the above are applied to the individual half-gateway itself, while the other parts are settled into the half-gateway in TMS-IBM Ring.

It is worth mentioning that the parts (3) and (4) could be performed by a specific processor. However, in our case, the half-gateway in TMS-IBM Ring, for its faster processing rate than that in XLNET, is also used to do these, based on the architecture of the internet as mentioned in section 5.3. The approach has the advantage of saving the cost of an extra hardware which is used to perform parts (3) and (4) in the whole system. The cost saving and the successful

implementation outweights the fact that the extra load on the half-gateway (or end node) in the TMS-IBM Ring makes the performance of the Ring system somewhat degraded.

Each part of the above will be presented in the following sub-sections.

In our implementation, the XLNET and TMS-IBM Ring are shown in Fig. 5.4 and Fig. 5.5, respectively. The gateway interface card is shown in Fig. 5.6 while the connection is by a ribbon cable shown in Fig. 5.7. The whole internet system is shown in Fig. 5.8.

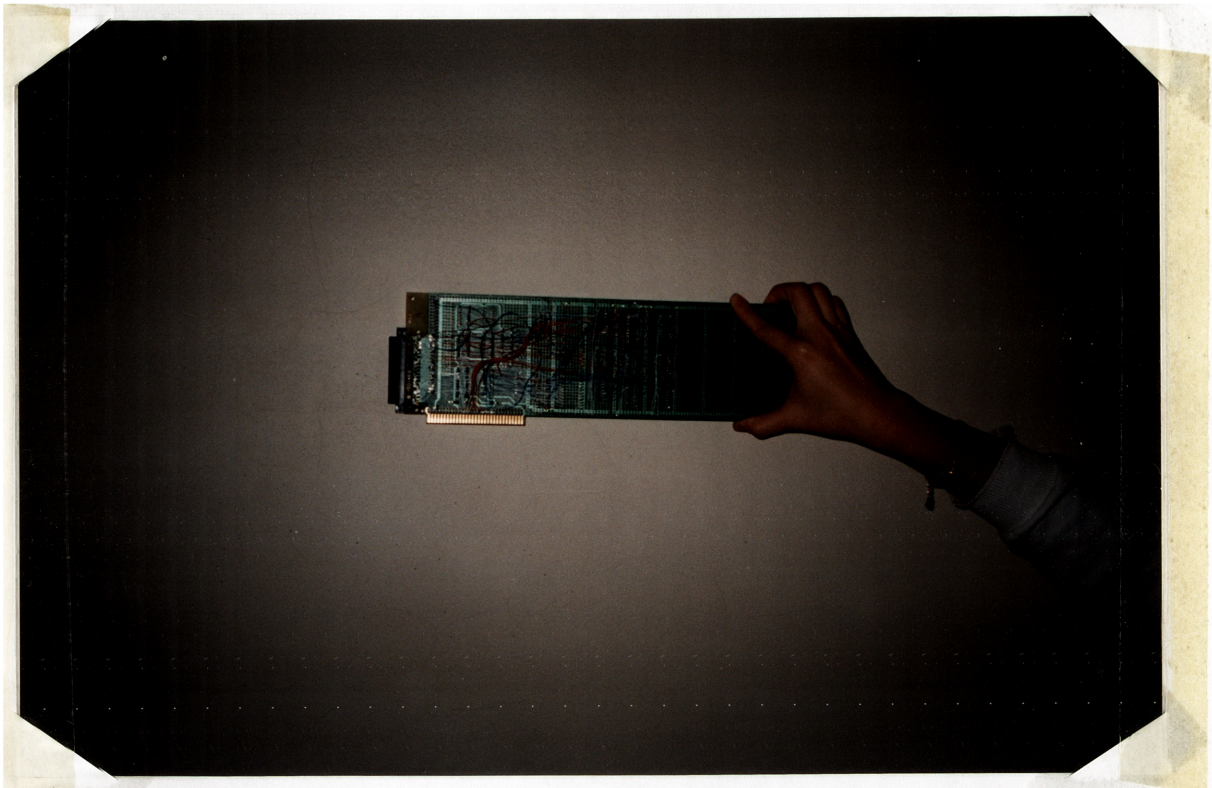
5.4.1 Addressing

The key requirement to transmit or receive an internet packet is the "addressing". In the existing Network layer of both subnets, the subnet-wide addressing and routing functions have been provided to the subnetworking communications. What needs to be done is to add the Network layer and to provide the network-wide addressing and routing freedoms.

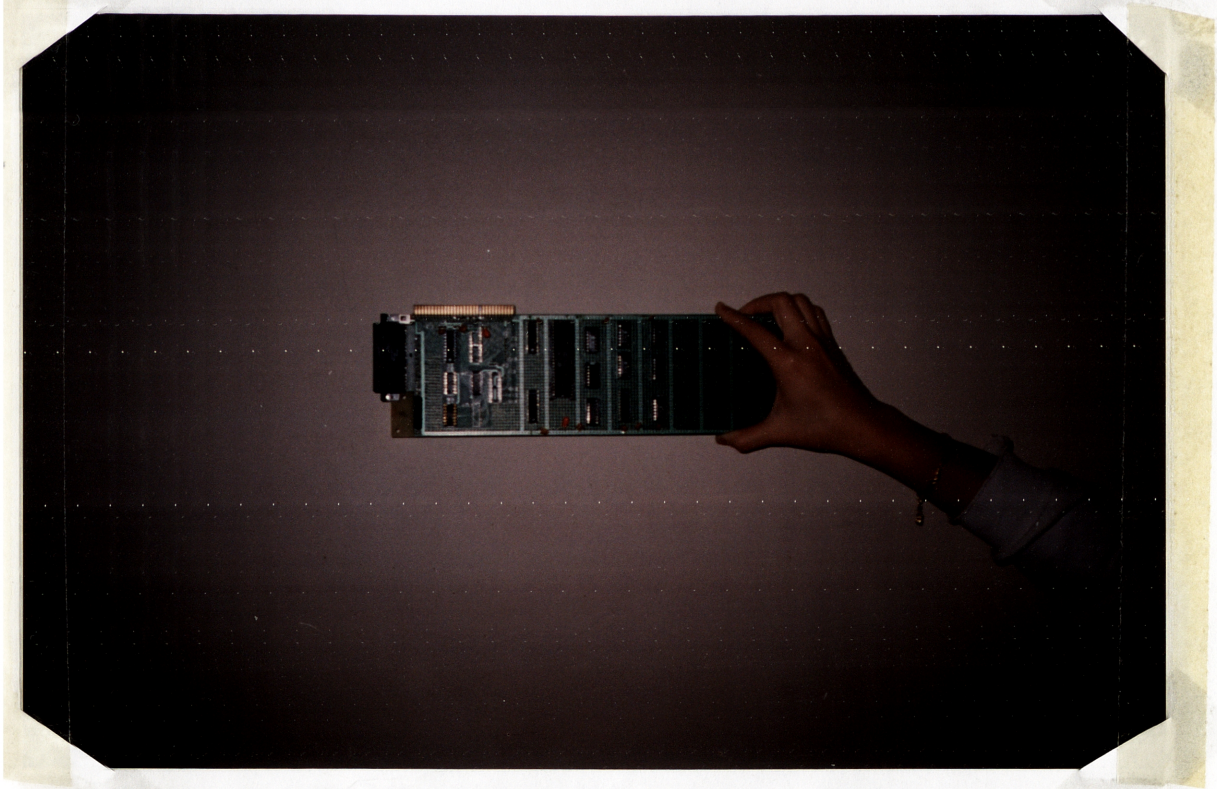
Given that the XLNET's node must be numbered within the range from 0 to 255 (described in the decimal code), and according to the regulation of addressing in the TMS-IBM chipset which performs the functions of the Physical and MAC layers, the TMS-IBM Ring's nodes must be numbered within the range from 400000 to 7FFFFF (described in the hexadecimal code). To send an internet packet, the sending node simply addresses the packet to a half-gateway, then the gateway converts the format of the packet to that of the destination subnet.

Fig. 5.4 The experimental XLNET

Fig. 5.5 The TMS-IBM Token Ring



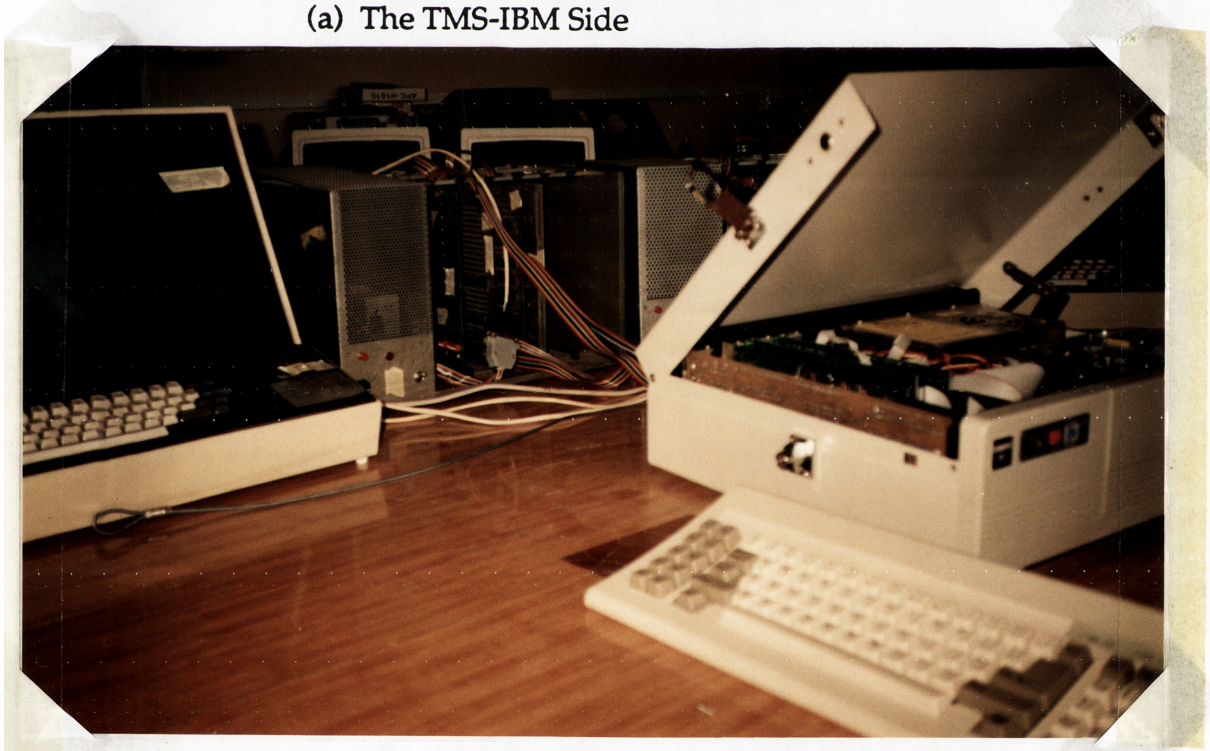
(a) Wiring Side



(b) Component Side

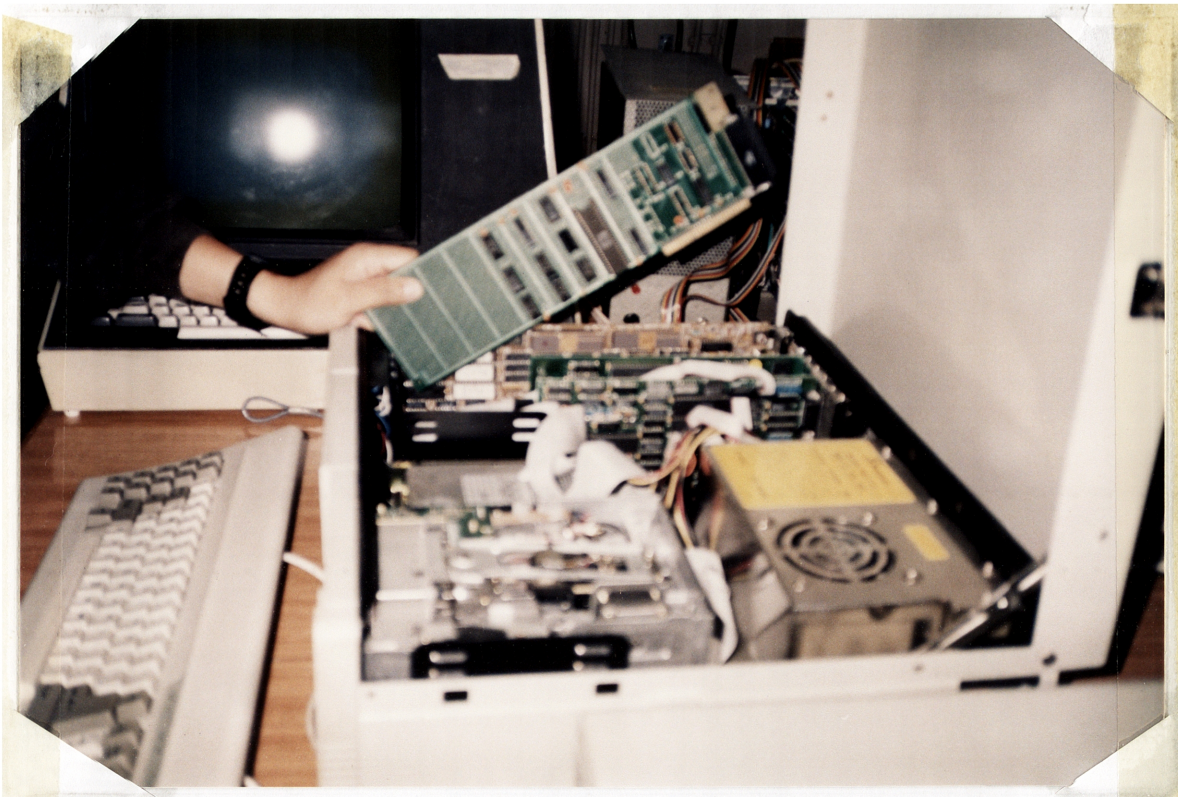
Fig. 5.6 The Gateway Interface Card

(a) The TMS-IBM Side



(b) The XLNET Side

Fig. 5.7 The Connection of the Gateway Interface Card



(c) The connection between the interface card and the TMS-IBM Ring's nodal hardware

Fig. 5.8 The Form of the Experimental Internet

This approach is simple but not practical in the internet. Since, in the internet structure of our case, a half-gateway in a subnet is meant to carry out its function in the internet, in addition to acting as a local node. Furthermore, the software relating to the addresses in TMS-IBM Ring is part of the MAC-Sublayer firmware and is not accessible. So it is impossible, at the present stage, to assign two different addresses, one to the local node and the other to the half-gateway, to any node in both rings. Under these circumstances, whenever a node in a subnet wants to send a packet to node K in the other subnet, it has to specify that the destination of node K, is not in its subnet, but in the other subnet. To achieve the objective, extra bits in the packet header need to be assigned to pursuit correct routing. In our case, the LAN address fields in both rings are 8 bits long, thereby permitting up to 256 distinct LAN addresses.

In addition, an extra destination node address field (6 bytes) called the Real Destination Node Address (RNDA) field, is necessary to be added in the packet header in TMS-IBM Ring. Under the transmission policy provided by the inaccessible firmware of the TMS-380 chipset, the internet packet sent from TMS-IBM Ring must be destined to the half-gateway in TMS-IBM Ring first by inserting the address of the gateway into the Destination Node Address field. Then the half-gateway will use the content of the Real Destination Node Address field to provide the real destination node address to the process of the frame reformation and mapping (see later). Finally the reformatted internet packet will, based on the address, be sent to its destination by the half-gateway in XLNET. Otherwise, the packet will be discarded in its source ring and cannot be transmitted to its destination.

Even this approach violates both the principle of not requiring existing networks to change their protocols and the principle of not degrading intranet traffic for the benefit of internet traffic.

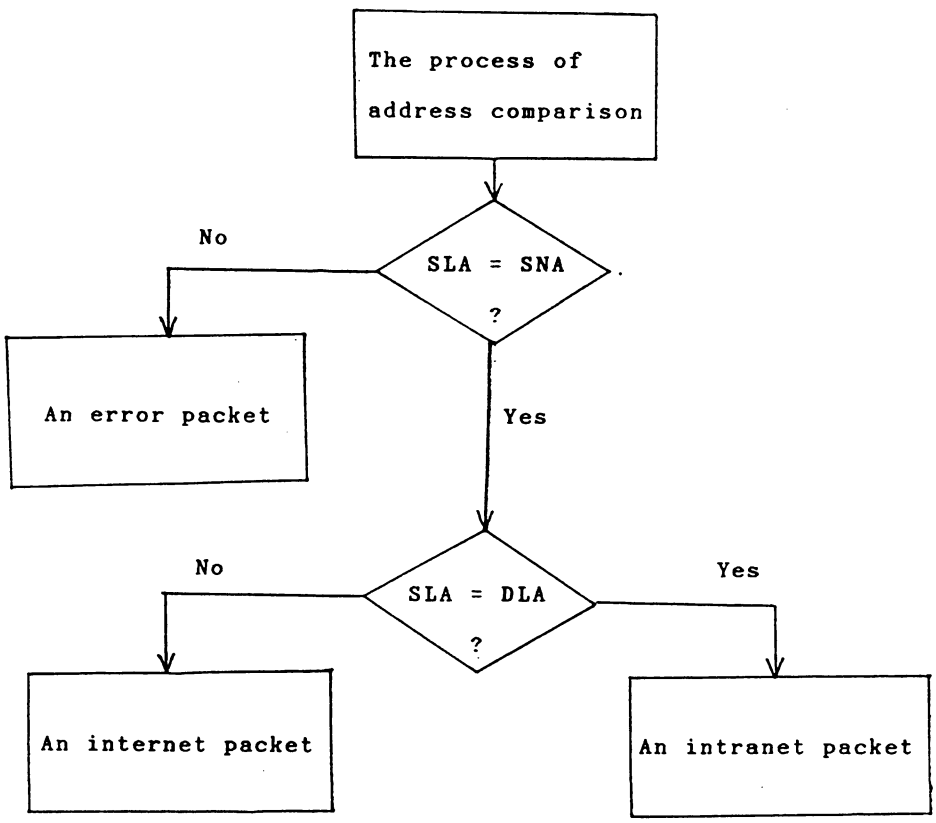
5.4.2 Routing

For transmitting an internet packet, the two new fields, Destination LAN Address (DLA) and Source LAN Address (SLA), are designated by users and are added in the packet header in the Session layer. In addition, the new field in subnet system, subnet address (SNA), is designed for internet routing. The field is defined by users when the subnet system is set up. The decision of packet routing is based on the comparison among the three address fields: DLA, SLA and SNA, as explained below.

Within an end node (or a half-gateway), when a packet is served down to the Network layer from higher layers for transmission, the following three kinds of results are possible, based on the addresses comparison as shown in Fig. 5.9:

(1) If the SLA is the same as the SNA and DLA, the packet is considered as an intranet packet intended to the subnet itself. Then the packet will be transmitted down through the Data Link Control and Physical layers into the ring of the subnet.

(2) If the SLA is the same as the SNA but different from the DLA, the packet is considered as an internet packet which is intended for the other subnet. Then the packet will be served by the shared memory scheme and sent to the Network layer of the other half-gateway in the other subnet.



* DLA = Destination LAN Address.

* SLA = Source LAN Address.

* SNA = Subnet Address.

Fig. 5.9 Flow chart indicating the logical decisions to be made in routing a packet to correct destination. Case 1: the packet is from higher layers to the Network layer of a half-gateway.

(3) In other conditions, the packet is considered as an error packet and will be discarded.

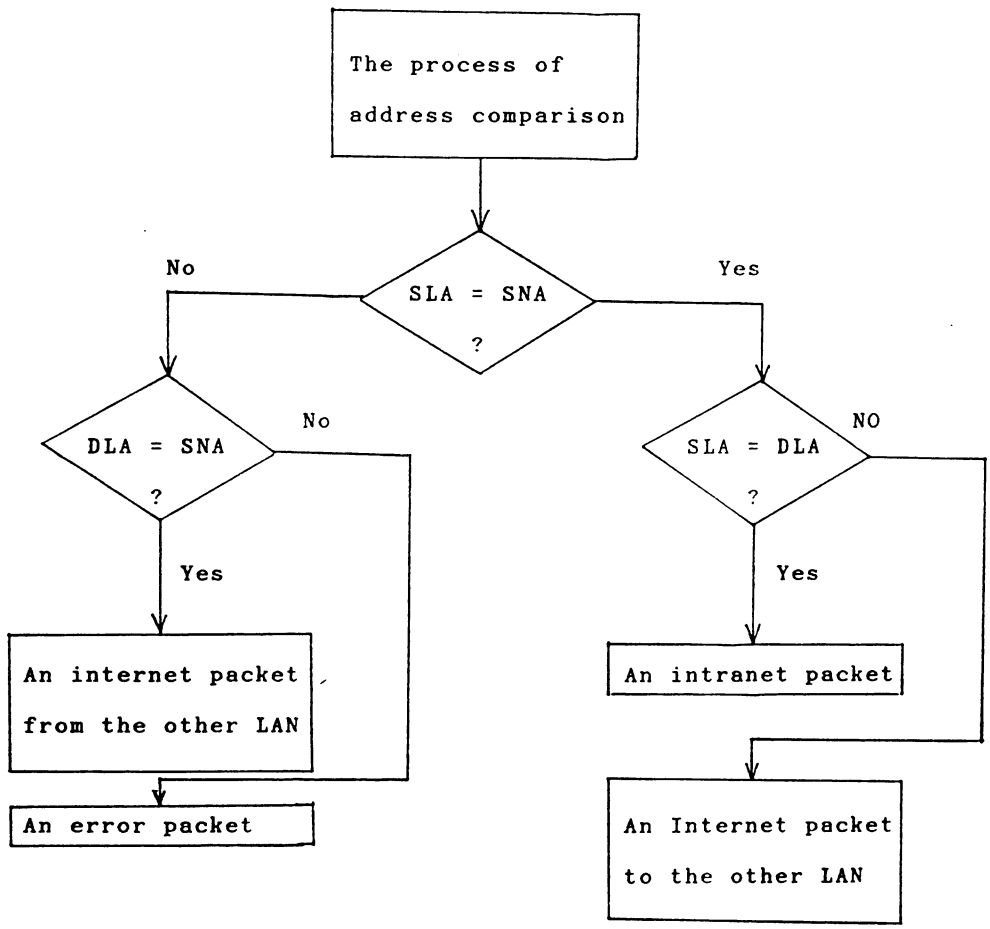
On the other hand, within a half-gateway, when a received packet is served to the Network layer, the following four kinds of results are possible, based on the addresses comparison as shown in fig. 5.10:

(1) If the SLA is the same as the SNA and DLA, the packet is considered as an intranet packet and destined to the half-gateway (end node) itself. Then the packet will be served to the higher layers.

(2) If the SLA is the same as the SNA but different from the DLA, the packet is considered as an internet packet which is intended for the other subnet. Then the packet will be served by the shared memory scheme to the Network layer of the other half-gateway in the other subnet.

(3) If the SLA is different from the SNA but the same as the SNA, the packet is considered an internet packet which is delivered by shared memory scheme from the other half-gateway. Then the address in the Destination Node Address field of the packet will be checked to determine the packet is destined to the half-gateway (end node) itself or other nodes in its subnet.

(4) In other conditions, the packet is considered as an error packet and will be discarded.



* DLA = Destination LAN Address.

* SLA = Source LAN Address.

* SNA = Subnet Address.

Fig. 5.10 Flow chart indicating the logical decisions to be made in routing a packet to currect destination. Case 2: the packet is from lower layers or the other half-gateway to the Network layer of a half-gateway.

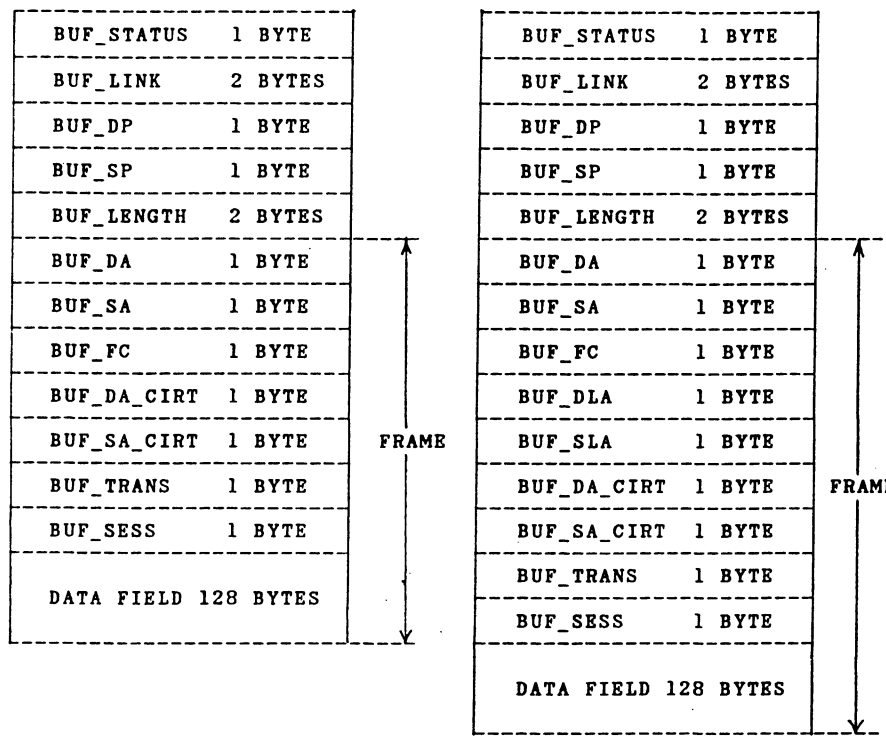
5.4.3 Frame Reformation and Mapping

To add the DLA, SLA and RDNA fields in a packet to the address of a destination of an internet packet, the frames in XLNET and TMS-IBM Ring have to be reformatted as shown in Fig.5.11 and Fig.5.12, respectively.

In addition, the frame reformation of an internet packet is always made in the half-gateway in the TMS-IBM Ring, no matter where the packet comes from (e.g. from TMS-IBM Ring to XLNET, or vice versa). For example, when an internet packet sent from one node in XLNET, to its destination in TMS-IBM Ring arrives in the half-gateway in XLNET, the packet is delivered through dual-port RAMs by shared memory scheme to the half-gateway in TMS-IBM Ring. At this time, the format of the packet is still that of XLNET as shown in Fig.5.11 to that of TMS-IBM Ring as shown in Fig.5.12. However, if the internet packet is delivered in the opposite direction i.e. from the TMS-IBM Ring to XLNET, the format of the packet is changed to that of XLNET in the gateway of TMS-IBM Ring before the packet is sent to the gateway in XLNET via the dual-port RAMs. In other words, when the internet packet arrives at the half-gateway in XLNET, it can be processed as the local packet without any modification.

In the procedure of the mapping, the following four aspects are worth mentioning:

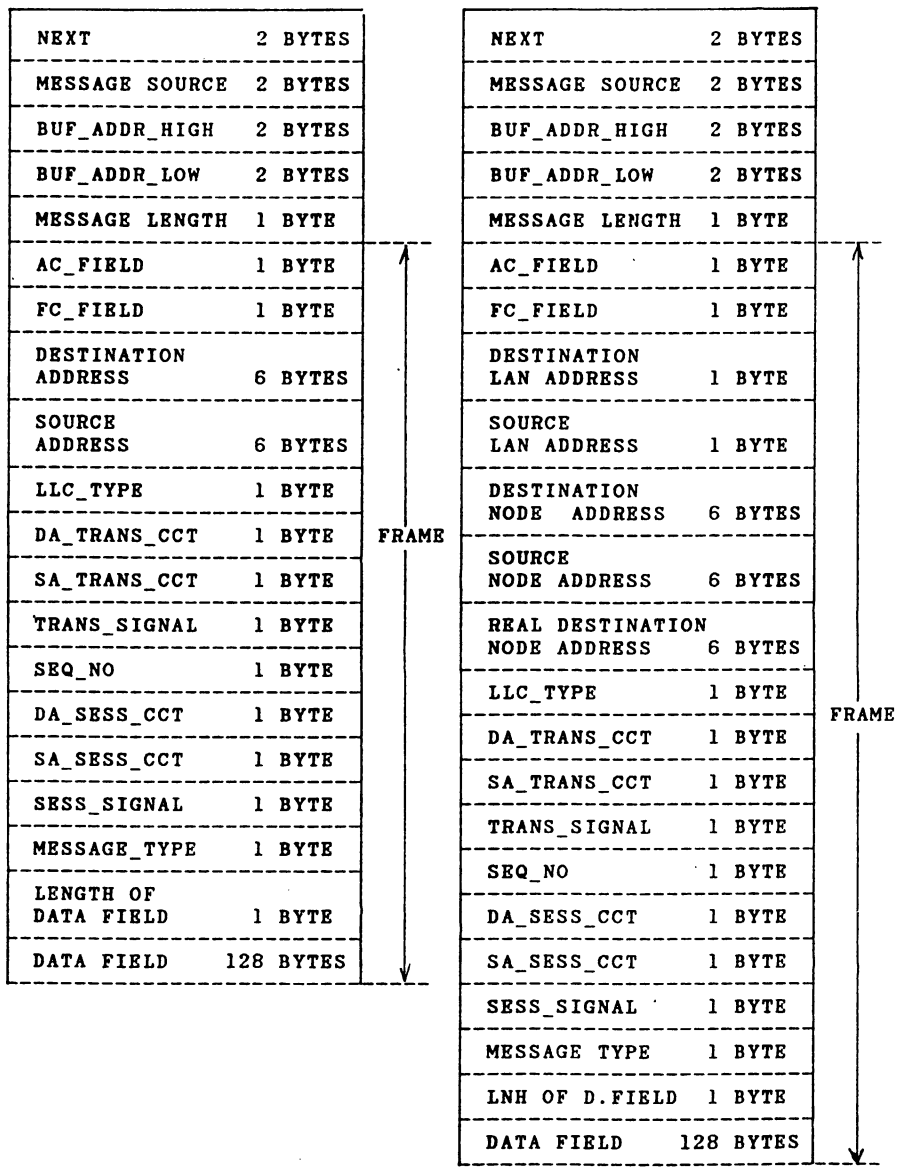
(1) Since both subnets have the same or similiar fields in their packet, most of the fields are easily mapped into each other, such as FC, address fields, transport and session virtual circuit numbers, information in data field, information type and length of message.



(A) An unmodified frame format of XLNET (135 bytes).

(B) A modified frame format of XLNET (137 bytes).

Fig. 5.11 The unmodified and modified frames of XLNET



(A) An unmodified frame format (152 bytes).

(B) A modified frame format (160 bytes).

Fig. 5.12 The unmodified and modified frames of TMS-IBM Ring.

(2) Since the sequence number field is included in the TMS-IBM Ring's packet but not in XLNET's one, a table is established in the gateway of TMS-IBM Ring to be an index for the internet packet transfer. Whenever an internet packet from TMS-IBM Ring arrives at the gateway, the gateway will save the sequence number in proper position of the table indexed by the transport virtual circuit number of the packet, and vice versa.

(3) Since the transmission of an internet packet is achieved by the shared memory scheme, the mapping of the information in the data field is conveniently and efficiently made by data transfer from the dual-port RAMs to the gateway's packet buffers.

(4) The mapping of the transport and session signals will be presented in the next sub-section, because both the subnets have different protocols in Transport and Session layers.

Since the maximum size of the data field in both packet formats are the same, and reformatting an internet packet and mapping the information in the field are made through the dual-port RAMs, the frame segmentation and reassembly are unnecessary.

5.4.4 The Protocol Translation

As mentioned in Chapter Four, only the Transport and Session layer protocols need be translated because of the advantage of using the shared memory scheme.

As Fig. 4.11 suggests, to splice level 4 (Transport layer) of station A (of XLNET) onto level 4 of station B (of TMS-IBM Ring), and vice versa, it is necessary to examine the set of services that an entity at

the level 4 of A would normally provide to one in level 5 (Session layer) of A, and compare that with the services that level 4 of B would normally provide to level 5 of B. Then the conversion must make the Transport layer of A think it is providing services to the Session layer of A, and must make the Transport layer of B think it is providing services to the Session layer of B.

Protocols in a level can be classified as interface protocols, peer protocols and control protocols. An entity at the layer N engages with the next higher layer (N+1) in the same node by a set of interface protocols, but also executes peer protocols with a peer layer in another node, and performs control protocol with a control entity within the same node. The interface protocols will not provide the right services if there are serious mismatches involving the other two protocols. In our study, although the protocols in the Transport layer of both subsets have differences as shown in Appendix B, even they are selected from the OSI Connection-oriented Transport protocol Class 4, there is no significant mismatch in the protocols. The case in the Session layer protocols of both subnets is the same as well, even though the protocols are different (see Appendix B).

Under the above circumstance, the peer protocol signals in an internet packet header must be mapped in the gateway to those of destination network. Table 5.2 lists the mapping of the transport and session protocol signals. For example, an internet packet from XLNET having the transport protocol signal, "call-req", in its packet header, will be mapped into the "t-open-ind" in the transport signal field in a reformatted packet header. In addition, the session protocol signal, "s-open-ind", is put into the packet header. Then the interface and control protocols of Transport and Session layers of TMS-IBM Ring's

node will perform the proper function, once the mapped packet is received in the Network layer of the same node.

Table 5.2 The mapping of peer protocol signals of Transport and Session layers between XLNET and TMS-IBM Token Ring

XLNET				TMS--IBM TOKEN RING			
Trans-signal	Code pattern (Hexadecimal)	Sess-signal	Code pattern (Hexadecimal)	Trans-signal	Code pattern (Hexadecimal)	Sess-signal	Code pattern (Hexadecimal)
call-req	0B	*	*	t-open-ind	02	S-open-ind	01
info-ind	0C	info-ind	0C	t-info-ind	0C	S-open-ind	0D
clr-req	0D	*	*	t-close-ind	0E	*	*
confirm	06	*	*	t-open-acc	03	*	*
call-ans	0A	*	*	t-open-con -ind	09	S-open-con -ind	0B
call-rej	09	*	*	t-open-rej -ind	0A	S-open-rej -ind	0C
t-confirm	16	*	*	lt-confirm	10	*	*

* The symbol of star means "Nil" which means the sess-signal is not included in the peer protocol signal packet.

CHAPTER SIX

PERFORMANCE ANALYSIS

6.1 Introduction

The interconnected network system has two rings, connected to each other through the gateway. The gateway comprises of two half-gateways. Each ring has its local nodes and a half-gateway. Here, these half-gateways do not only transfer the internet packets to/from XLNET from/to TMS-IBM Ring, respectively, but also handle the intranet packets in their own connecting rings.

The focus of this performance analysis is to determine the mean response time experienced by the gateway which processes an internet packet (voice or data) for transmission. This is defined as the mean time it takes to transmit an internet packet: from the moment of its arrival in the queue of the gateway to its departure from the gateway. The packet throughput of the gateway (the actual number of packets/unit of time that get through the gateway) is discussed here as well. Both analytical and experimental simulation models are developed.

Throughout this thesis, the Kendall notation, A/B/C, is used to describe the kind of queueing model used. The symbol A represents the arrival distribution, B represents the service distribution, and C denotes the number of servers used.

In the next section, the simulation model is described. The performance including the mean delay time and throughput of the gateway will then be evaluated in Section 6.3. The simulation results will be presented in Section 6.4.

6.2 Description of the Model

The queueing delays of the gateway, can be modeled by two queueing systems as shown in Fig. 6.1, where each server is responsible for one direction of an internet packet transmission. We assume that the packets arrive randomly, at an average rate of λ packets/unit of time. They queue up for service in the buffer as shown and are then served, following FCFS (first come first served) service discipline, at an average rate of μ packets/unit of time.

The arrivals to each of the queues in the two sub-networks, XLNET and TMS-IBM Ring, can be assumed to follow an independent Poisson process. This assumption however, can be very unrealistic for the queues of the gateway that receive packets from adjacent sub-networks, because this implies that the departures from these sub-networks also follow Poisson processes.

From Chiarawongse [44], we know, that under some circumstances, the Poisson assumption could be a good approximation for modeling the departure process for the sub-networks. Consider, for example, TMS-IBM Ring and XLNET, which would be modeled as single server polling systems, where the service times at all local nodes would be assumed to follow the same distribution. We can approximate the behaviour of these sub-networks by M/G/1 (Poisson arrivals, general service distribution, and a single server) queueing models. Thus the departure process from these sub-networks, which in turn constitutes the arrival process to the gateway, can be approximated by the interdeparture process of an M/G/1 queueing system.

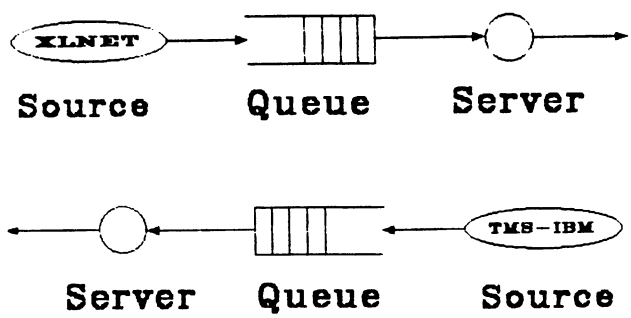


Fig. 6.1 The queueing model of the gateway between XLNET and TMS-IBM Ring

We assume that the internet packet, arrivals are Poisson distribution. It is further assumed that each arrival at the gateway is either a packet that contains user information (either voice or data) which has been generated for transmission over the network, or it is an Acknowledgement (ACK) of such a transmission. Under these assumptions, the arriving packets have a bimodal distribution with 50% of the packet being user data packets and the remaining 50% being accounted for by the ACK packets.

The packet service time is related to the length of the packet. Here, the lengths of all packets are assumed to be fixed (137 bytes for XLNET, 160 bytes for TMS-IBM Ring). With this assumptions, all packets from the same source have the same service time. Thus we end up with an M/D/1 queue.

The gateway provides a limited buffer of 2K-bytes for storing the internet packets (1K-bytes for each direction of transmission).

6.3 Performance Evaluation

Each station in the network can generate internet packets for transmission through the gateway over the network. By making the assumption that the arrival process to the model of gateway is Poisson, it is possible to use analytical methods to evaluate the performance of the model. The analytical results are presented here. The result is, of necessity, approximate but the usefulness of the model is examined.

6.3.1 Response Time

The time delay in the gateway has three parts:

1. The queueing time in a half-gateway E(Q).

2. Protocol conversion time E(p).

3. User data transfer time E(t).

Therefore, the total time delay in the model of the gateway is equal to

$$E = E(Q) + E(p) + E(t)$$

The objective of the following analysis will be to determine each of the delay components.

6.3.1.1 The Delay in the Queue in a Half-gateway

Since the offered traffic in the internet system includes voice and data, it is necessary to queue these packets separately, ready for service. The gateway server gives service to the two queues by interruption. Furthermore, since the voice packets are delay sensitive, the voice is given non-preemptive priority of service over data.

Throughout this thesis, we designate voice packets by the label 1 and data packets by label 2. The symbols relating to higher-priority class carry a suffix 1. The arrival rate and average service rate for the two classes of packets are λ_1, λ_2, μ_1 and μ_2, respectively. The combined traffic intensity is $\rho = \rho_1 + \rho_2$, where $\rho_1 = \lambda_1/\mu_1$ and $\rho_2 = \lambda_2/\mu_2$.

We now discuss the queueing time for the two classes of packets, assuming non-preemptive and FIFO service. Consider class 2 (data packet). Let a packet of this class arrive at an arbitrary time t_0 . Its random queueing time Q2, measured from its arrival time until it enters service, is due to contributions from three sources:

1. It must wait a random amount of time Tc until the packet currently in service completes service.

2. It must wait a random amount of time Tk time until all packet of priority 1 or priority 2, already queued at the arrival time t_o, complete service. Therefore,

$$Tk = T1 + T2$$

Where T1 and T2 are the random amounts of the waiting time for all packets of priority 1 and 2, from t till such time the gateway server completes their services, respectively.

3. It must wait a random time T1' to service packets of priority 1 arriving during the wait time Q2.

Putting the above terms together, we write

$$Q2 = Tc + Tk + T1'$$
$$= Tc + T1 + T2 + T1'$$

While, the queueing time Q1 for the class 1 (voice) packets is

$$Q1 = Tc + T1$$

Taking expectations term by term, the average waiting time E(Q2) of priority 2 and E(Q1) of priority 1 are obviously given by

$$E(Q2) = E(Tc) + E(Tk) + E(T1') \quad \text{----(1)}$$
$$= E(Tc) + E(T1) + E(T2) + E(T1')$$

and

$$E(Q1) = E(Tc) + E(T1) \quad \text{----(2)}$$

respectively.

To find each component in Eq. (1), note first that E(T2) is due to an average number E(N2) packets of class 2 waiting in the gateway. It requires $1/\mu 2$ units of service on the average, so that we immediately have

$$E(T2) = E(N2) / \mu 2 \quad \text{----(3)}$$

Similarly,

$$E(T1) = E(N1) / \mu 1 \qquad \text{----(4)}$$

But by Little's formula, we also have E(N2) related to the average wait time E(Q2). Specifically,

$$E(N2) = \lambda 2 * E(Q2) \qquad \text{----(5)}$$

Similarly,

$$E(N1) = \lambda 1 * E(Q1) \qquad \text{----(6)}$$

Combining Eqs. (3) and (5), we immediately have

$$E(T2) = \rho 2 * E(Q2), \rho 2 = \lambda 2 / \mu 2 \qquad \text{----(7)}$$

Similarly,

$$E(T1) = \rho 1 * E(Q1), \rho 1 = \lambda 1 / \mu 1 \qquad \text{----(8)}$$

Now consider the term E(T1') in Eq. (1). This is due, on the average, to E(N1') packets of class 1 arriving during the interval E(Q2). Since the arrival rate is $\lambda 1$ and the service rate is $\mu 1$, we have

$$E(T1') = \lambda 1 * E(Q2) / \mu 1 = \rho 1 * E(Q2) \qquad \text{----(9)}$$

Finally, the term E(Tc) in Eq. (1) is the residual service time of a packet in service. From Pollaczek-Khinchine formulas and [45], we find that the average wait time of an M/G/1 queue is

$$E(Tc) = \lambda * E(\tau^2) / 2 \qquad \text{----(10)}$$

Where, the second moment of the composite stream is given by the weighted sum of the second moments:

$$E(\tau^2) = (\lambda 1 / \lambda) * E(\tau_1^2) + (\lambda 2 / \lambda) * E(\tau_2^2)$$

Using Eqs. (7), (8), and (9) in Eqs. (1) and (2) solving for the waiting time of each class recursively. From [2], we have

$$E(Q2) = E(Tc) / [(1-\rho_1) * (1-\rho)]$$

$$\rho = \rho_1 + \rho_2$$

and

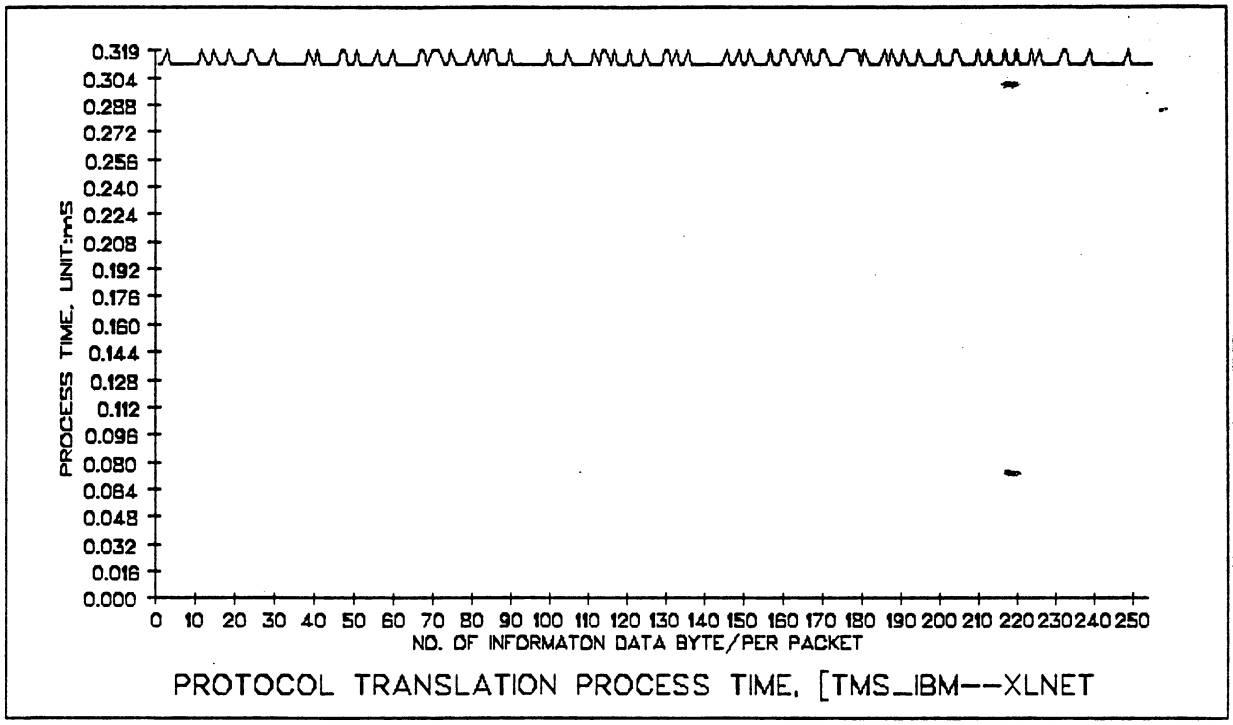
$$E(Q1) = E(Tc) + (1-\rho_1)$$

6.3.1.2 The Delay in the Protocol Conversion

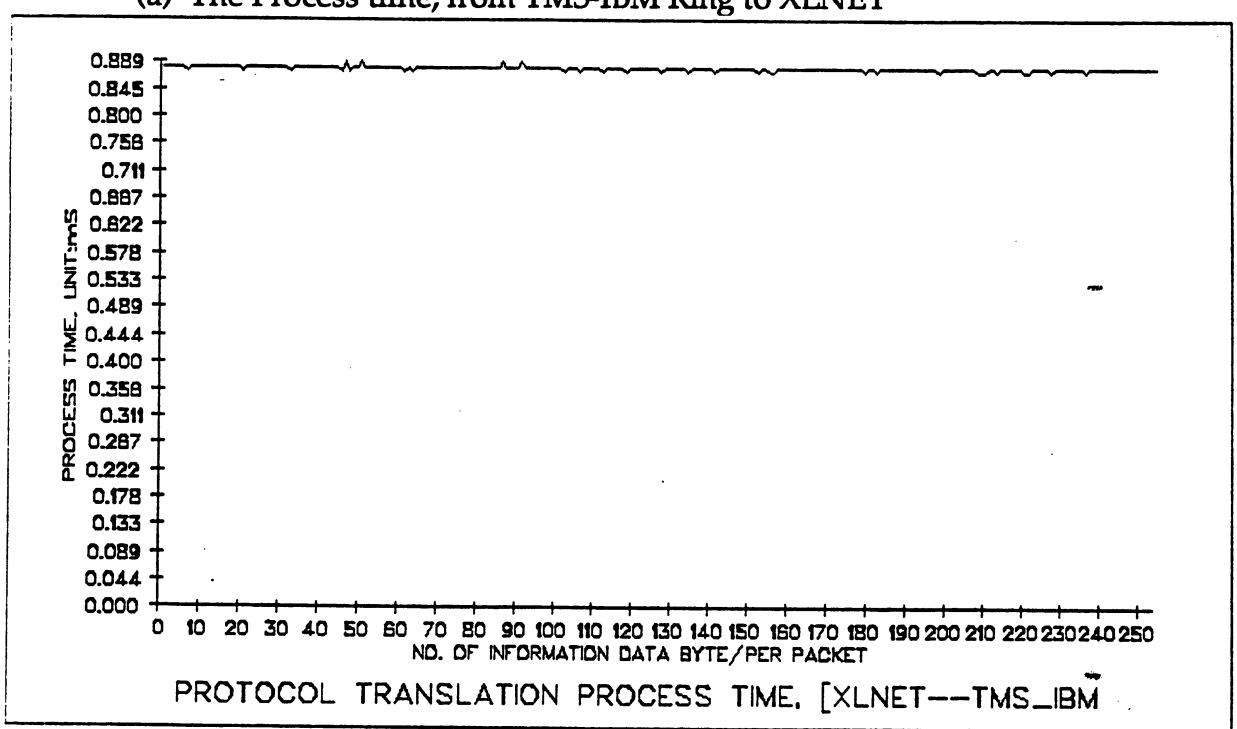
In Chapter Four, it was explained that the packets are received by a gateway at the Network Layer. Therefore, the layers involved in the protocol conversion are just the Transport Layer and Session Layer. Under this circumstance, the time spent on the protocol conversion is equal to the time required for mapping the peer protocol signals of Transport and Session Layers between XLNET and TMS-IBM Ring. The table for the layer mapping is shown in Table 5.2.

The key factor of the time delay due to protocol translation is proportional to the length of the protocol information. From Fig. 5.11 (b) and 5.12 (b), we know that the lengths of the protocol data of the packets of XLNET and TMS-IBM Ring are 9 bytes and 32 bytes, respectively. From Fig. 6.2, it is observed from experiments that the protocol translation process time is almost constant for each direction, e.g. the time for the protocol translation from XLNET to TMS-IBM Ring takes about 0.880 msec, and that from TMS-IBM Ring to XLNET takes about 0.311 msec. These delays include the data access time from the buffer of the gateway.

Because the length (32 bytes) of the protocol data of the TMS-IBM Ring's packet is longer than that (9 bytes) of the XLNET's packet, more time needs to be spent on the protocol translation from XLNET to TMS-IBM Ring.



(a) The Process time, from TMS-IBM Ring to XLNET



(b) The process time, from XLNET to TMS-IBM Ring

Fig. 6.2 The protocol translation process time (Experimental results)

6.3.1.3 The Delay in the User Data Transfer

As mentioned previously, by means of the shared memory scheme, the key factor to determine the time delay for the user data transfer is the length of the user data information.

By using the loop algorithm of the Turbo Pascal Ver. 3, the process of the information transfer is made byte by byte. The size of the loop is chosen from the field of Message Length in the TMS-IBM Ring's packet or the field of the Buf-length in the XLNET's packet, depending on the direction of the transfer.

From Fig. 6.3, it is observed that the delay rises almost linearly with the length of the user information. Similarly, the delay includes the data access time from the buffer of the gateway.

Here, to match the assumption of the M/D/1 queueing model, in our simulations presented later, the length of an internet packet is fixed. And from Figs. 5.11 (b) and 5.12 (b), we know that the size of the data fields of packets of XLNET and TMS-IBM Ring are identical, 128 bytes. Therefore, from Fig. 6.3, the time spent on the 128-byte long user data transfer is about 0.650 msec.

6.3.2 Throughput

The queueing gateway can be considered as a model of store-and-forward system. A characteristic of this type of system is that, as the offered load is increased from zero upwards, the gateway throughput increases to a maximum and then turns down and decreases sharply to a very low value (possibly zero).

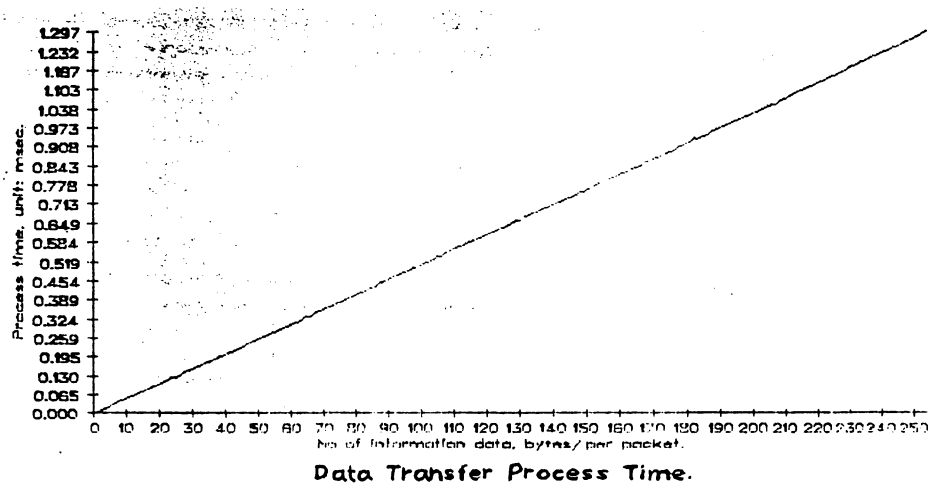


Fig. 6.3 The experimental data transfer time

Physically, when the system is congested, some processes may be blocked, and data may be lost or held back due to a lack of buffer space. In this case, the degradation in throughput is often caused by the deadlock (zero throughput). Hence, a control mechanism is needed to guard against it. Here, an approach of the input-buffer limiting is applied. If the number of packets already presented at the input queue exceed the capacity, N, of the provided input buffers, the newly arriving packets are blocked. With the probability of blocking given by Pb, the net arrival rate is then $\lambda * (1-Pb)$. But, this must be the same as the throughput γ. We thus have

$$\gamma = \lambda * (1-Pb)$$

From [39], we know that the blocking probability is given by

$$Pb = (1 - \rho) * \rho^N / (1 - \rho^{N+1})$$

for the finite M/M/1 queue.

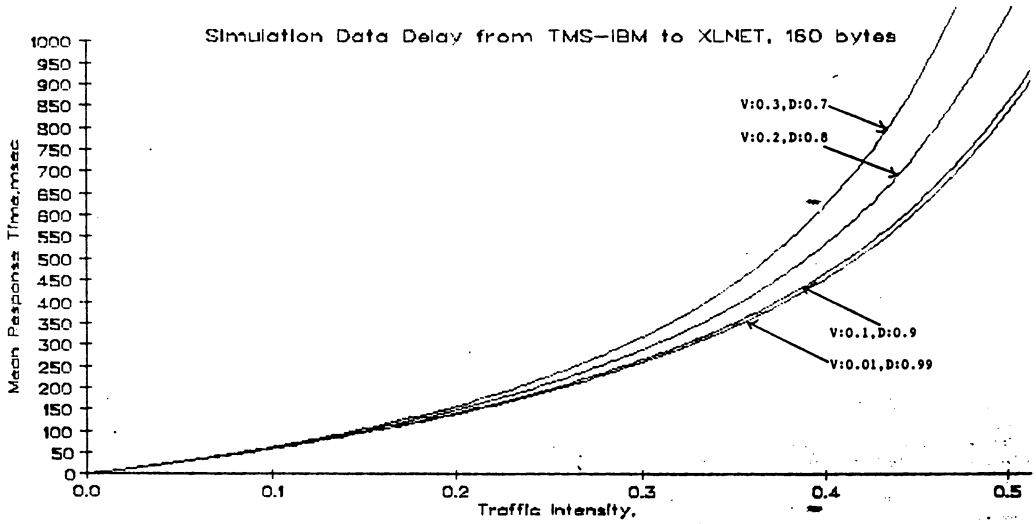
6.4 Simulation Results

Two sets of experiments were designed to test the gateway performance under a variety of conditions. The objectives and results of each experiment are summarized in the following sub-sections.

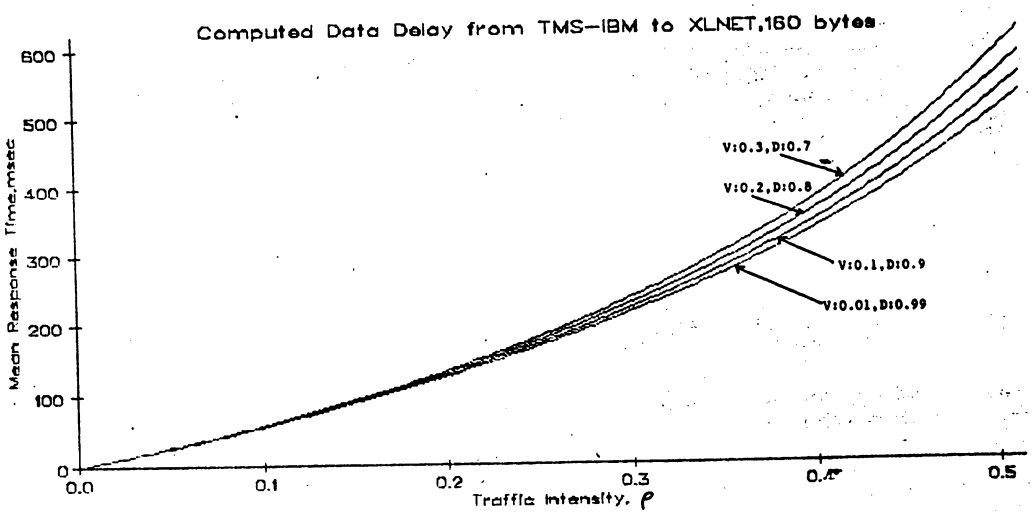
6.4.1 Packet Arrival Rate

We observed the gateway performance regarding the throughput and mean response time as a function of the packet arrival rate.

The laboratory simulation and computed results based on M/D/1 model for the mean response time to data and voice packet from TMS-IBM Ring to XLNET and form XLNET to TMS-IBM Ring are shown in Figs. 6.4, 6.5, 6.6 and 6.7, respectively.

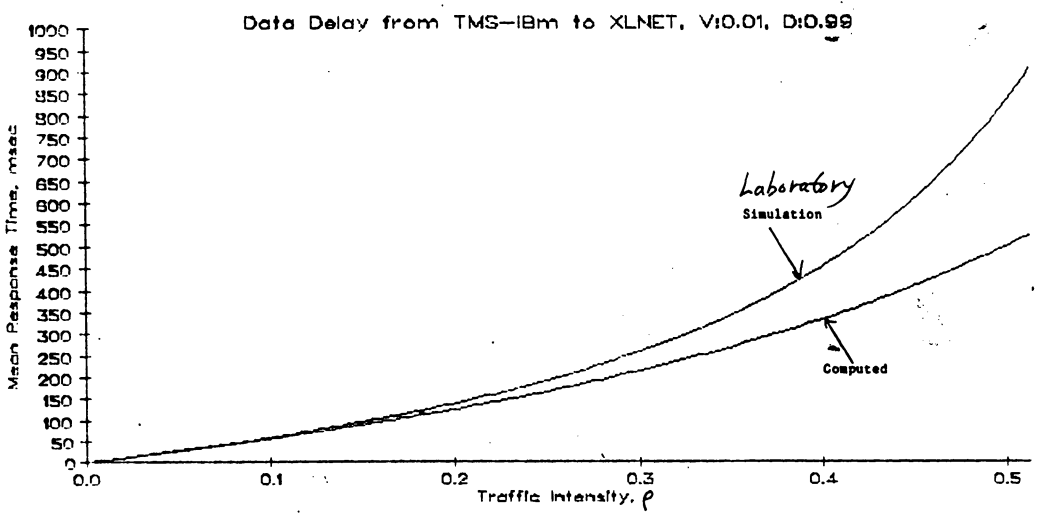


(a) Laboratory simulation delay time

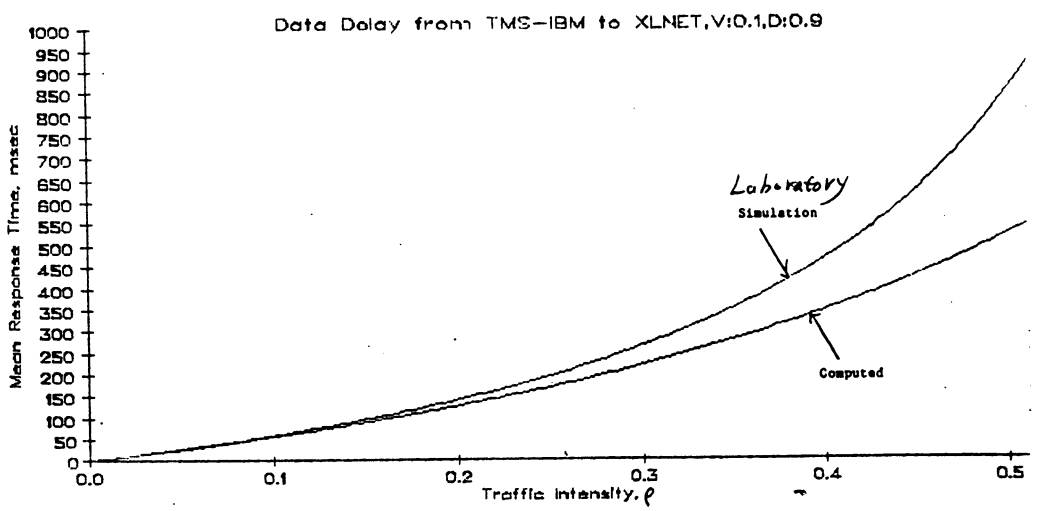


(b) Computed delay time

Fig. 6.4 The data delay from TMS-IBM Ring to XLNET through the gateway

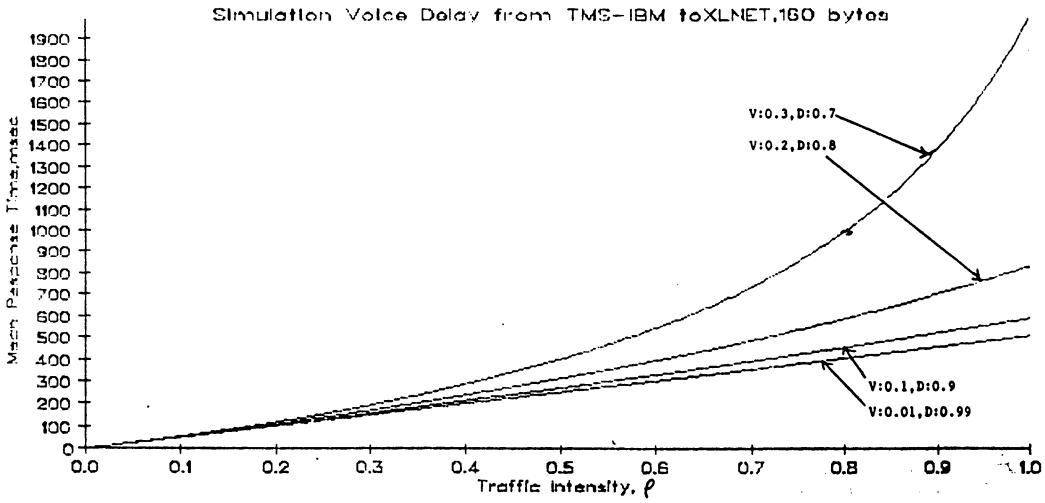


(c) Delay time (1% traffic to voice, 99% traffic to data)

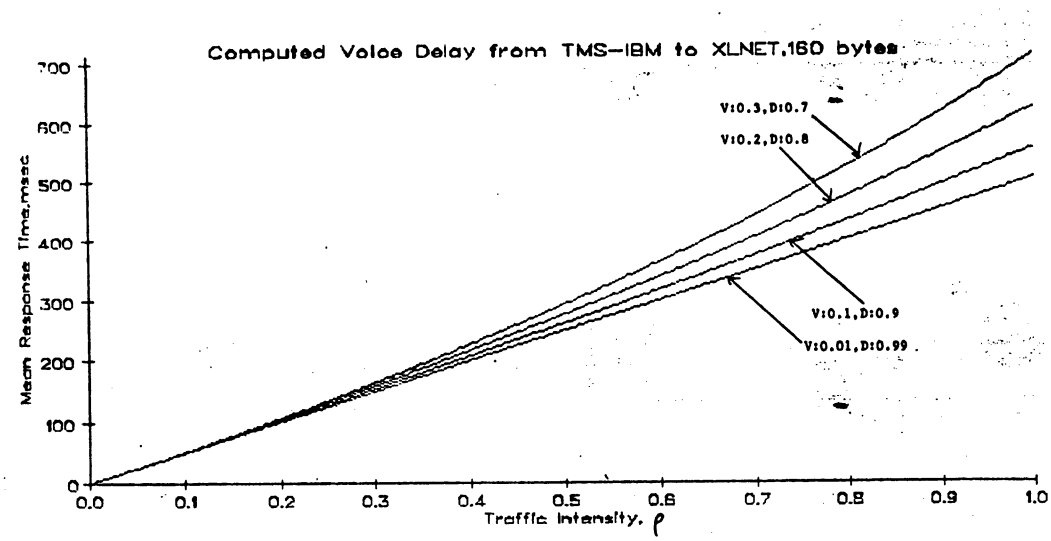


(d) Delay time (10% traffic to voice, 90% traffic to data)

Fig. 6.4 The data delay from TMS-IBM Ring to XLNET through the gateway

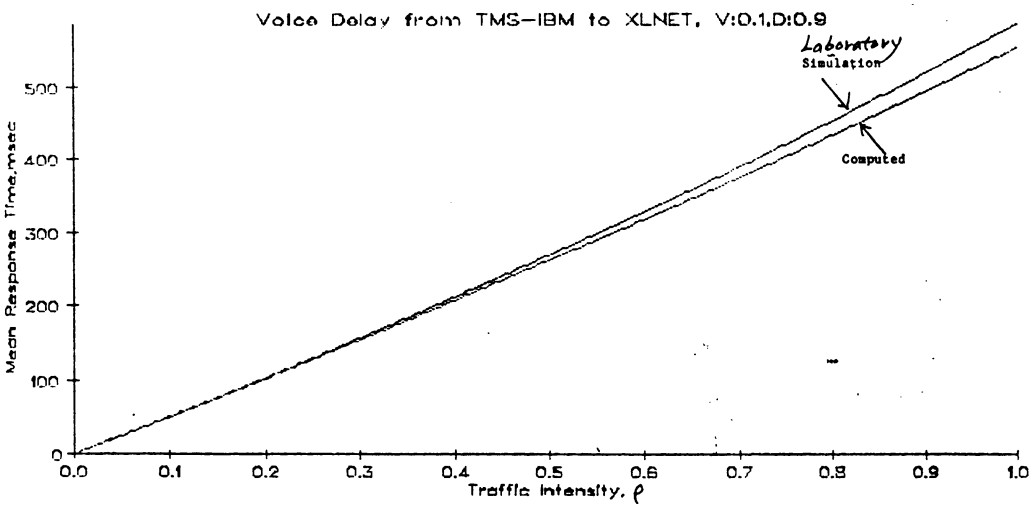


(a) Laboratory simulation delay time

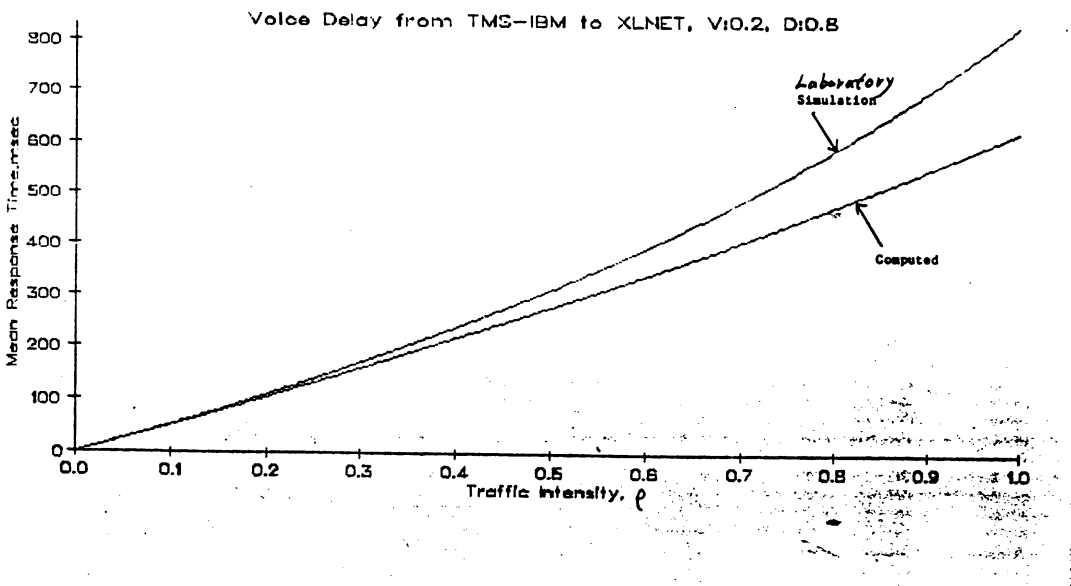


(b) Computed delay time

Fig. 6.5 The voice delay from TMS-IBM Ring to XLNET through the gateway

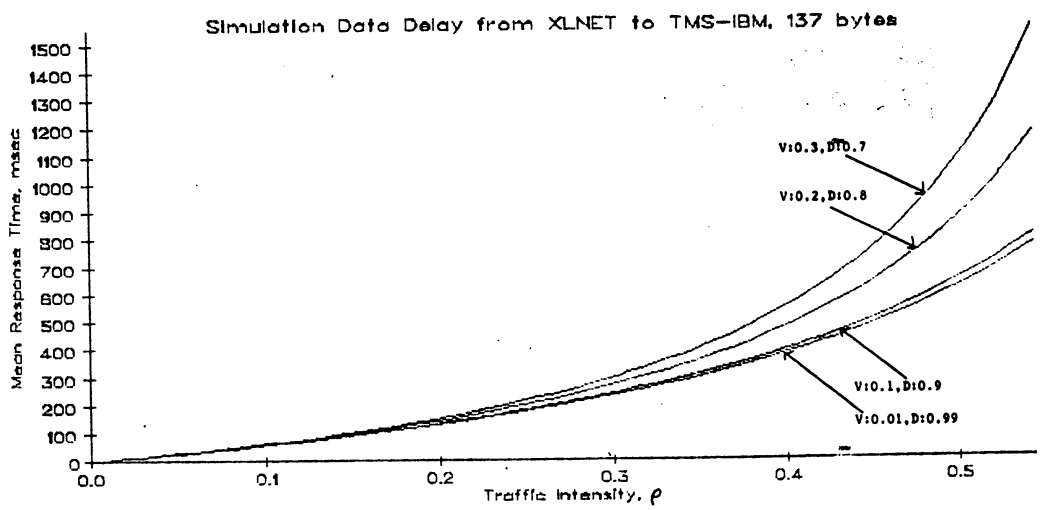


(c) Delay time (10% traffic to voice, 90% traffic to data).

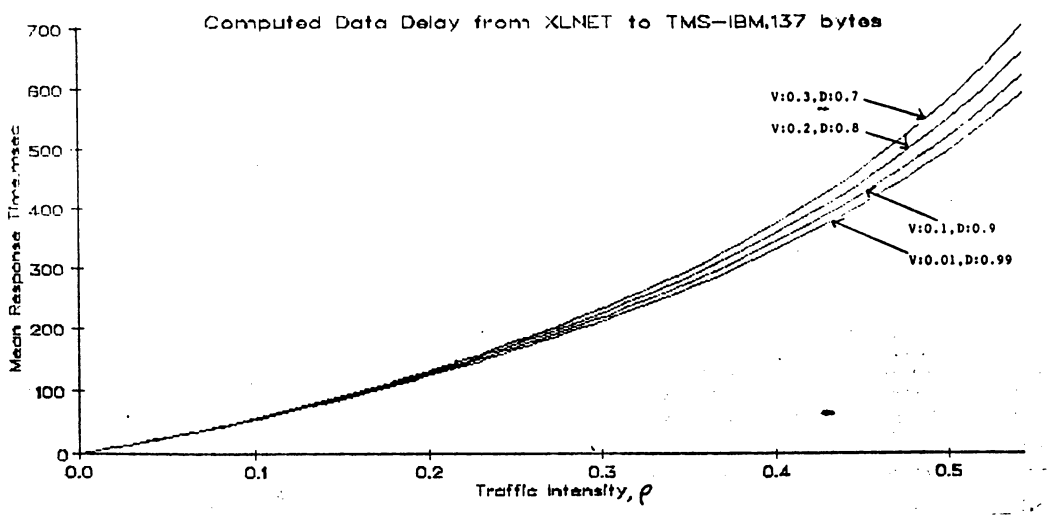


(d) Delay time (20% traffic to voice, 80% traffic to data).

Fig. 6.5 The voice delay from TMS-IBM Ring to XLNET through the gateway

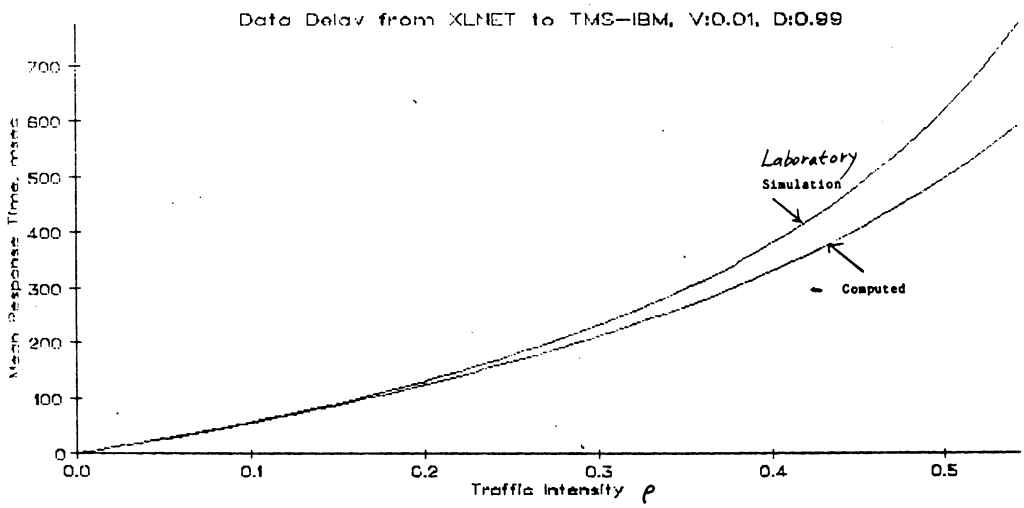


(a) Laboratory simulation delay time

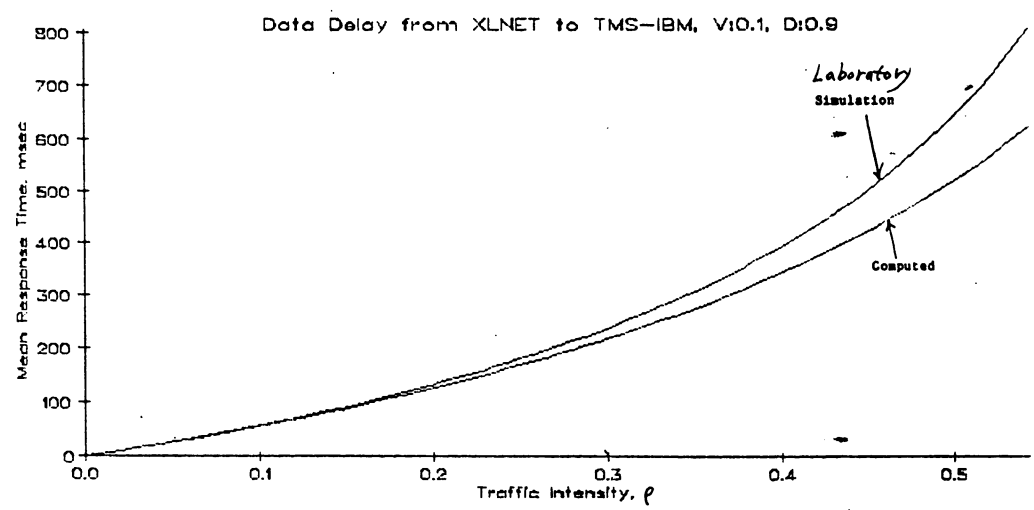


(b) Computed delay time

Fig. 6.6 The data delay from XLNET to TMS-IBM Ring through the gateway

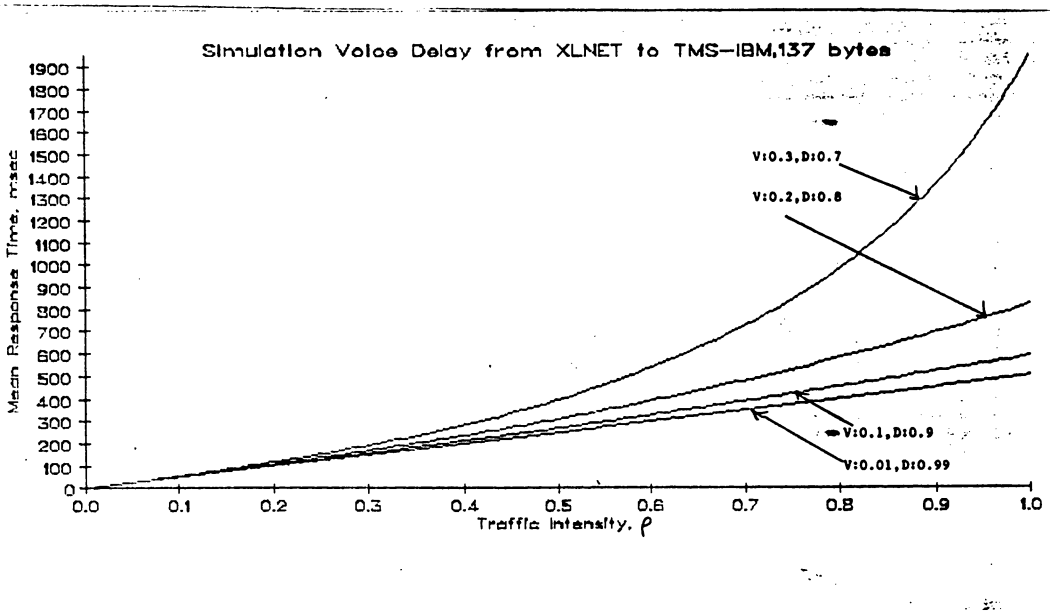


(c) Delay time (1% traffic to voice, 99% traffic to data)

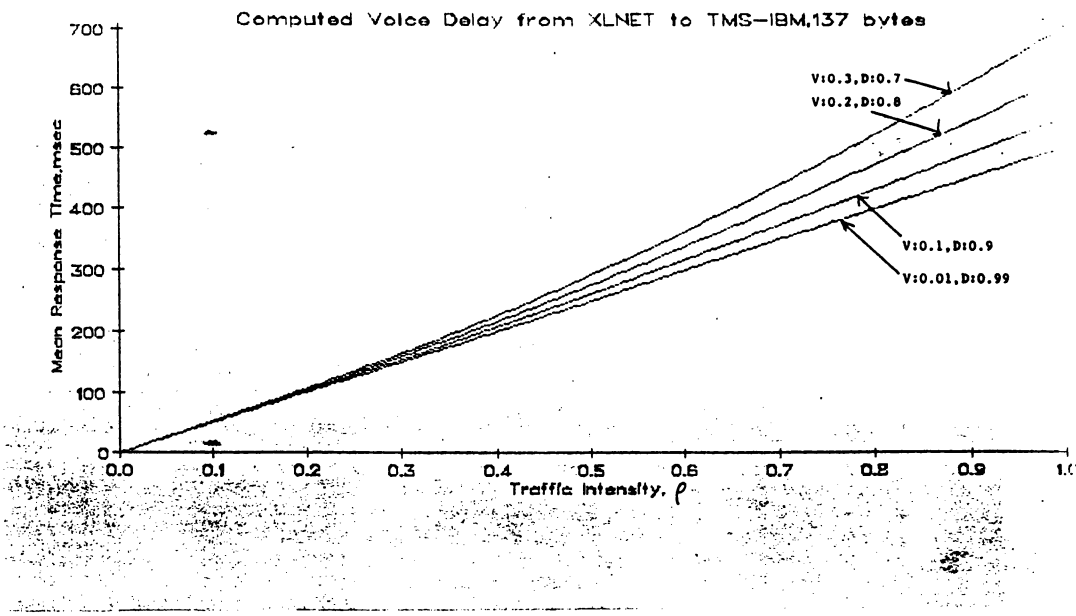


(d) Delay time (10% traffic to voice, 90% traffic to data)

Fig. 6.6 The data delay from XLNET to TMS-IBM Ring through the gateway

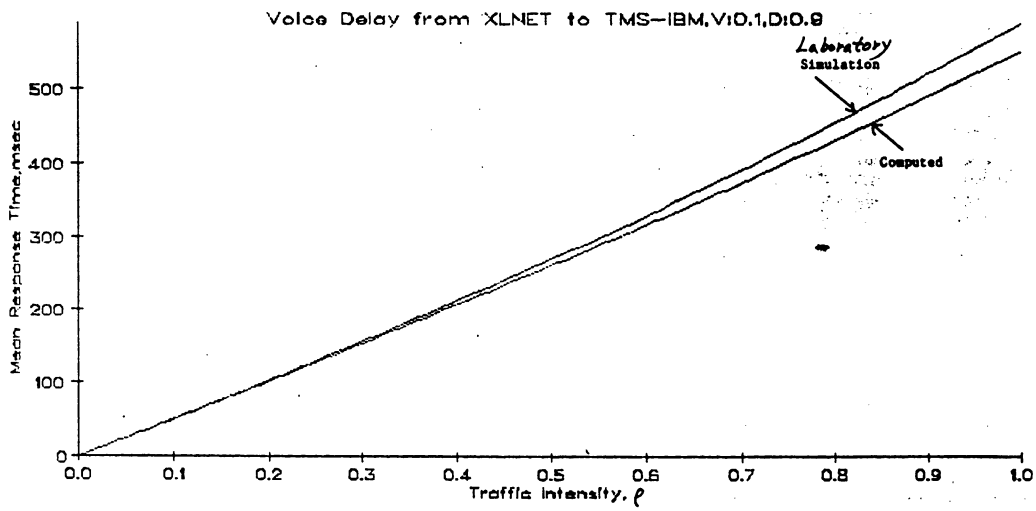


(a) Laboratory simulation delay time

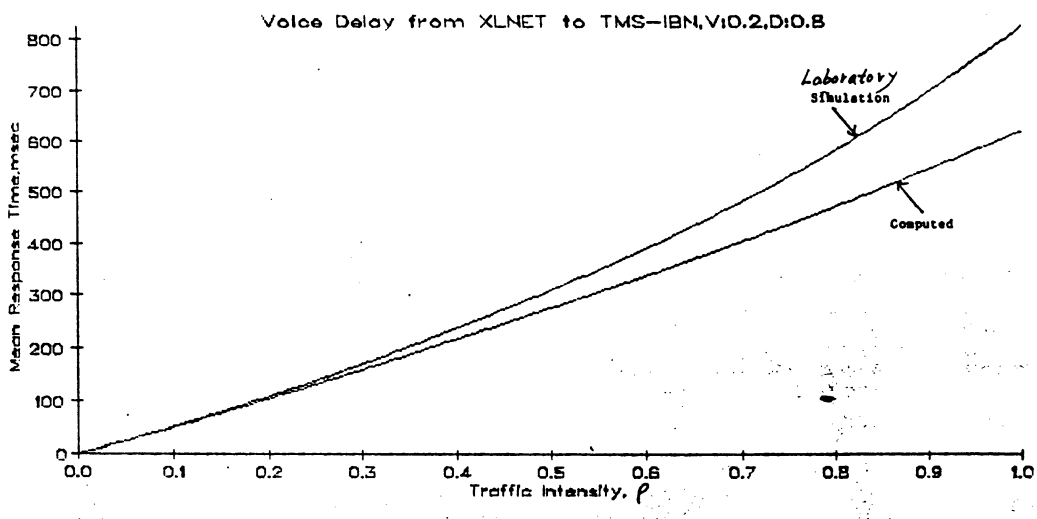


(b) computed delay time

Fig. 6.7 The voice delay from XLNET to TMS-IBM Ring through the gateway



(c) Delay time (10% traffic to voice, 90% traffic to data)



(d) Delay time (20% traffic to voice, 80% traffic to data)

Fig. 6.7 The voice delay from XLNET to TMS-IBM Ring through the gateway

The results, regarding the different percent of the total traffic due to data and voice packets (such as 99% to data, 1% to voice, and 90% to data, 10% to voice), are shown in these figures as well.

From these figures we note that, under high traffic load, the delay of voice packets is obviously shorter than that of data packets. It is a consequence of the non-preemptive priority.

From Fig. 6.4 (c), it is observed that the delay time for a data packet (128 data bytes) from TMS-IBM Ring to XLNET arriving at the gateway, under the conditions of 0.3 traffic intensity, 1% to voice and 99% to data, takes 215.752 msec by the calculated simulation. However, it takes 260.483 msec from experimental results. The time includes 0.311 msec for protocol translation, 0.650 msec for data transfer and 259.522 msec for queueing in the buffers of the gateway. While, from Fig. 6.6 (c), it is observed that the delay for the packet from the opposite direction, under the same conditions as above, takes 214.493 msec by the calculated simulation. However, it takes 234.518 msec from the experimental results. The time includes 0.880 msec for protocol translation, 0.650 msec for data transfer and 232.988 msec for the queueing in the buffers of the gateway. From the above observations, we find the agreement between the laboratory and theoretical results is reasonable, and the discrepencies between them will be explained later.

Regarding the throughput, the gateway is provided with finite input buffers (2K-bytes for storing the internet packets, 1K-bytes for each direction of transmission) making congestion possible. Figs. 6.8 and 6.9 show the throughput as a function of traffic intensity. The simulation results are in close agreement with calculated results using the M/D/1 model for traffic intensity of less than 0.3. Although the qualitative

agreement is good, the results show that the calculated prediction based on M/D/1 model are too optimistic for traffic intensities in excess of 0.3. For out particular design, the gateway throughput drops to zero for traffic intensity in excess of 0.5.

Comparing Figs. 6.8 and 6.9, we find that the maximum throughput from TMS-IBM Ring to XLNET is about twice as large as the throughput from the opposite direction. The key factor to determine the throughput is the service rate provided by the gateway server. The higher the service rate, the larger the throughput. From the model as shown in Fig. 6.1, there are two servers. They are provided by the IBM PC/XT (4 M bits/sec) machine and the XLNET model hardware (1.5 Mbits/sec). Each is responsible for each direction of transmission. The ratio of the two service rates is about 2.6, which proves the result of the comparison mentioned above.

The discrepancies between the above simulation and computed results can be accounted for as follows:

1. Because each half-gateway also plays the functions as a local node in its ring, the overheads for cycle and ring management result in the extra delay.

2. In this particular resolution, only two nodes in XLNET and two nodes in TMS-IBM Ring are available. The simulation method used here [46] necessitated to designate one node as a half-gateway, and the others simulate a number of nodes to generate the required traffic to the gateway. In order to represent a large number of nodes, the simulation node must be able to generate the equivalent amount of traffic of a number of nodes. In this case, the simulation

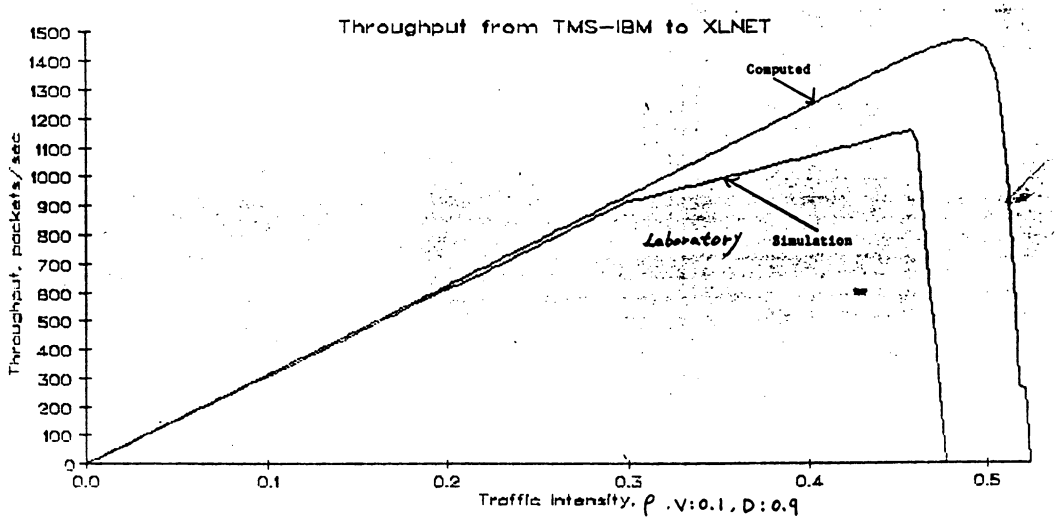


Fig. 6.8 The throughput from TMS-IBM Ring to XLNET

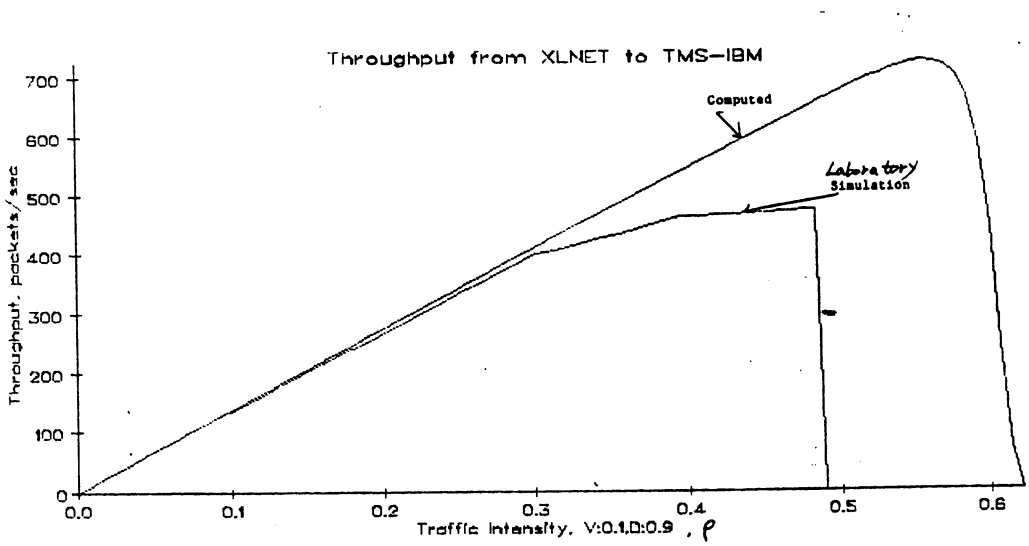


Fig. 6.9 The throughput from XLNET to TMS-IBM Ring

node and the half-gateway are then burdened with software delay in sending and processing packets, respectively.

3. The mathematical model uses M/D/1 formalism which is only an approximation.

6.4.2 Packet Size Distribution

Using the M/D/1 queueing model of the gateway, the packet size is a key factor to determine the packet service time. The larger the packets, the longer the service time. This aspect was studied in some detail and typical results are shown in Fig. 6.10. The results have the covert trend but quantitative agreement is limited on account of the simplifying assumptions embedded in the theoretical formulas.

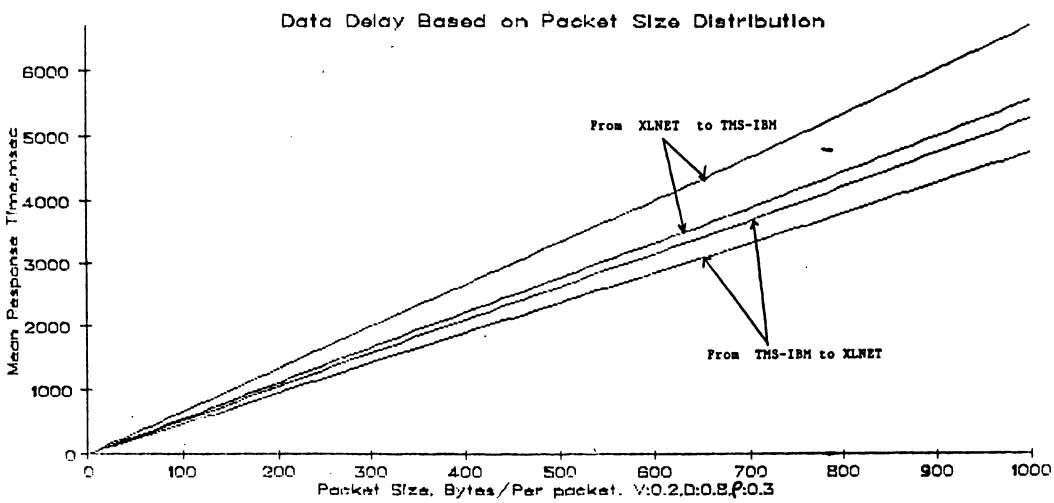


Fig. 6.10 The experimental data delay based on packet size distribution (traffic intensity : 0.3, 20% to voice, 80% to data)

CHAPTER SEVEN

LIMITATIONS OF THE PRESENT DESIGN AND SUGGESTED IMPROVEMENTS

7.1 Introduction

The performance of the laboratory gateway was limited on account of lack of better laboratory facilities. Each ring TMS-IBM and XLNET was equipped with two nodes only, and a PC was used as a processor. In addition the software was written for convenience in Pascal and has not been optimized in any way.

7.2. Hardware Aspect

Since the two nodes B and C shown in Fig. 5.1 are designed to perform two functions: (1) each node acts as a local node on its own LAN, and (2) each node acts as a half-gateway in the internet system with the resources of the gateway shared. Under this condition, extra delay may result from the hardware shared to handle the two directional traffic and provide the overhead for the local ring management. This is one of the main reason to cause the performance degradation. It would be better to use dedicated hardware for the gateway. For reasons of economy, in our implementation, a PC (4 MHz) was used as the processor. A faster processor would upgrade the performance.

7.3 Software Aspect

The software designed to perform the protocol conversion and user data transfer is written in Turbo Pascal 3 whose executing speed is somewhat limited. It is suggested that the software program be rewritten in more suitable language such as C or assembler. Moreover, in our

implementation, the loop algorithm is employed in the process of the data transfer, and the process is made byte by byte. Then the size of the loop, which depends on the length of the user data, is the key factor to determine the time delay for the process. The larger the size of the loop, the longer the process time. Under this condition, if the length of the user data in a packet is fixed, it is suggested that the process be made block instead of a byte. In this case, the transfer delay will be significantly reduced due to a smaller number of loops.

CHAPTER EIGHT

CONCLUSIONS

The increased use of networks has resulted in the need for gateways which enable the interconnection of heterogeneous networks. In attempting to implement the gateway, a detailed study of the architecture and the characteristics of the two dissimilar LANs has been undertaken.

There are significant differences on the transport and session layers of the two LANs. Software has been written to enable peer-to-peer communication between the two LANs. It was, therefore, necessary to have such a protocol translator. Neither a bridge nor X.75 protocol would be adequate. The software consists of addressing, routing, frame reformation and mapping, and protocol translation.

Hardware for the gateway was designed. For our particular case, the structure of the gateway was not only designed to be able to perform the required functions of a gateway in the internet system, but also acted as local node in a subnet. This consists of buffers and controls of incoming and outgoing packets.

Moreover, due to the parallel transmission and fast memory chip access time contributed by the shared memory scheme, the delay time of the internet packet transmission through the gateway would be significantly decreased. Concurrently, the cost of the physical equipment for the interconnection was saved as well.

A simplified model was used to analyze the performance of the gateway. In the simulation, we found the experimental results are in close agreement with calculated ones for traffic intensity of less than 0.3.

Furthermore, the performance of the gateway was limited in throughput, because of processor and other laboratory constraints, but was adequate to carry out a study of the gateway performance in presence of simulated voice and data traffic.

Finally, it is suggested that further research be conducted to provide advanced gateway between TMS-IBM Ring and XLNET with adequate functions. Therefore, it is able to integrate image, video and other different traffics through the gateway onto the interconnecting network system.

Appendix A TMS380 Adapter Chipset

The following information is derived from [20].

Appendix A.1 The Diagram of the TMS38010 Communication Processor

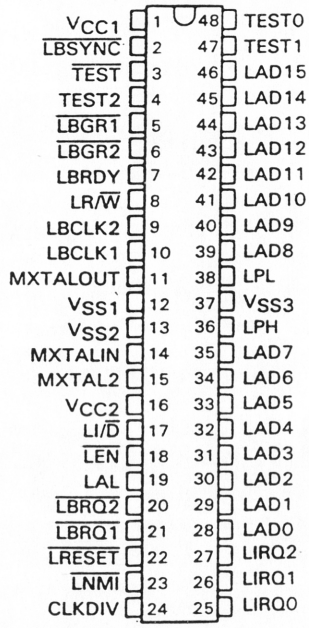
ADVANCE INFORMATION

TMS38010 COMMUNICATIONS PROCESSOR

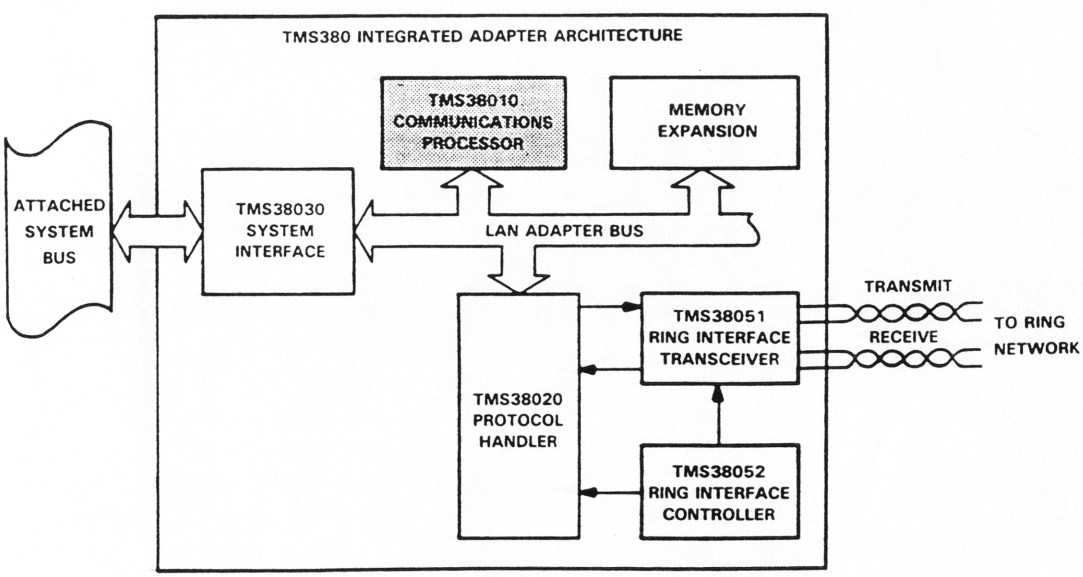
SEPTEMBER 1985

- High-Performance 16-Bit CPU for Processing Communications Protocols
  - —333-ns Machine and Bus Cycle Time
  - —Single Cycle Pipelined Bus Arbitration
  - —9 Interrupt Priority Levels
  - —8-Bit General Purpose Timer
- On-Chip 2.75K-Byte RAM for Buffering Network Data
  - —1408 x 18-Bit Organization
  - —Byte Parity Protection
  - —6 Megabyte per Second Data Transfer Rate
- Expandable Program and Data Memory Space up to 256K Bytes
- Built-in Real Time Error Detection
- Test Pins for Hi-Z, Module-in-Place Testing
- Single 5-V Supply
- 24-MHz Crystal Oscillator or Crystal Input (Internal Oscillator Option)
- Low-Power Scaled-NMOS Technology

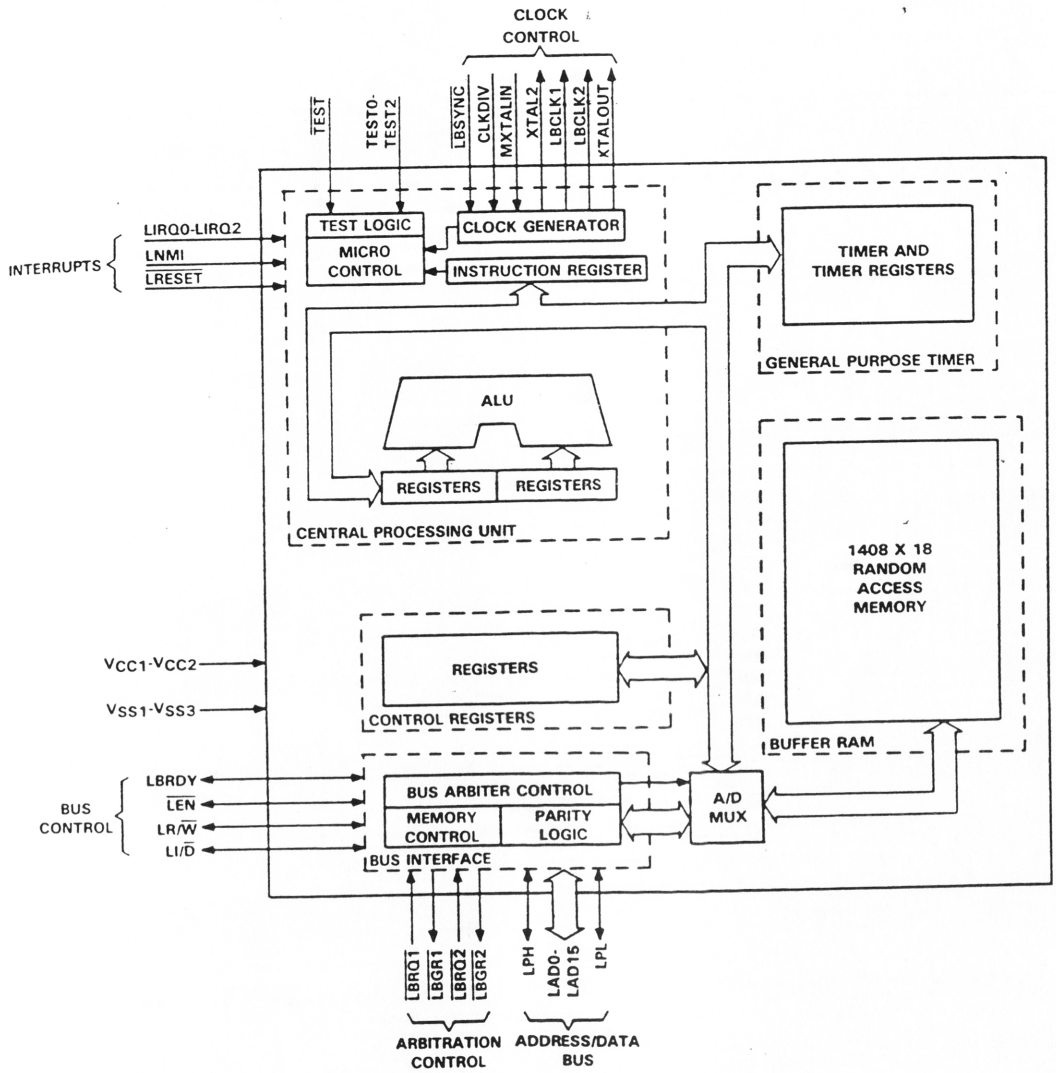
JD PACKAGE
(TOP VIEW)



token ring LAN application diagram



functional block diagram



Details of the functions are given in [20].

Appendix A.2 The Diagram of the TMS38020 Protocol Handler

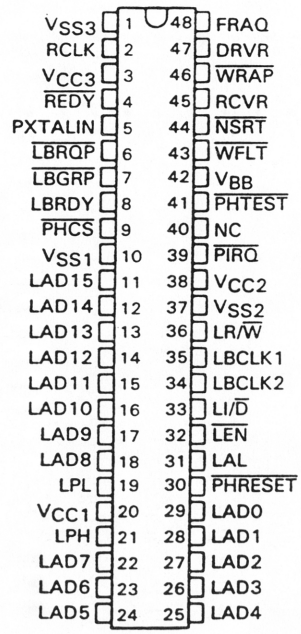
ADVANCE INFORMATION

TMS38020 PROTOCOL HANDLER

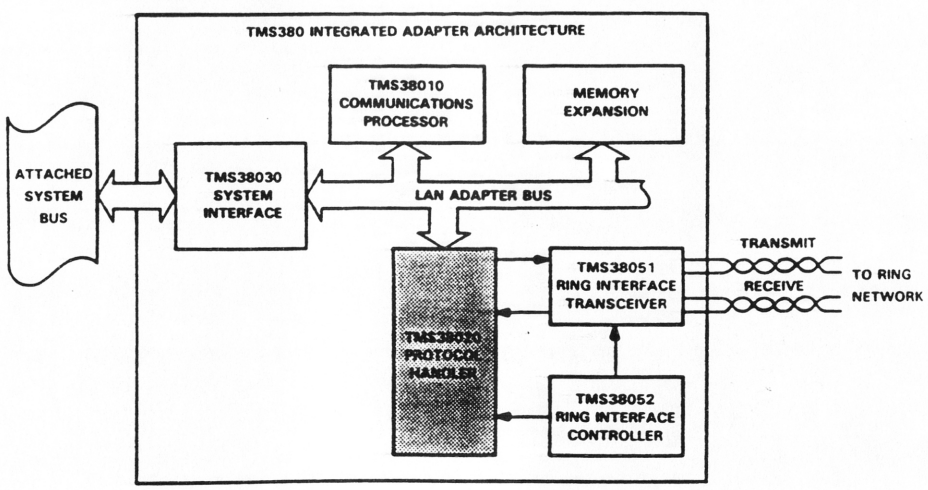
SEPTEMBER 1985

- Compatible with IEEE Std 802.5-1985 Token Ring Access Method and Physical Layer Specifications
  - Differential Manchester Code Conversion on 4M-Bit per Second Serial Data Stream
  - Address Recognition (Functional, Group and Specific)
  - Manchester Code Violation Detection
  - Starting and Ending Delimiter Generation and Detection
  - CRC Generation and Checking
  - High-Speed Frame Repeat Path Minimizes Ring Latency (2-Bit Times)
  - Token Transmit and Priority Control
  - Monitor Functions
- Separate Pairs of DMA Channels for Receive and Transmit
- Automatic Frame Buffer Management
- On-Chip 16K-Byte ROM for Adapter Software
  - 8K x 18-Bit ROM with Byte Parity Protection
  - Single Word Prefetch
- Test Pin for Hi-Z, Module-in-Place Testing
- 48-Pin, 600-Mil, Ceramic Dual-in-Line Packaging
- Low-Power Scaled-NMOS Technology

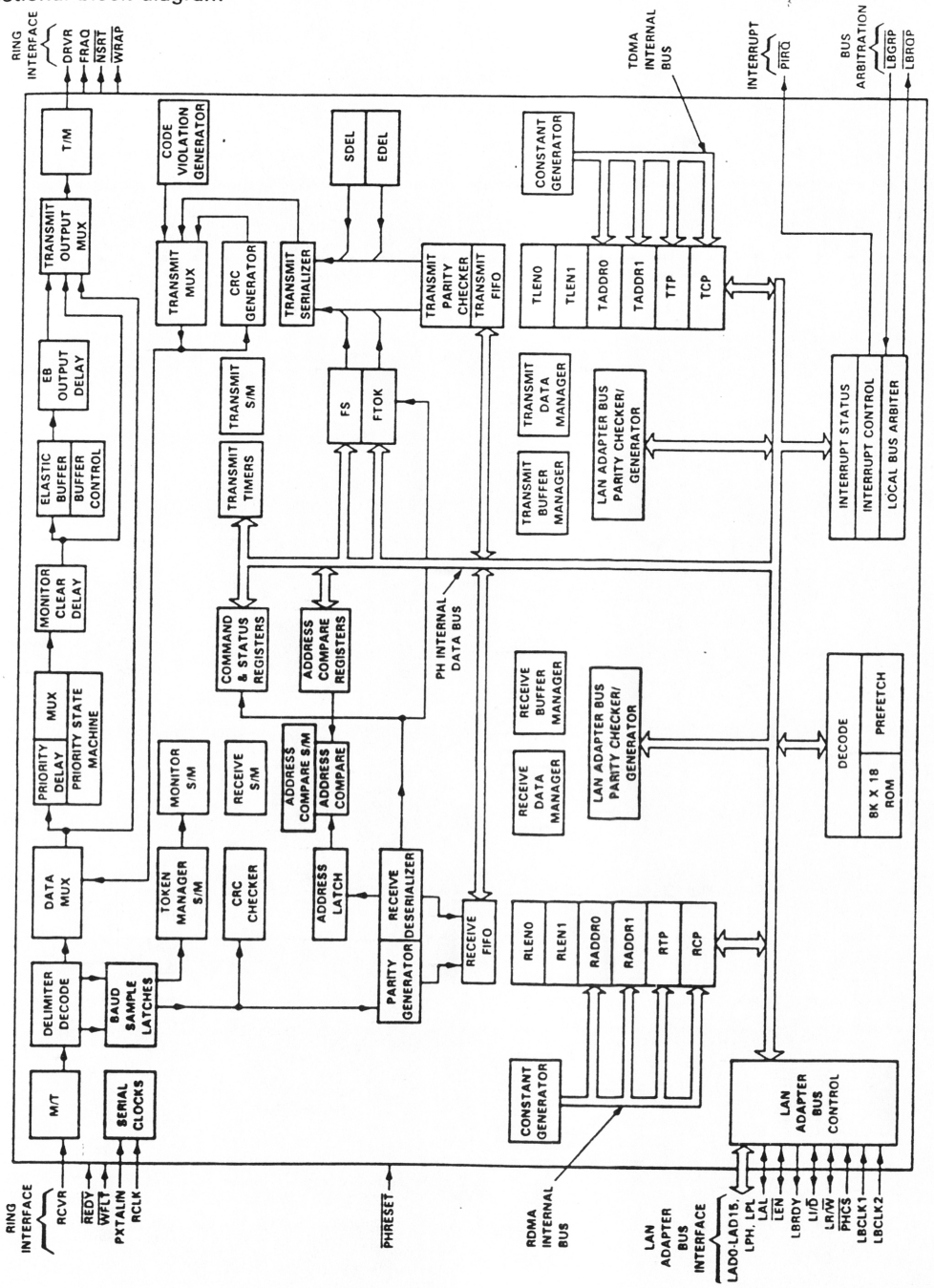
JD PACKAGE
(TOP VIEW)



token ring LAN application diagram



functional block diagram



Details of the functions are give in [20].

Appendix A.3 The Diagram of the TMS38030 System Interface

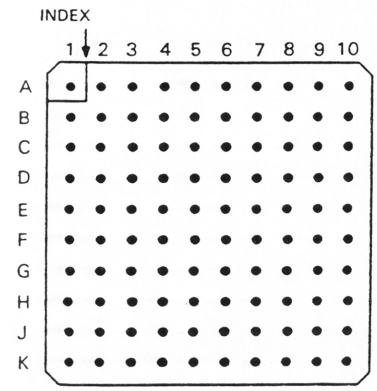
ADVANCE INFORMATION

TMS38030 SYSTEM INTERFACE

SEPTEMBER 1985

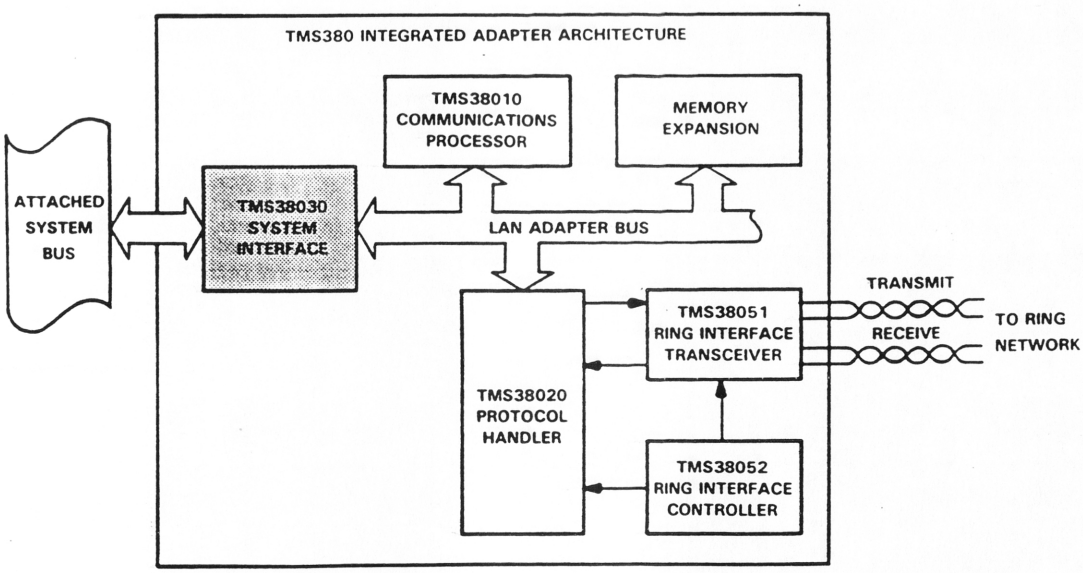
- Connects Two High-Speed Asynchronous Buses
  - —Up to 5M Bytes/Second DMA on Host System Bus
  - —6M Bytes/Second DMA on LAN Adapter Bus
- Provides Dual-Port DMA and Direct I/O Transfer Between Buses
- Selectable Host System Bus Options
  - —808X- or 680XX-Type Bus and Memory Organization
  - —8- or 16-Bit Data Bus for 808X-Type Buses
  - —Optional Parity Checking
- Provides Direct Control of Latches and Drivers on Host System Bus Interface
- Test Pin for Hi-Z, Module-In-Place Testing
- Single 5-V Supply
- 100-Pin Ceramic Grid Array Package
- Low-Power Scaled-NMOS Technology

GB PACKAGE†
(TOP VIEW)

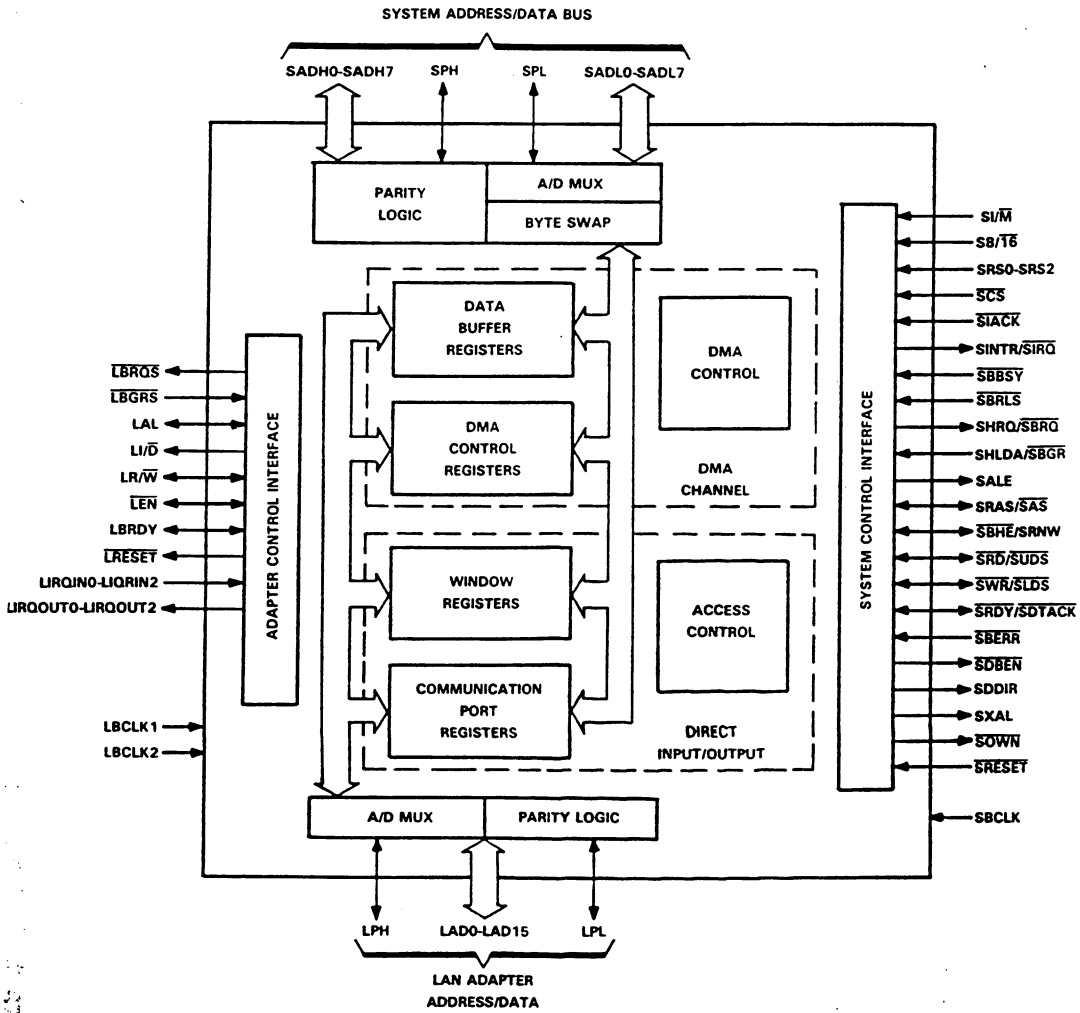


†See pin description table (Page 2) for location and description of all pins.

token ring LAN application diagram



functional block diagram†



†For signal names separated by a slash (/), the first signal name given is for the 808X mode and the second signal name is for the 680XX mode.

Details of the functions are given in [20].

Appendix A.4 The Diagram of the TMS38051 Ring Interface Tranceiver and teh TMS38052 Ring Interface Controller

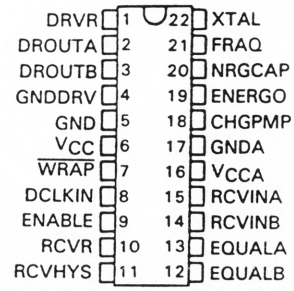
ADVANCE INFORMATION

TMS38051, TMS38052
RING INTERFACE CIRCUITS

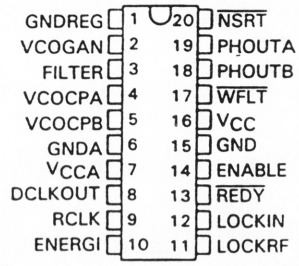
SEPTEMBER 1985

- Token Ring Electrical Connection
- Compatible with Electrical Interface of IEEE Std 802.5-1985 Token Ring Access Method and Physical Layer Specifications
- Phase-Lock Loop for Clock Generation from Data Signal
- 4 Megabit per Second Differential Manchester-Encoded Data Rate
- Independent Transmit and Receive Channels
- Phantom Drive for Physical Insertion into Ring
- Cable Wire-Fault Indication
- Receive Data-Loss Detection
- Receiver Frequency Equalization and Low-Level Hysteresis Circuit
- Loop Back (Wrap Mode) for Self-Test Diagnostics
- Two Chip Set
  - —TMS38051 Ring Interface Transceiver (22 Pin)
  - —TMS38052 Ring Interface Controller (20 Pin)
- Single 5-V Supply
- Low-Power Schottky Technology

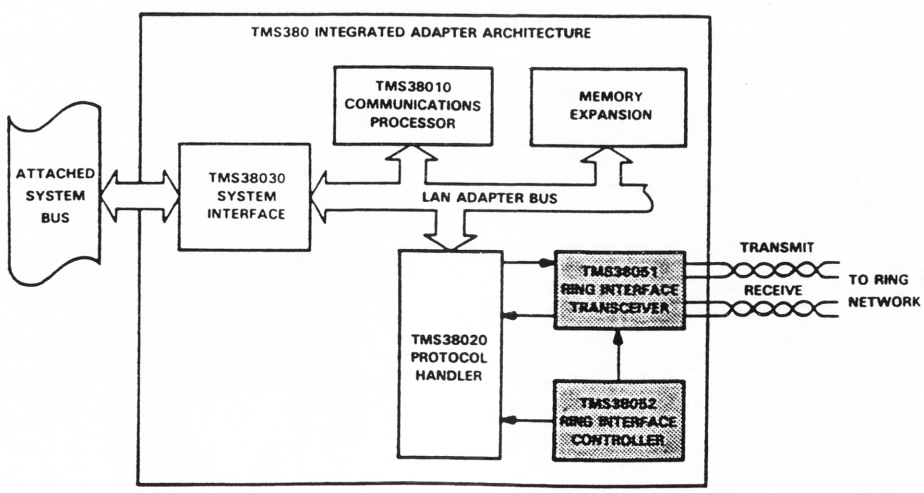
TMS38051 . . . N PACKAGE
(TOP VIEW)



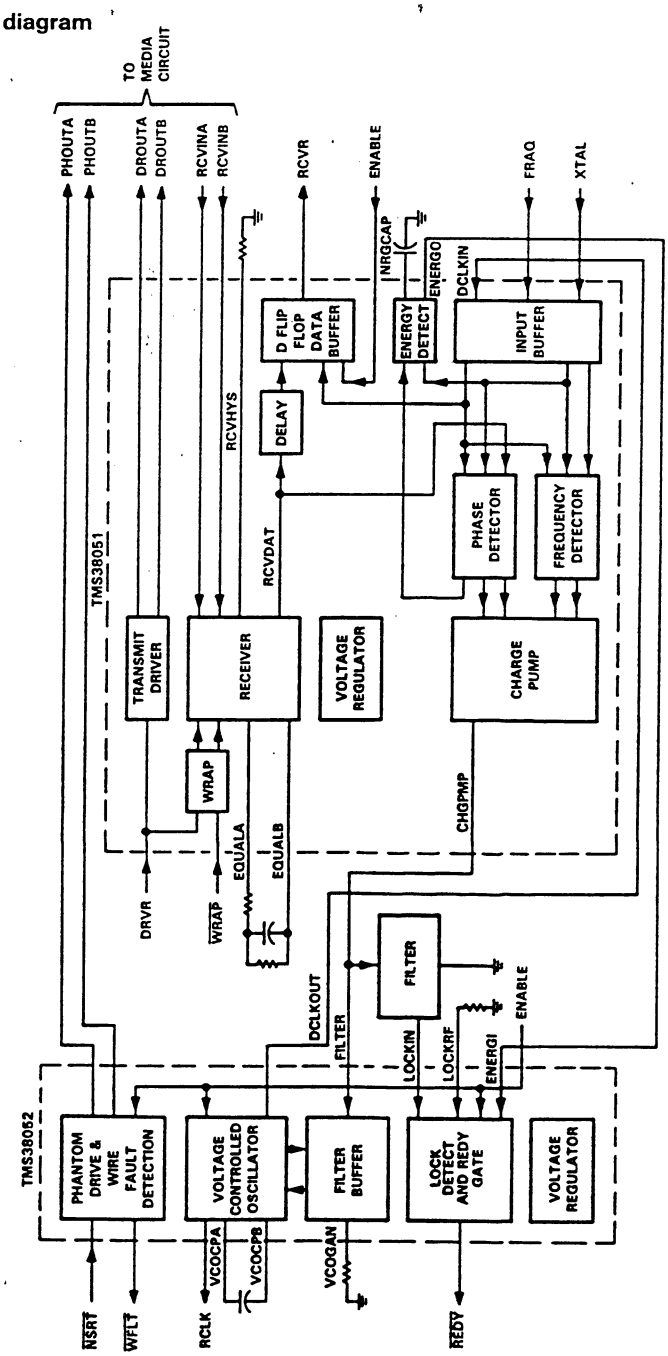
TMS38052 . . . N PACKAGE
(TOP VIEW)



token ring LAN application diagram



functional block diagram



Details of the functions are given in [20].

Appendix B The Dissimilarity between XLNET and TMS-IBM Ring in Layer Protocols

Appendix B.1 Session Layer

In addition to the format of the Session layer signal, part of which is shown in Tab.5.2, the difference existing between the two subnets in the Session layer as illustrated in Figs. B.1 and B.2 is summarized as follows:

1. There are 6 states in the Session Layer in TMS-IBM Ring and 8 states in XLNET. The two more states in XLNET are "sess-dial-rec-1" and "sess-dial-rec-2" which are concerned with the dial reception. These states have been merged in the "sess-closed" state in TMS-IBM Ring.

2. In the "sess-await-info-con" state of TMS-IBM Ring, once a close signal arrives, the state will move to the "sess-closed". However, in XLNET, the only one state related to the "sess-await-info-con" is the "sess-opened" state. In other words, it is not able to move the state to the "sess-closed" state directly.

3. Some signals travelling between these states are different in the two subnets. For example, the signals travelling from the "sess-opened" state to the "sess-closed" state in TMS-IBM Ring include 's-close-ind', 'switch-hook', 'sess-close' and 'ts-close'. But in XLNET, they are just 'switch-hook' and 'T-close' (the same as the 'ts-close' in its partner).

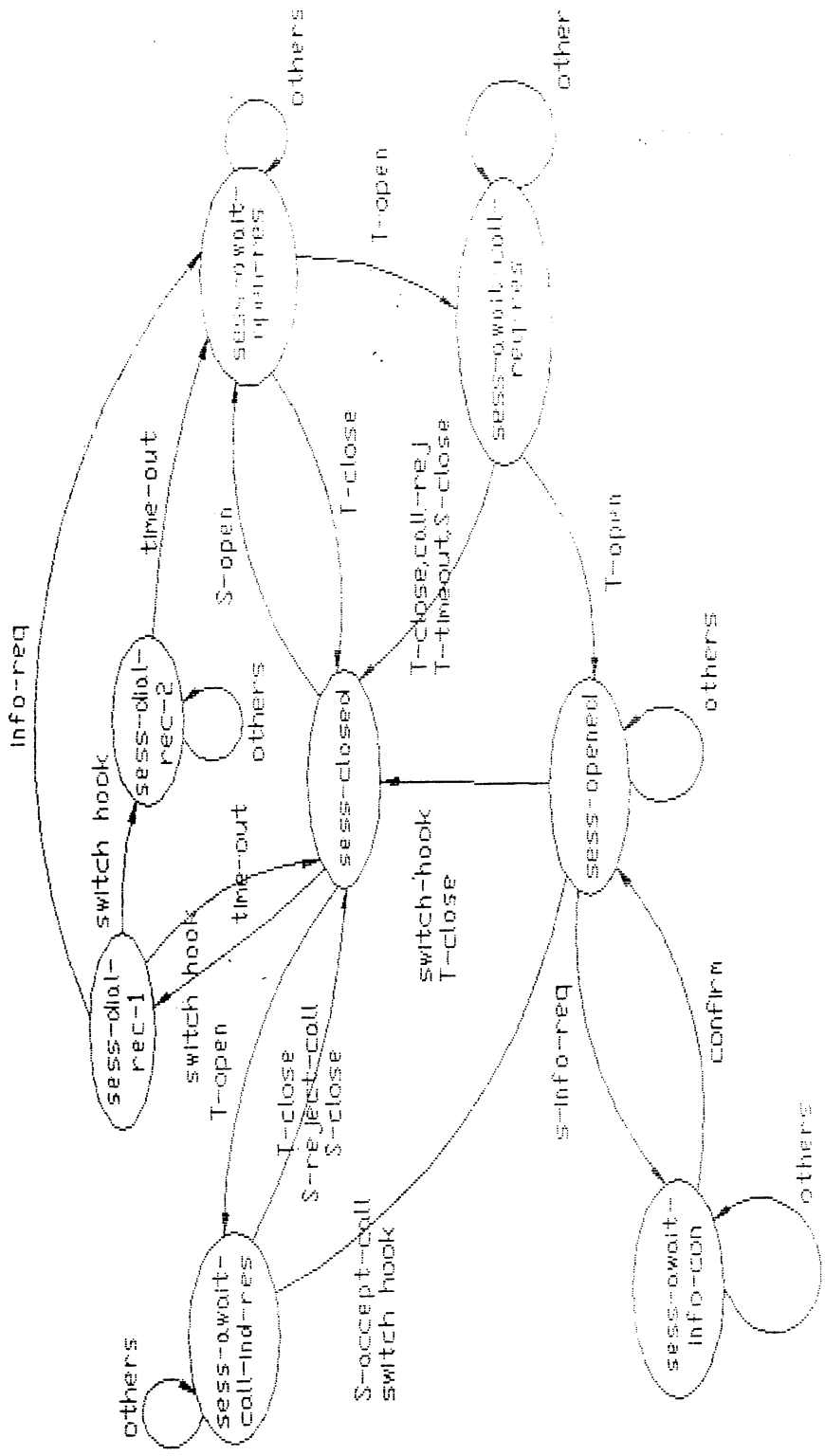


Fig. B.1 Session Layer Circuit State Diagram (XLNET)

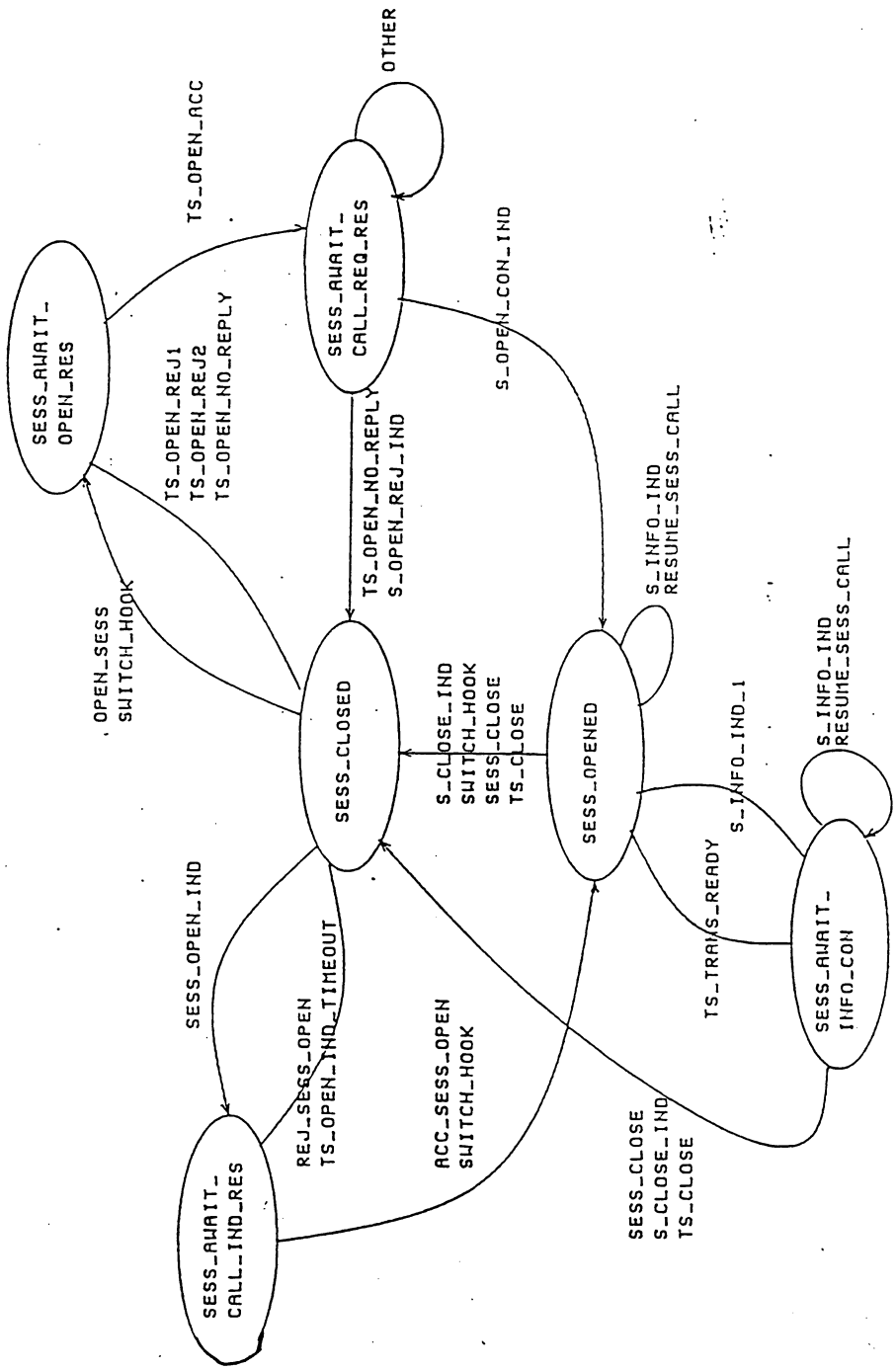


Fig. B.2 Session Layer circuit State Diagram (TMS-IBM Ring)

Appendix B.2 Transport Layer

In addition to the format of the Transport layer signal, part of which is shown in Tab. 5.2, the difference existing between the two subnets, XLNET and TMS-IBM Ring the Transport layer as illustrated in Figs. B.3 and B.4, is summarized in the following points:

1. There are 8 states in the Transport layer in TMS-IBM Ring and 10 states in XLNET. The two more states in XLNET are "trans-await-info-confirm" and "trans-await-clr-confirm". They are provided for the information confirm and transport close confirm, respectively. These states have been merged in the "trans-await-info-req-con" and "trans-closed" states, respectively.

2. There is a path between the states of "trans-closed" and "trans-opened" in TMS-IBM Ring, but no path between those in XLNET. It needs to pass through the "trans-await-clr-confirm" state before it reaches the "trans-closed" state.

3. Some signals travelling between these states are different in the two subnets. For example, the signals travelling from the "trans-await-open-acc" to the "trans-closed" state in TMS-IBM Ring include 't-open-rej2-ind', 'lt-open-rej' and 't-timeout'. But in XLNET, only the 'timeout' signal travels the path.

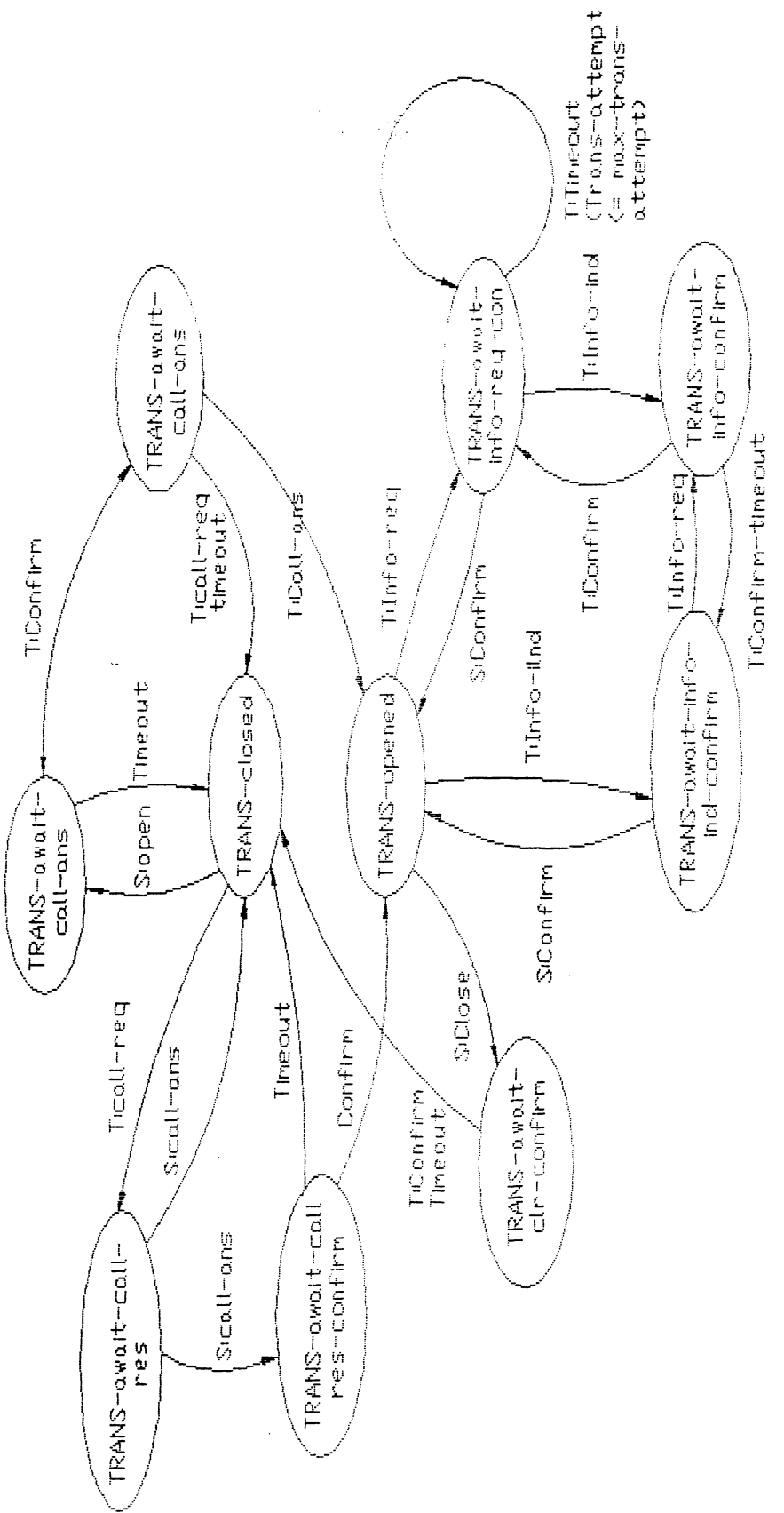


Fig. B.3 **Transport Layer Circuit State Diagram (XLNET)**

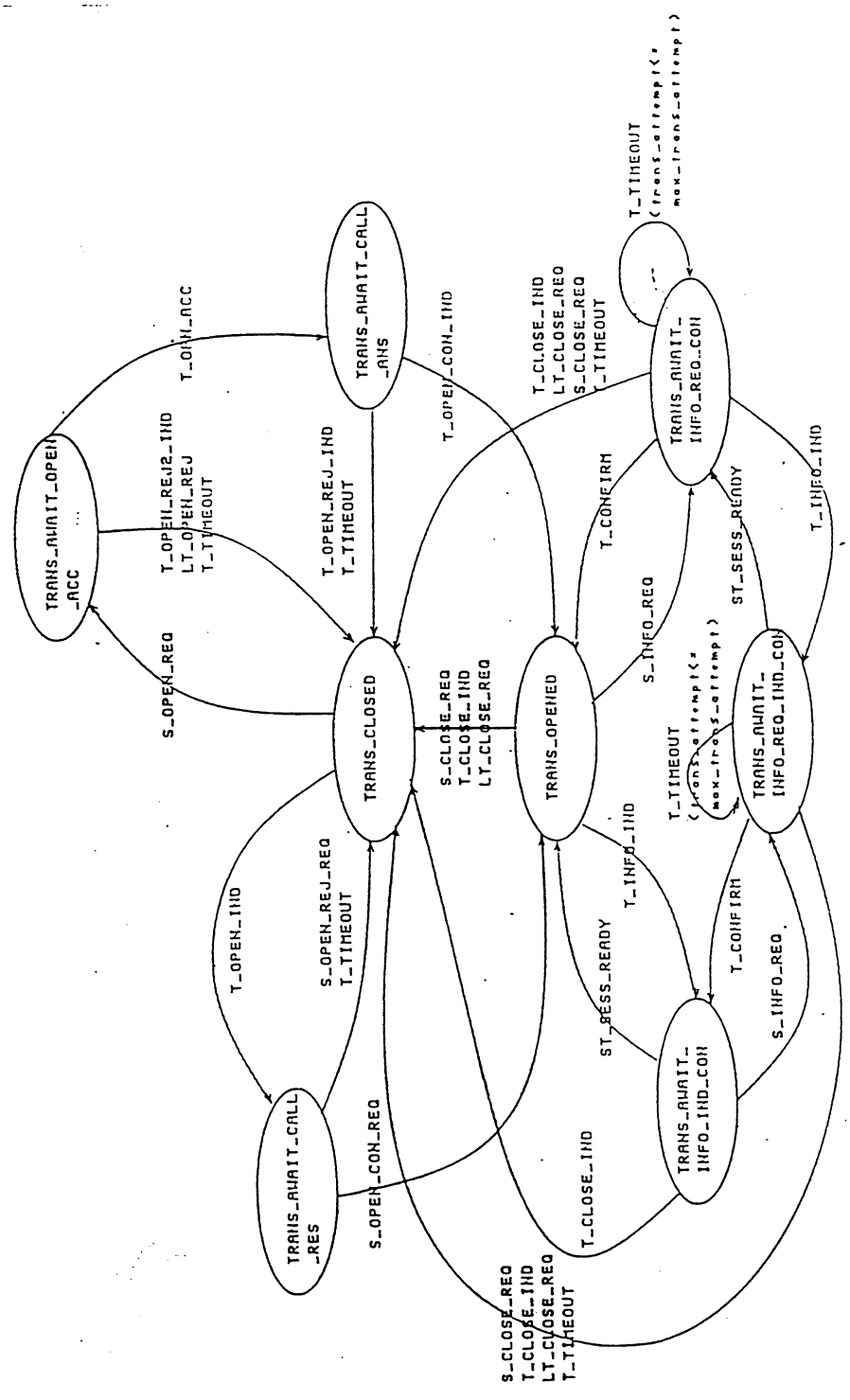


Fig. B.4 Transport Layer Circuit State Diagram (TMS-IMB Ring)

Appendix B.3 Call Establishment Protocol

From Figs. B.5 and B.6, we find, in XLNET, when the Transport Layer of the node, which calls for a connection to its partner, receives the answer from its partner, it needs to issue a confirm packet to its partner. But, there is no matching function in TMS-IBM Ring.

Fig. B.5 Call Establishment Protocol in XLNET

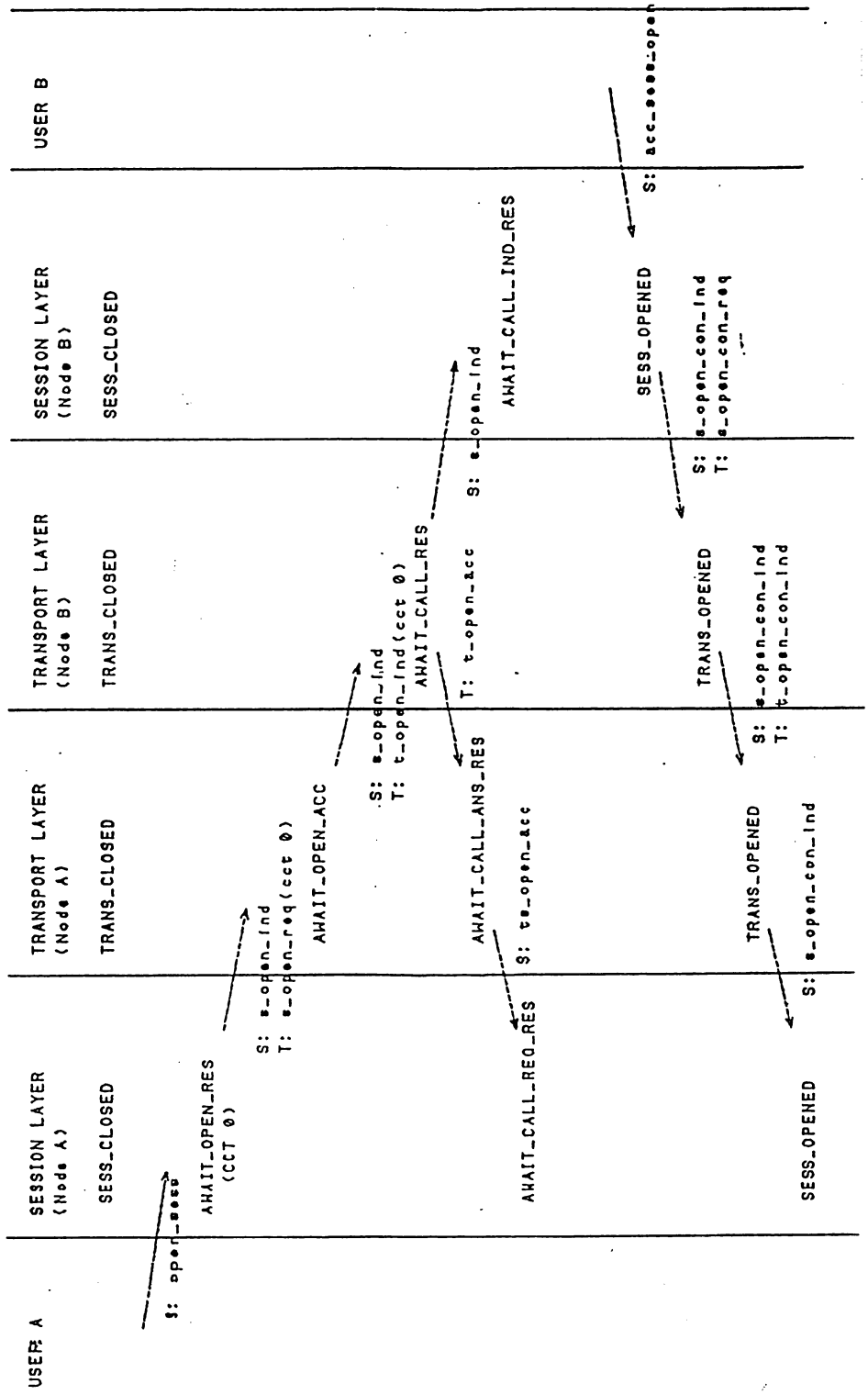


Fig. B.6 Call Establishment Protocol in TMS-IBM Ring

Appendix B.4 Conclusion

From the above observations, we find that, most of the Transport and Session Layer protocols having the difference between the two subnets are of the types of the interface and control protocol, which together with the dissimilarities in these layer state transitions, do not have much influence on the peer-to-peer communication.

Regarding the peer protocol, the call establishment protocol in XLNET is modulated to that of TMS-IBM Ring. In other words, when the node calling for the connection receives the answer from its partner, it does not send a confirm message to its partner any more.

Appendix C The Data Sheet of the VLSI VT7132A Dual- Port RAM

VLSI TECHNOLOGY, INC.

PRELIMINARY **VT7132A • VT7142A**

HIGH-SPEED 2,048 x 8 CMOS DUAL-PORT RAM

FEATURES

- High speed:
  — 30, 35, and 45 ns access
- Fully static operation
- Full contention arbitration
- VT7142A slave for bus expansion
- Output enable function
- Separate port power-down
- Advanced CMOS technology
- Dual interrupt flags in PLCC
- Low power: 150 mA (max) operating
- 48-pin DIP or 52-pin PLCC

DESCRIPTION

The VT7132A and VT7142A are 16,384-bit dual-port static random access memories that are organized as 2,048 8-bit words. The VT7132A is designed to be used as a stand-"master" dual-port RAM with the VT7142A "slave" dual-port RAM in a system application larger than 8 bits. The master/slave approach in large bus systems requires no external contention logic.

The VT7132A/VT7142A feature two separate I/O ports that each allow independent access for read or write to any location in the memory. The memory is designed to permit read and/or write operations to be performed at both ports at the same time. Contention arbitration logic is provided to eliminate overlapping operations to the same memory location.

The on-chip contention logic arbitrates and delays one port until the other port's operation is completed. When this occurs, a Busy flag is sent to the side delayed. This flag stays set until the first operation is complete. When both sides request at exactly the same time, the left port takes priority.

When used in the 52-pin PLCC package, a dual-level interrupt function is available. The interrupt function acts like writable flags and is provided to allow communication between systems. When the flag's location is written from one side, the other side's $\overline{INT}$ pin goes LOW until the flag location is read by that side.

One flag is set during a write operation to any location and the other flag is set during a write to location 7FF/7FE.

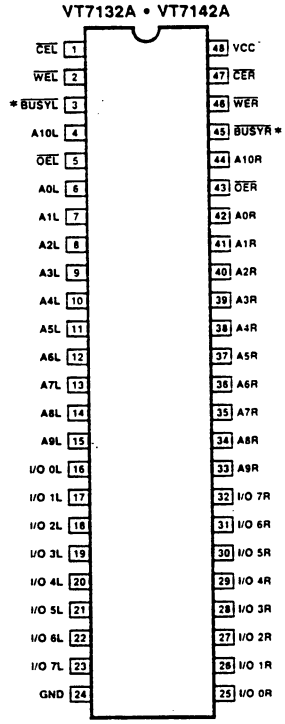
Both Interrupt and Busy flags are open drain for simple wired OR operation.

Automatic power down for each port is controlled independently by its Chip Enable input.

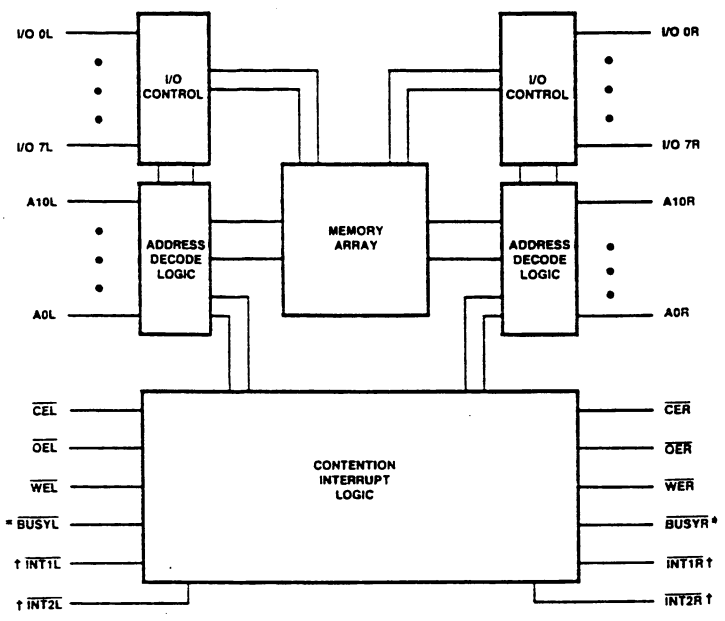
Interfacing to the VT7132A/VT7142A is further simplified by the incorporation of an Output Enable control for each port.

PIN DIAGRAM

VT7132A • VT7142A



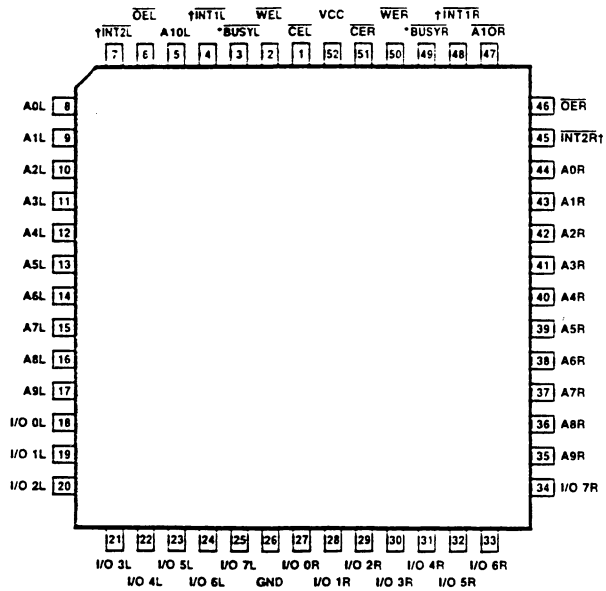
BLOCK DIAGRAM



* OPEN-DRAIN OUTPUTS FOR VT7132A. INPUTS FOR VT7142A.
† AVAILABLE ONLY WITH 52-PIN PLCC.

PIN DIAGRAM

VT7132A • VT7142A



* OPEN-DRAIN OUTPUTS FOR VT7132A. INPUTS FOR VT7142A.
† AVAILABLE ONLY WITH 52-PIN PLCC.

FUNCTIONAL DESCRIPTION

The VT7132A and VT7142A are 16,384-bit dual-port RAMs that feature two separate I/O ports. Each allows independent access for reading or writing to any location in the memory.

PORT ENABLING

The VT7132A/VT7142A feature separate left- and right-port chip enable controls ($\overline{CEL}$ and $\overline{CER}$) that activate their respective ports when they go LOW (see Table 1). When a port is active, it is allowed access to the entire memory array. When a chip enable signal is HIGH, its side of the device remains in a standby power mode as long as it does not change state.

Each port has an output enable control ($\overline{OEL}$ and $\overline{OER}$) that keeps its respective output in a high-impedance state when HIGH. When a port's $\overline{OE}$ is LOW and its write enable ($\overline{WE}$) input HIGH, its output bus drivers are turned on.

Separate write enable inputs ($\overline{WEL}$ and $\overline{WER}$) control writing of new data into any location in the RAM from either port. For example, when the left-port write enable ($\overline{WEL}$) is LOW, new data can be written into the location selected by the left-port address field. When a port's $\overline{WE}$ input is HIGH, data can be read from that port if its respective $\overline{OE}$ line is LOW. For example, when $\overline{WEL}$ is HIGH and $\overline{OEL}$ LOW, data can be read from the location selected by the left-port address field. Similarly, $\overline{WER}$ HIGH and $\overline{OER}$ LOW allows data to be read from the location selected by the right-port address field.

CONTENTION ARBITRATION

Contention for a memory location can occur when both the left and right ports are active and the port addresses match. Two modes of operation are provided for this condition, with the $\overline{OE}$ inputs controlling which mode is actually used:

1. On-chip control logic arbitrates the situation.
2. The contention is ignored and both ports are given access to the addressed memory location.

On-chip control logic arbitration is used if the addresses at the ports match and both $\overline{CEL}$ and $\overline{CER}$ go LOW while $\overline{OEL}$ and $\overline{OER}$ are HIGH. In this case, priority is given to the port whose $\overline{CE}$ first becomes valid; the other port is not allowed access to the memory core until the first port's operation is completed. If $\overline{CEL}$ and $\overline{CER}$ become valid simultaneously, the arbitration logic gives priority to the left port.

If both $\overline{CE}$ inputs are valid while the $\overline{OE}$ controls are HIGH and an address change occurs that causes an address match, priority is given to the port whose address becomes valid first. If both addresses become valid at the same time and match, the left port is given priority.

Contention is ignored and either one or both ports has access to the memory core if the $\overline{OE}$ inputs are LOW when the contention occurs.

That is, the core is accessible to a port, even if the on-chip logic would have delayed its access, if:

1. Its $\overline{OE}$ is already LOW when its $\overline{CE}$ goes LOW while the addresses at the ports match.
2. Its $\overline{OE}$ is already LOW and both ports are active when an address change occurs that causes an address match.

It is therefore possible for both ports to have access to one memory location at the same time, even in a WRITEL-WRITER situation.

BUSY FLAGS

Separate Busy flags ($\overline{BUSYL}$ and $\overline{BUSYR}$) are provided to signal when a port's access to the memory core has been delayed. This permits the user to stop the processor connected to the losing port and add wait states, if desired.

When both ports try to access the same memory location, the on-chip arbitration logic causes the Busy flag to go LOW on the side that is delayed. This occurs rapidly enough that, if the user wishes, the processor's address and data can be preserved. The Busy flags are operational even when the device is in the ignore-contention mode, so they can be used to indicate that contention has occurred and data may have been changed.

INTERRUPTS

Interrupt logic is included on-chip to provide a means for two processors to communicate with one another. The interrupt function acts like a writeable flag, so that when the location of an Interrupt flag is written from one port, the other port's interrupt input goes LOW until that port reads the flag's location. If, for example, the left port writes to memory location 7FF, the right port Interrupt flag ($\overline{INTR}$) is latched LOW until the right port reads data from the same location. Similarly, if the right port writes to memory location 7FE, the left port Interrrupt flag ($\overline{INT1L}$) is latched LOW until the left port reads from that location. If both ports are enabled and contention occurs, the Busy circuitry disables the address decoder from setting or resetting the Interrupt flags.

The VT7132A and VT7142A also provide a second, more general Interrupt flag ($\overline{INT2L}$, $\overline{INT2R}$) at each port. When this feature is used, either port writing to any location sets the other port's flag. If, for example, the left port writes to any location, the right port second Interrupt flag ($\overline{INT2R}$) is latched LOW until the right port reads from any location. This allows the total memory to serve as a system mail box.

MASTER/SLAVE OPERATION

Expanding the data bus width beyond 8 bits in a dual-port RAM system implies that more than one memory chip will be active at the same time. Because of system timing and skews, this could result in $\overline{BUSYL}$ being active on one device while $\overline{BUSYR}$ is active on another. To avoid this lock-out condition, the VT7132A/VT7142A master/slave approach allows arbitration to be performed by only one of the memory devices.

In such a system, one VT7132A master would be used in conjunction with one or more VT7142A slaves, which would be configured to fill the additional bus width. The $\overline{BUSY}$ inputs of the VT7142A interface with the $\overline{BUSY}$ outputs of the VT7132A without the need for external components, maintaining system performance.

When expanding the width of dual-port RAMs, writing to the slave RAMs must be delayed until after the $\overline{BUSY}$ input has settled. If this is not done, the slave chip may begin a write cycle during a contention situation. Conversely, the write pulse must extend a hold time beyond $\overline{BUSY}$ to ensure that a write cycle occurs after the contention is resolved. The write pulse to the slave should be delayed by the maximum arbitration time of the master so that, if contention occurs, the write to the slave will be inhibited by the $\overline{BUSY}$ signal from the master.

TABLE 1 READ/WRITE CONTROL FUNCTIONS

Left-Port Signals							Right Port Signals							
$\overline{CEL}$	$\overline{OEL}$	$\overline{WEL}$	$\overline{BUSYL}$	$\overline{INT1L}$	$\overline{INT2L}$	A0L–A9L	$\overline{CER}$	$\overline{OER}$	$\overline{WER}$	$\overline{BUSYR}$	$\overline{INT1R}$	$\overline{INT2R}$	A0R–A9R	Function
H	X	X	H	X	X	X	X	X	X	H	X	X	X	Left Port Power-Down
X	X	X	H	X	X	X	H	X	X	H	X	X	X	Right Port Power-Down
L	L	H	H	X	X	X	X	X	X	X	X	X	X	Read Left Port
L	L	L	H	X	L	X	X	X	X	X	X	X	≠	Write Left Port
X	X	X	X	X	X	=	L	L	H	H	X	X	X	Read Right Port
X	X	X	X	X	X	=	L	L	L	H	X	L	≠	Write Right Port
L	L	L	L	X	X	=	L	L	L	H	X	L	=	Left Port Busy
L	L	L	H	X	L	=	L	L	L	L	X	X	=	Right Port Busy
L	X	L	H	L	L	7FF	X	X	X	H	L	X	X	Left Flag Right Interrupt
X	X	X	H	L	X	X	L	X	L	H	L	L	7FE	Right Flag Left Interrupt

ABSOLUTE MAXIMUM RATINGS

Ambient Operating Temperature	−10°C to +80°C
Storage Temperature	−65°C to +150°C
Supply Voltage to Ground Potential	−0.5 V to +7.0 V
Applied Output Voltage	−0.5 V to +7.0 V
Applied Input Voltage	−0.5 V to +7.0 V
Power Dissipation	1.0 W
DC Output Current	50 mA

Stresses above those listed may cause permanent damage to the device. These are stress ratings only, and functional operation of this device under these or any conditions above those indicated in this data sheet is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC CHARACTERISTICS T_A = 0°C to +70°C, V_{CC} = 5 V ±10%

Symbol	Parameter	Limits Min	Typ	Max	Unit	Conditions
IIL	Input Leakage			5	µA	VCC = 5.5 V
IOL	Output Leakage			5	µA	VCC = 5.5 V, $\overline{CE}$ = VIH
ICC1	Active Current, Outputs Open			150	mA	VCC = 5.5 V, $\overline{CE}$ = VIL
ICC2	Standby Current, Both Ports			10	mA	$\overline{CEL}$ = $\overline{CER}$ = VIH
ICC3	Standby Current, Active-Port Outputs Open			70	mA	$\overline{CEL}$ or $\overline{CER}$ = VIH
ICC4	Standby Current, CMOS Levels			100	µA	$\overline{CEL}$ = $\overline{CER}$ = $\overline{OEL}$ = $\overline{OER}$ ≥VCC −0.2 V or ≤0.2 V
VIL	Input LOW Voltage	−0.5		0.8	V	
VIH	Input HIGH Voltage	2.2		VCC +1	V	Note 1
VOL1	Output LOW Voltage			0.4	V	IOL = 6 mA
VOL2	Output LOW Voltage, Open-Drain Outputs			0.5	V	IOL = 16 mA
VOH	Output HIGH Voltage	2.4			V	IOH = −4.0 mA

DATA RETENTION DC CHARACTERISTICS T_A = + 25°C, V_{CC} = 2 V

Symbol	Parameter	Limits Min	Typ	Max	Unit	Conditions
VDR	Data Retention VCC	2.0			V	Note 2
ICCDR	Data Retention Current		1	50	µA	Note 2

CAPACITANCE T_A = +25°C, f = 1 MHz

Symbol	Parameter	Typ	Max	Unit	Conditions
COUT	Output Capacitance		10	pF	Note 3
CIN	Input Capacitance		10	pF	Note 3

AC TEST CONDITIONS

Input Voltage Levels	0 V to +3 V
Input Rise and Fall Times	5 ns
Input Reference Levels	1.5 V
Output Reference Levels	1.5 V
Output Load	Figure 1, 2, and 3

AC TESTING LOAD CIRCUITS

FIGURE 1. OUTPUT LOAD CIRCUIT A

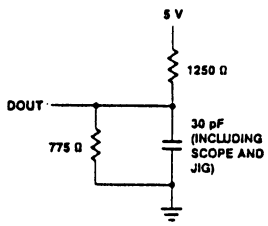


FIGURE 2. OUTPUT LOAD CIRCUIT B (Note)

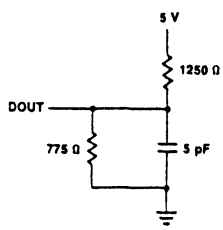
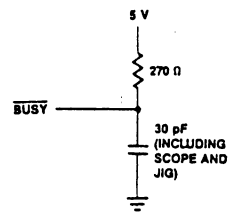
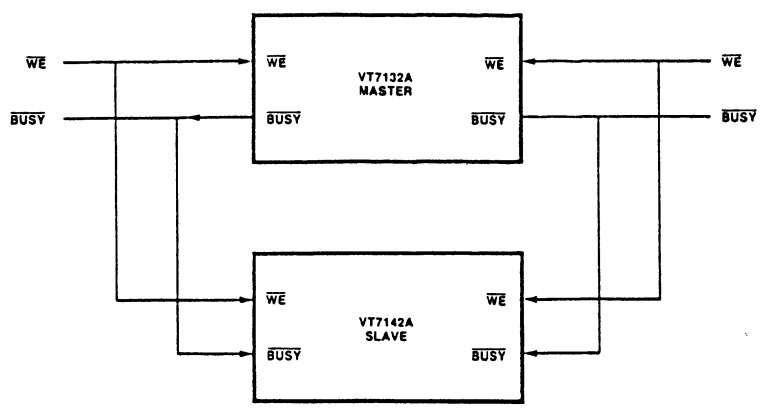


FIGURE 3. $\overline{\text{BUSY}}$ OUTPUT LOAD CIRCUIT (VT7132A)



NOTE: FOR tHZ, tLZ, tWZ, and tOW.

MASTER SLAVE EXPANSION TO 16-BIT MEMORY SYSTEM



TIMING CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5\ V \pm 10\%$

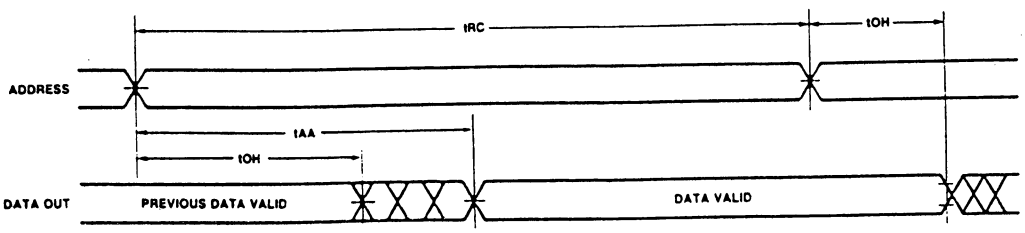
Symbol	Parameter	VT7132A-30		VT7132A-35		VT7132A-45		Units	Conditions
		Min	Max	Min	Max	Min	Max		
READ CYCLE									
tRC	Read Cycle Time	30		35		45		ns	
tAA	Address Access Time		30		35		45	ns	
tACE	Chip Enable Access Time		30		35		45	ns	Note 1
tAOE	Output Enable Access Time		15		15		20	ns	
tOH	Output Hold from Address Change	0		0		0		ns	
tLZ	Output Low Z Time	0		0		5		ns	Notes 2, 3
tHZ	Output High Z Time	0	15	0	15	0	20	ns	Notes 2, 3
tPU	Chip Enable to Power-Up Time							ns	Note 2
tPD	Chip Disable to Power-Down Time		15		15		15	ns	Note 2
WRITE CYCLE									
tWC	Write Cycle Time	30		35		45		ns	Note 4
tEW	Chip Enable to End of Write	30		30		35		ns	
tAW	Address Valid to End of Write	25		30		35		ns	
tAS	Address Set-Up Time	0		0		0		ns	
tWP	Write Pulse Width	20		20		30		ns	
tWR	Write Recovery Time	0		0		0		ns	
tDW	Data Valid to End of Write	15		15		20		ns	
tDH	Data Hold Time	0		0		0		ns	
tWZ	Write Enable to Output High Z	0	15	0	15		20	ns	Notes 2, 3
tHZ	Output High Z Time	0	15	0	15	0	20	ns	
tOW	Output Active from End of Write	0		0		0		ns	Notes 2, 3

TIMING CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5\ V \pm 10\%$

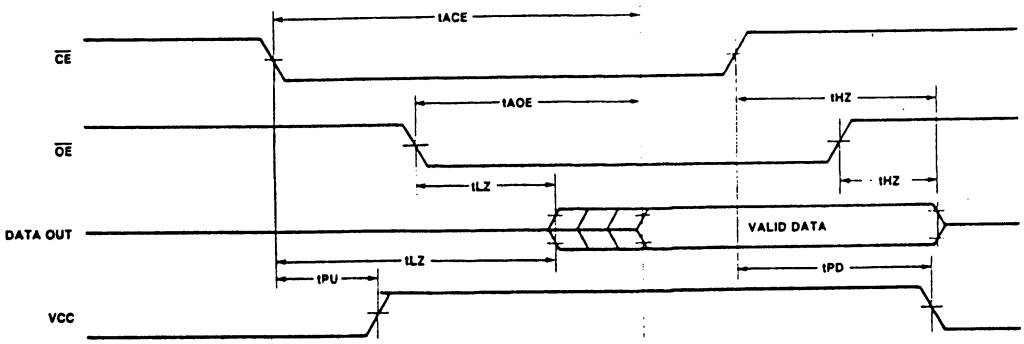
Symbol	Parameter	VT7132A-30		VT7132A-35		VT7132A-45		Units	Conditions
		Min	Max	Min	Max	Min	Max		
CONTENTION CYCLE									
tRC	Read Cycle Time	30		35		45		ns	
tWC	Write Cycle Time	30		35		45		ns	
tWB	Write to $\overline{BUSY}$	-5		-10		-10		ns	Notes 1, 2
tWH	Write Hold after $\overline{BUSY}$	20		20		20		ns	Note 5
tBAA	$\overline{BUSY}$ Access Time to Address		25		25		30	ns	
tBDA	$\overline{BUSY}$ Disable Time to Address		20		25		30	ns	
tBAC	$\overline{BUSY}$ Access Time to Chip Enable		25		25		30	ns	
tBDC	$\overline{BUSY}$ Disable Time to Chip Enable		20		25		30	ns	
tWDD	Write Pulse to Data Delay		35		40		50	ns	Note 3
tDDD	Write Data Valid to Read Data Delay		35		40		50	ns	Note 3
tBDD	$\overline{BUSY}$ Disable to Valid Data		Note 4		Note 4		Note 4		Note 4
tAPS	Arbitration Priority Set-Up Time	2		5		5		ns	
tAOS	Arbitration Over-Ride Set-Up Time	5		5		5		ns	
INTERRUPT TIMING									
tRC	Read Cycle Time	30		35		45		ns	
tWC	Write Cycle Time	30		35		45		ns	
tOEH	Output Enable Hold Time	5		5		5		ns	
tOER	Output Enable Recovery Time	0		0		5		ns	
tAS	Address Set-Up Time	0		0		0		ns	
tWR	Write Recovery Time	0		0		0		ns	
tINS	Interrupt Set Time		20		25		30	ns	
tINR	Interrupt Reset Time		25		25		30	ns	
tIAR	Interrupt Address Recovery Time		20		25		30	ns	
DATA RETENTION TIMING									
tR	Operation Recovery Time	tRC		tRC		tRC		ns	Notes 6, 7
tCDR	Chip Deselect to Data Retention Time	0		0		0		ns	Note 7

TIMING DIAGRAMS

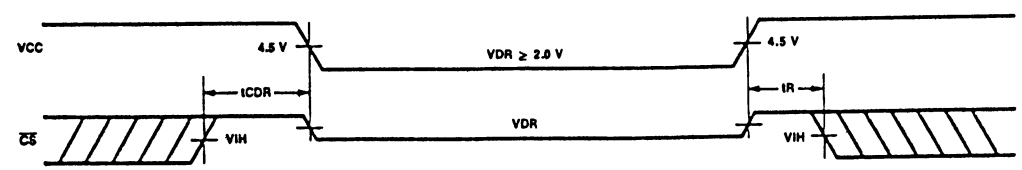
READ CYCLE NO. 1, EITHER SIDE, Notes 1 and 2



READ CYCLE NO. 2, EITHER SIDE, Notes 1 and 3

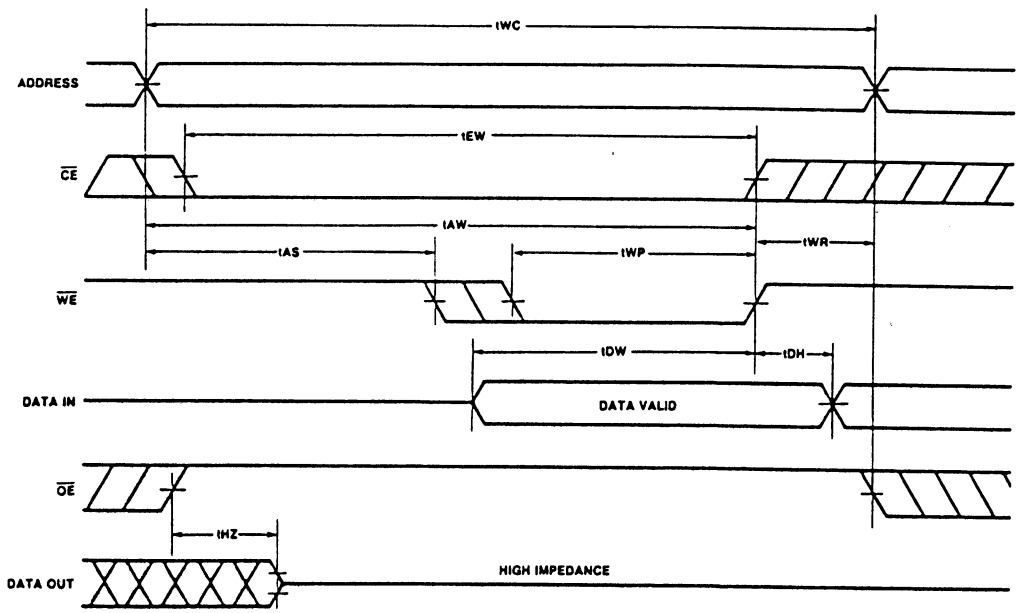


DATA RETENTION MODE

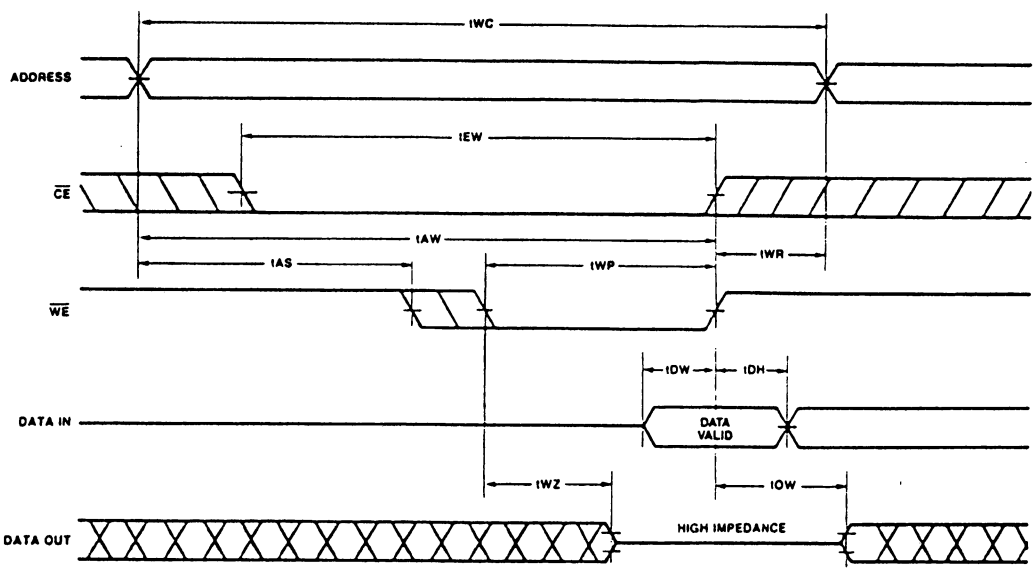


TIMING DIAGRAMS

WRITE CYCLE NO. 1, EITHER SIDE , Note 1



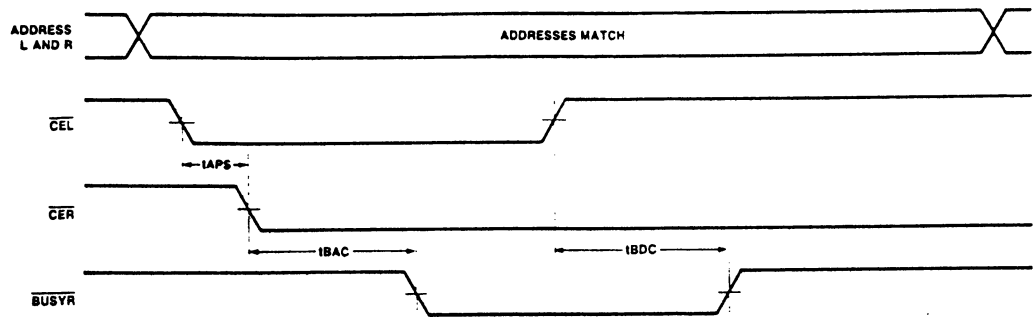
WRITE CYCLE NO. 2, EITHER SIDE ($\overline{OE}$ = VIL), Note 1



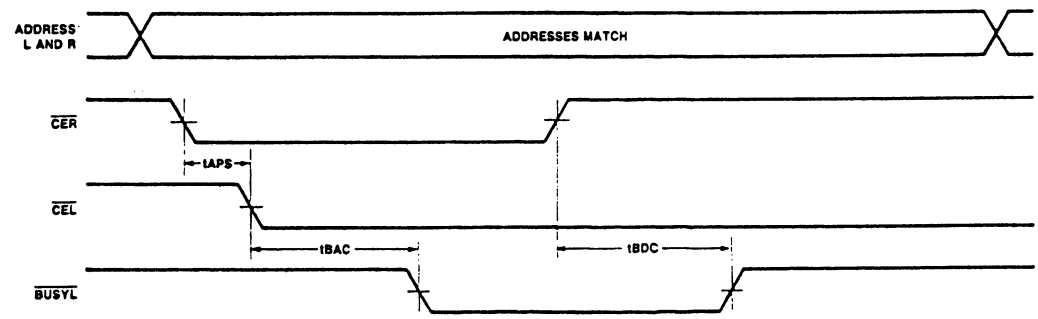
TIMING DIAGRAMS

CONTENTION CYCLE NO. 1, $\overline{CE}$ CONTENTION ARBITRATION MODE, Note 1, Page 5-65

$\overline{CEL}$ VALID FIRST

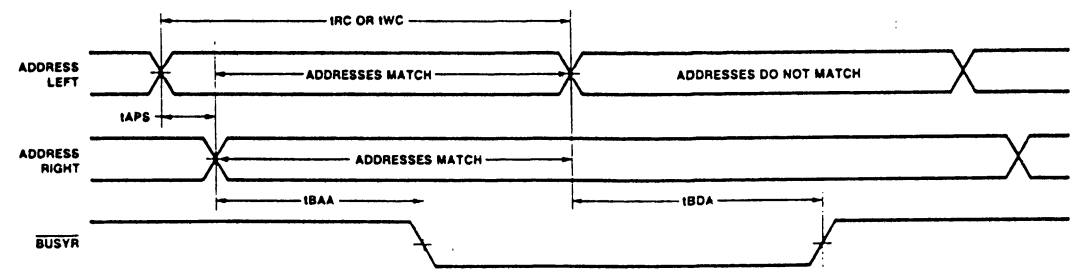


$\overline{CER}$ VALID FIRST

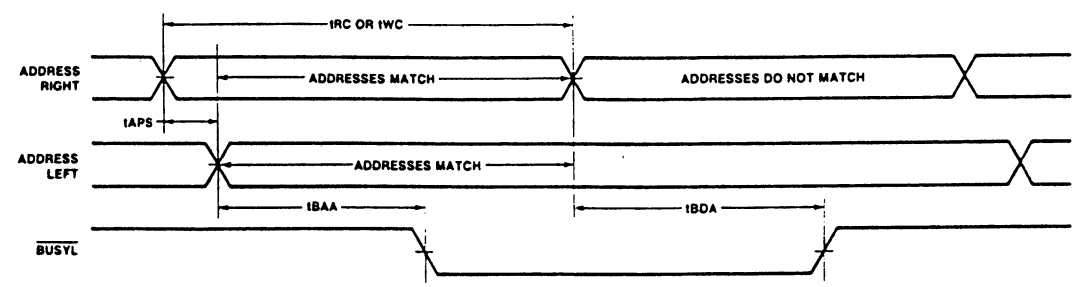


CONTENTION CYCLE NO. 2, ADDRESS CONTENTION ARBITRATION MODE, Notes 1 and 2, Page 5-65

ADDRESS LEFT VALID FIRST



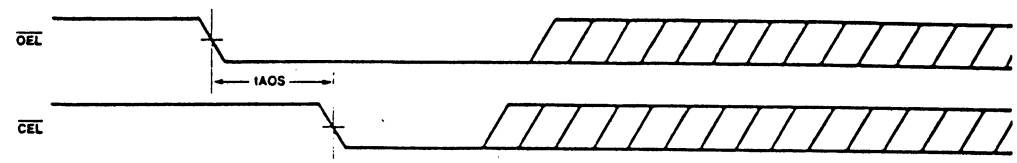
ADDRESS RIGHT VALID FIRST



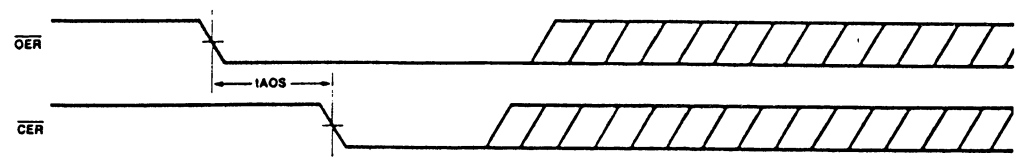
TIMING DIAGRAMS (Cont.)

CONTENTION CYCLE NO. 3, CONTENTION OVERRIDE MODE, Note 3

LEFT PORT CONTENTION IGNORED

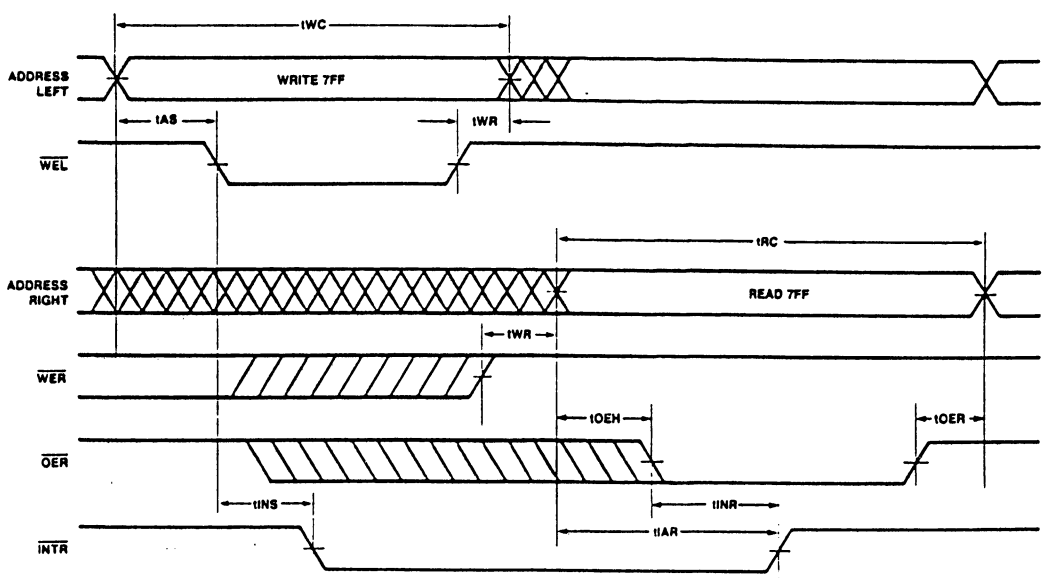


RIGHT PORT CONTENTION IGNORED



INTERRUPT MODE, Note 2

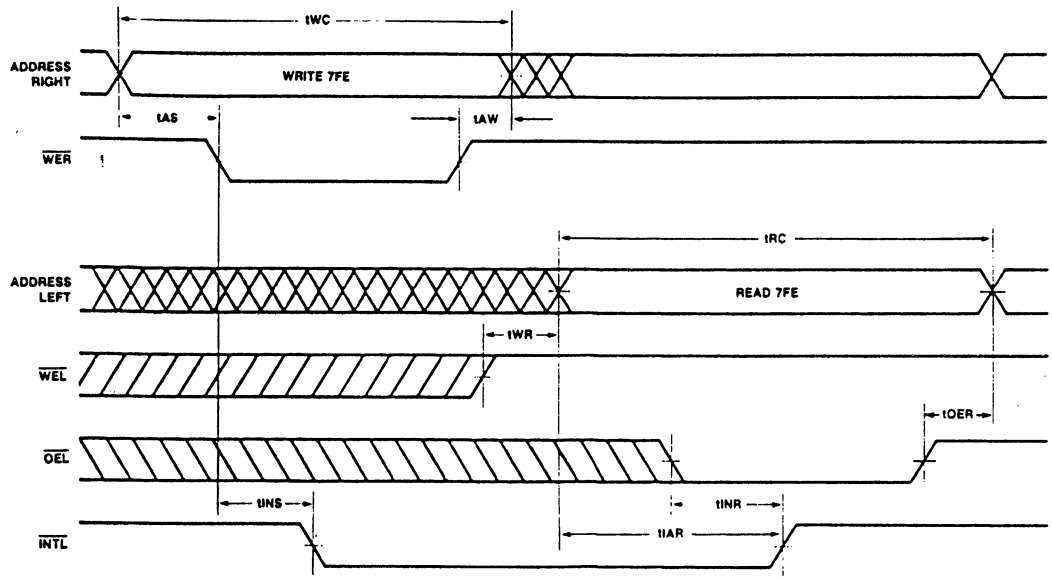
LEFT SIDE FLAGS RIGHT SIDE



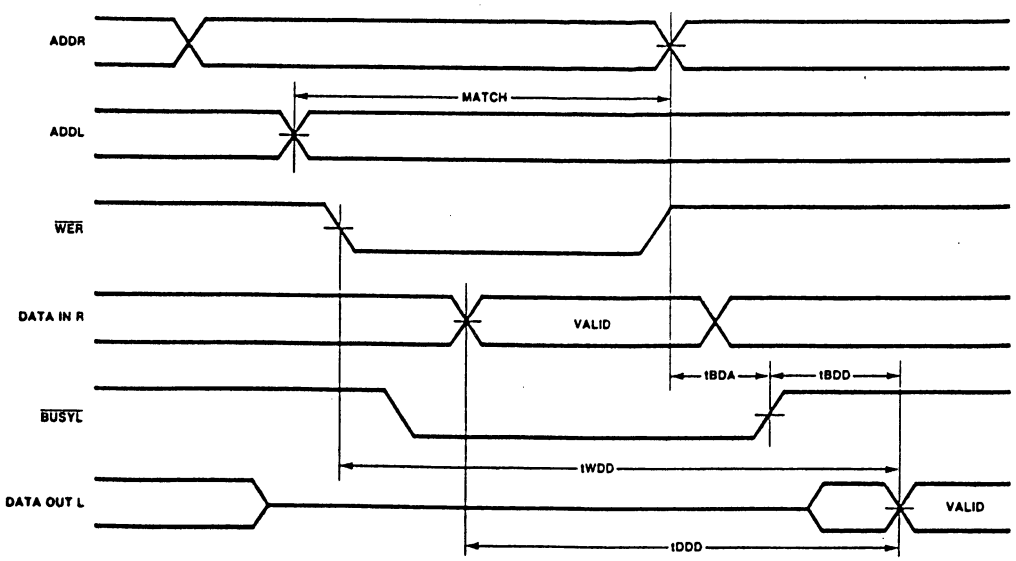
TIMING DIAGRAMS (Cont.)

INTERRUPT MODE, Note 1

RIGHT SIDE FLAGS LEFT SIDE

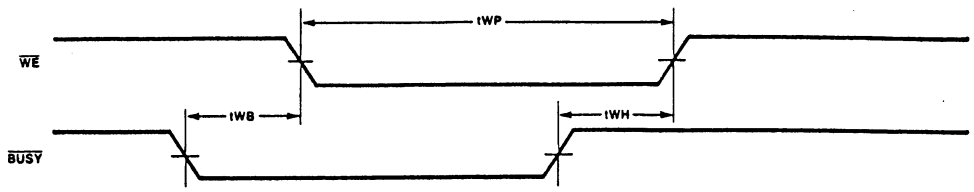


READ WITH BUSY



TIMING DIAGRAM (Cont.)

WRITE WITH $\overline{BUSY}$ SLAVE ONLY (7142A)



POWER DISTRIBUTION AND TRACE LINE TERMINATION CONSIDERATIONS

To achieve full compatibility with TTL-based devices, CMOS memories are typically designed to convert TTL input levels to the CMOS levels required for internal operation. Greater power efficiency is achieved, however, when an entire design takes advantage of the lower consumption capabilities of CMOS technology. When CMOS levels are used throughout a design and not only in the memory, lower current specifications can be achieved, resulting in a lower overall power requirement.

The operating margins of all devices on a board using very-high-speed memory can best be maintained by providing a quiet environment that is free of noise spikes, undershoot, and excessive ringing. Key elements in creating such an atmosphere are observing proper power distribution techniques and proper termination of TTL drive lines.

POWER DISTRIBUTION

A power distribution scheme that effectively maintains wide operating margins combines power trace layout with decoupling capacitor placement to minimize the series impedance in the decoupling path. This path runs from the power pin of a memory device through its decoupling capacitor to the ground pin.

The total impedance of this path is established by the power line impedance and the impedance of the capacitor itself. In practice, the capacitive effects of the decoupling path are minimal because of the very-high-frequency components of the current transients associated with memory operation. This makes the line inductance the dominant impedance factor.

The preferred technique for reducing power line impedance and improving the quality of VCC and ground is to use separate power and ground planes.

A somewhat-less-effective approach is to grid the power and ground traces. If this is done, the ground grid should extend to the TTL driver peripheral circuitry, providing a solid ground reference for the TTL drivers.

The decoupling capacitor, which provides energy for the high-frequency transients, should be placed as near the memory device as possible in order to have the shortest practical lead lengths. This capacitor should be of a low inductance type and, at a minimum, be 0.1 μF. For the greatest efficiency, it should be placed between the power supply and ground pins of each device.

Low-frequency current transients can be handled by larger tantalum capacitors placed near the memory board edge connector, where the power traces meet the backplane power distribution system. Such large capacitors provide bulk energy storage that prevents voltage drops caused by the long inductive path between the memory board and the power supply.

TRACE TERMINATION

On a memory board, trace lines have the appearance of shorted transmission lines to TTL-level driver signals. This can cause reflections of TTL signals propagating down the lines, particularly LOW-going signals. These reflections can be reduced or eliminated by proper line termination. Proper termination also reduces RFI emissions.

Trace line termination can be either series or parallel, although series termination is recommended. This type of termination has the advantage of drawing no dc current, and also requires the smallest number of components to implement. It simply calls for placing a series resistor in the signal line to dampen reflections. The resistor is placed at the output of the TTL driver, as close as possible to the driver package. The driver/termination combination should be placed close to the memory array to minimize lead length.

In most applications, a series resistor of between 10 ohms and 33 ohms is sufficient to dampen reflections. However, because the characteristic impedance of each layout is different, some experimentation may be necessary to determine the optimum value for a specific configuration.

SIGNAL FIDELITY

When the layout is complete and the power distribution and line termination requirements have been met, it is good procedure to verify signal fidelity by observation with a wide-band (300 MHz or faster) oscilloscope and probe.

Appendix D The Gateway Interface Card

The circuit of the interface is drawn in Fig. D.1. The name and function of each component are given:

* U2, U9 and U14 allow address signal transmission from the IBM I/O port to the Dual-port RAM.

* U1 allows data signal transmission from/to the IBM I/O port to/from the Dual-port RAM depending upon the logic level at the direction control (DIR) input. The DIR pin is connected to the '-MEMR' (memory read) of the IBM I/O ports. If the logic level at the input is low, in other words, the IBM system wants to read the data from the Dual-port RAM, then the signal transmission direction is from the Dual-port RAM to the IBM I/O port (shown in Fig. D.2). Otherwise, the transmission is in an opposite direction.

* U13 is the Dual-port RAM. It does not only buffer the internet packets in the gateway, but also provide the Priority Status Register to record the priority status of the packet currently processed by the gateway.

* U4, U5, U12 and U10 form an interrupt generator to generate an interrupt signal to the half-gateway in XLNET. Whenever the half-gateway in TMS-IBM Ring wants to ask the other one to handle an internet packet, the former one is asked to read the content of the specific address $DFFFF of the Dual-port RAM (accessed by the IBM I/O port). It is more important that the interrupt generator will be invoked after decoding to generate an interrupt signal to the other half-gateway in XLNET when the location is accessed.

* U8 and U7 form an interrupt generator. Its operation is similar to that mentioned above. But the specific address is $87FF and accessed by the XLNET nodal hardware. After decoding, the generator will issue an interrupt signal to the IBM I/O port.

* U3 and U12 are used to operate the U1. When the IBM system asks for the 'memory read' or 'memory write', the U1 is permitted to transfer the data signal between the two buses. Otherwise, the buses are effectively isolated.

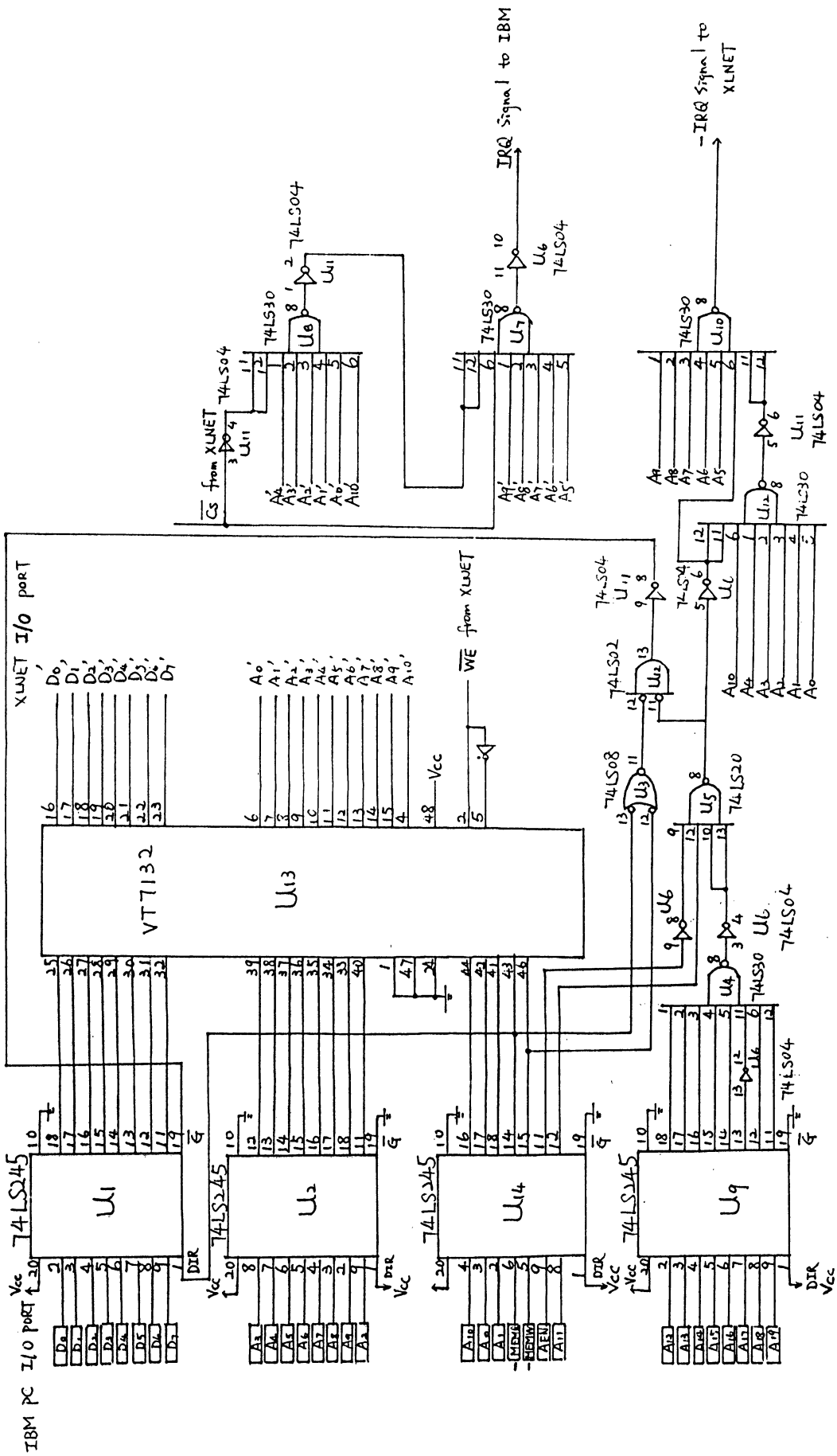


Fig. D.1 The Circuit of the Gateway Interface Card

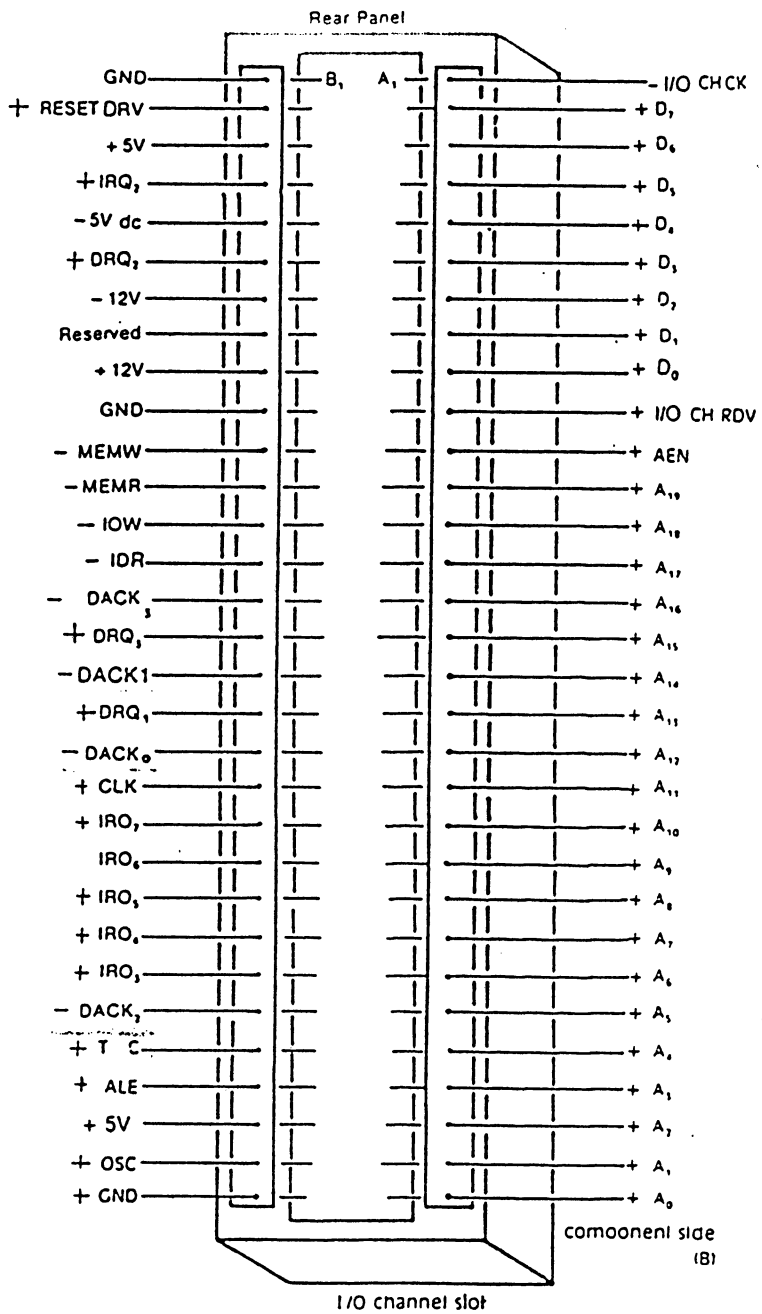


Fig. D.2 I/O Slots

Appendix E Information of the Memory System

E.1 Memory Address Space in IBM PC

The allocation of PC memory address space is shown in Fig. E.1. Since the ROM/jr cartridges is not installed in the PC memory, the addresses of the dual-port RAM are located in the range from $DF800 to $DFFFF.

E.2 Memory Card in XLNET Node

The addresses of the dual-port RAM accessed by XLNET is ranged from $8000 to $8700. The circuit diagrams of the memory card are shown in Figs. E.2 and E.3.

Figure 1-1. Allocation of PC Memory Address Space in 64K Blocks

Hex	****** 64K Block Memory Map ******
00000 0K	Vectors, data, DOS, Disk/Advanced BASIC
10000 64K	User program RAM, if filled *
20000 128K	User program RAM, if filled *
30000 192K	User program RAM, if filled *
40000 256K	User program RAM, if filled *
50000 320K	User program RAM, if filled *
60000 384K	User program RAM, if filled *
70000 448K	User program RAM, if filled *
80000 512K	User program RAM, if filled *
90000 576K	User program RAM, if filled *
A0000 640K	Future video reserved
B0000 704K	Mono/color video
C0000 768K	Future ROM / XT fixed disk ROM
D0000 832K	Future ROM / jr cartridges
E0000 896K	Future ROM / jr cartridges
F0000 960K	Tests, ROM BASIC, ROM BIOS
FFFFF 1024K	

* BASIC programs are limited to a 64K workspace

Fig. E.1 The Allocation of PC Memory Address Space in 64K Blocks

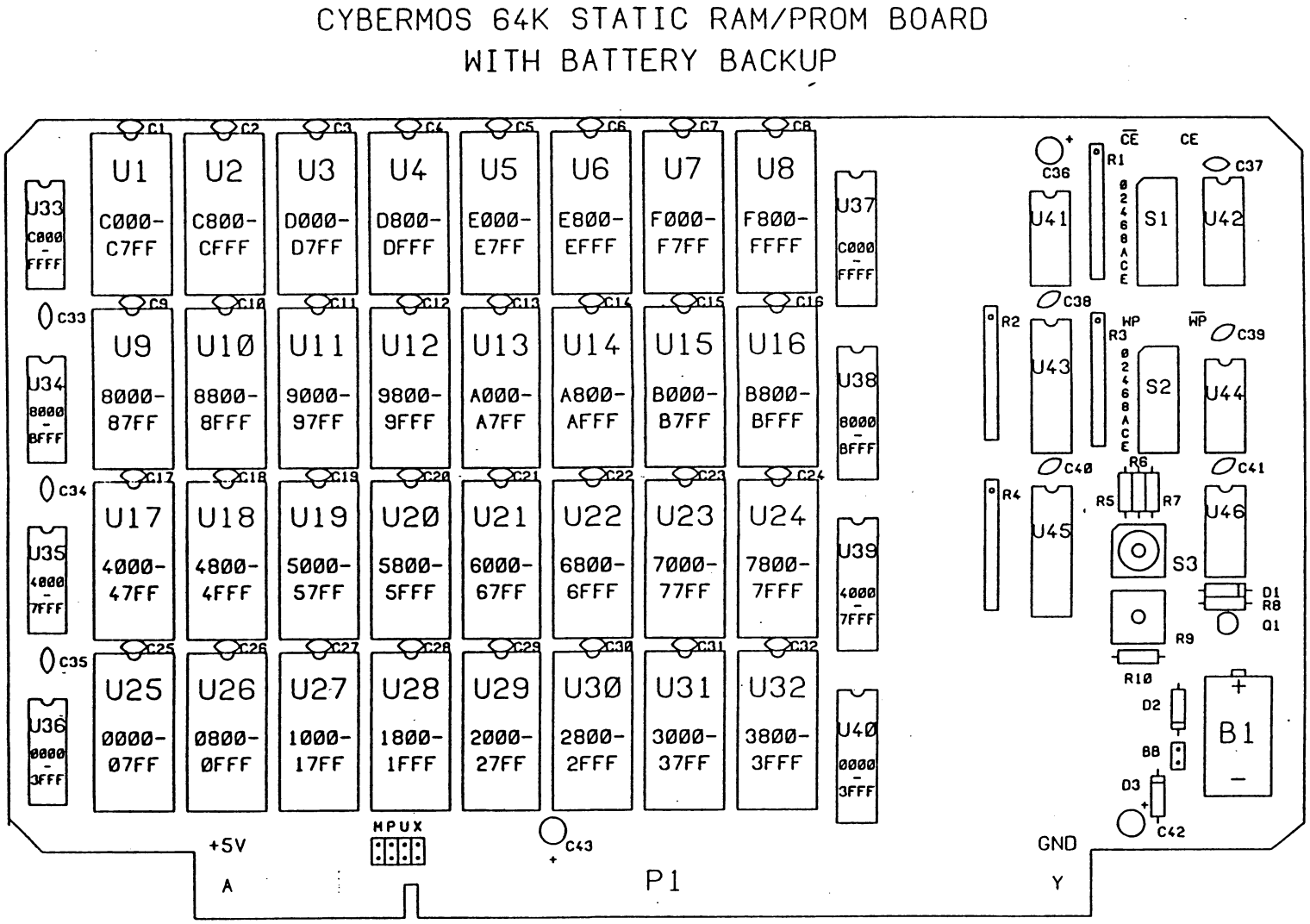


Fig. E.2 Diagram of the Memory Card of XLNET Node (1)

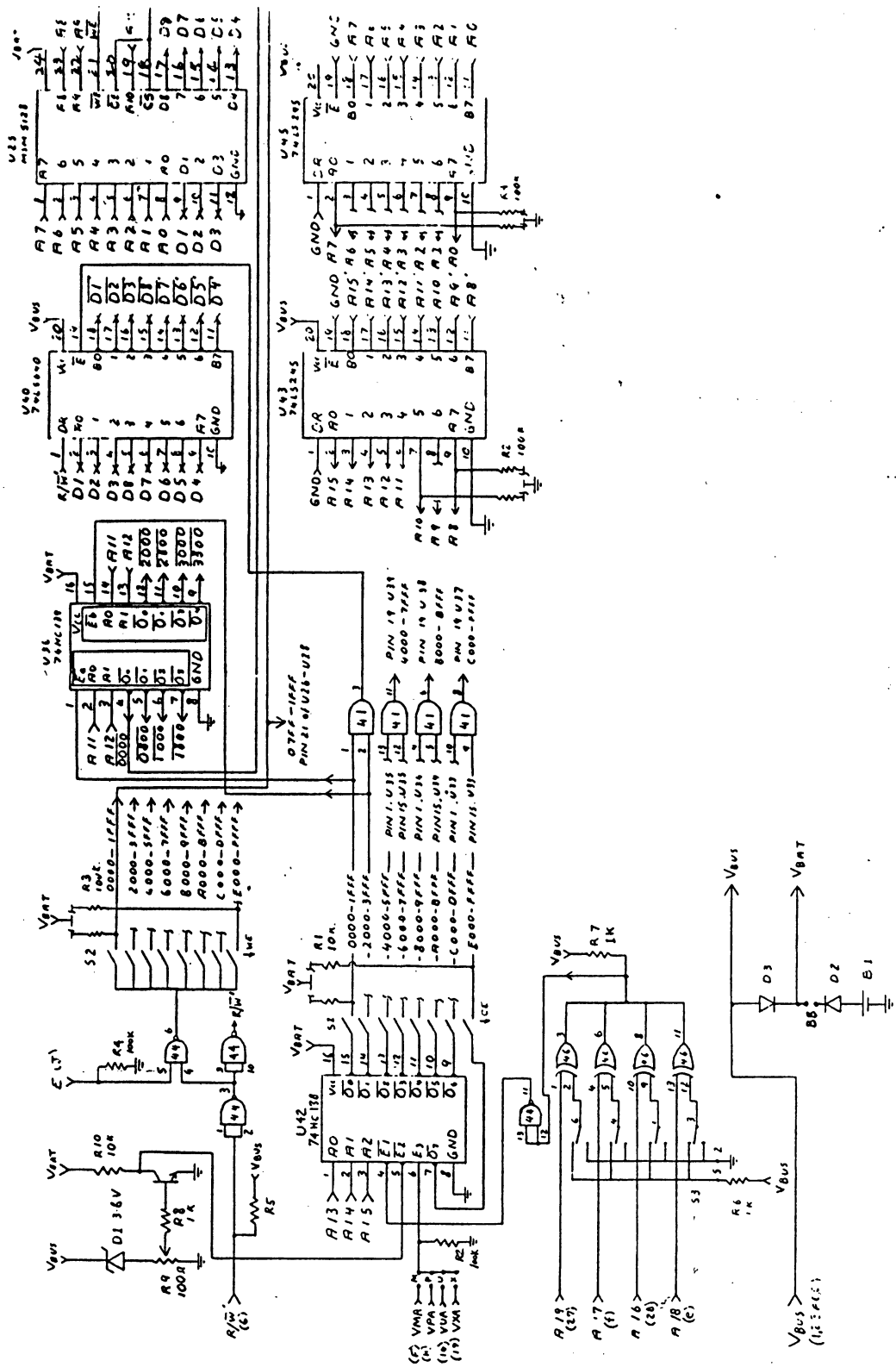


Fig. E.3 Diagram of the Memory Card of XLNET Node (2)

Appendix F Software Program of the Gateway

```
procedure Dtgate_layer(var mp1:message_pointer);{*TMS-IBM Ring to XLNET*}
var
    i,j:integer;
    temp:byte;
begin  (*Dtgate_layer*)
   with TDxlnetbf,mp1^ do
     begin
        if flag=1 then writeln('Attentiom please,the XLNET might be blocked !')
        else begin
             {* Begin frame reformat *}
              buf_status:=1;
              buf_link:=0;
              buf_sp:=2;
              buf_dp:=3;
              if message_length=0 then buf_length:=15
              else begin
                     buf_length:=message_length+15-1;
                   {* Begin user data transfer*}
                    for i:=1 to message_length-1 do
                        data_field[i-1]:=msg_string[i];
                   {* End user data transfer*}
                  end;
              buf_da:=RDA[5];
              buf_sa:=SA[5];
              buf_da_cirt:=DA_trans_cct;
              buf_sa_cirt:=SA_trans_cct;
              if SLA=1 then buf_sa:=buf_sa+128;
              if DLA=1 then buf_da:=buf_da+128;
              buf_fc:=64;
              buf_type:=100;
             {* Begin peer protocol conversion*}
              case trans_signal of
                   t_open_ind:  begin
                                  buf_trans:=call_req;
                                  buf_sess:=64;
                                end;
                   t_info_ind:  begin
                                  IBM_seq_no[SA_trans_cct]:=seq_no;
                                  buf_trans:=info_ind;
                                  buf_sess:=info_ind;
                                end;
                    t_close_ind:    buf_trans:=clr_req;
                    t_open_acc:     buf_trans:=confirm;
                    t_open_con_ind: buf_trans:=call_ans;
                    t_open_rej_ind: buf_trans:=call_rej;
                    t_confirm:   begin
                                    inc(XLN_seq_no[DA_trans_cct]);
                                    buf_trans:=tt_confirm;
                                 end;
                   end;(*End peer protocol conversion*)
                      {*End frame reformat*}
                   flag:=1;
                   i:=0;
                   repeat
                      xlnet:=1;
                   until (i=100) or (flag=0);
                   if flag=1 then begin
                      writeln('After 100 attempts,no response from XLNET.');
                      writeln('Please, CHECK !!!');
                                end;
          end;
       end;(*with*)
       FreeMsgBuf(mp1);
end;
```

```
procedure Dfgate_layer;(*XLNET to TMS-IBM Ring*)
var
    i:byte;
    OK:boolean;
    ADDR_TEMP:BLKs;
    temp1:logicBLK_type;
    mp:message_pointer;
begin  (*Dfgate_layer*)
        OK:=false;
        if  (FVxlnetbf.flag=1) then
            begin
                temp1:=FVxlnetbf;
                OK:=true;
            end
         else if  (FDxlnetbf.flag=1) then
            begin
                temp1:=FDxlnetbf;
                OK:=true;
            end;
          while OK=true do (* Begin frame format *)
            begin
                inline($FA);
                mp:=get_freemsgbuf;
                with mp^,temp1 do
                  begin
                    if buf_da>127 then begin
                       DA[5]:=buf_da-128;
                       DLA:=1;
                                          end
                    else begin
                           DA[5]:=buf_da;
                           DLA:=0;
                         end;
                    DA[0]:=$40;
                    if buf_sa>127 then begin
                       SA[5]:=buf_sa-128;
                       RDA[5]:=SA[5];
                       RDA[0]:=0;
                       SLA:=1;
                                          end
                     else begin
                       SA[5]:=buf_sa;
                       SLA:=0;
                       RDA[5]:=SA[5];
                       RDA[0]:=0;
                           end;
                     SA[0]:=$40;
                     for i:=1 to 4 do
                           begin
                             SA[i]:=0;
                             DA[i]:=0;
                             RDA[i]:=0;
                           end;
                      DA_trans_cct:=buf_da_cirt;
                      DA_sess_cct:=buf_da_cirt;
                      SA_trans_cct:=buf_sa_cirt;
                      SA_sess_cct:=buf_sa_cirt;
                      if (FDxlnetbf.flag=1) then
                         begin
                            FC_field:=$40;
                            AC_field:=$00;
                            LLC_type:=LLC_DATA;
                            message_type:=DATA;
                         end;

                      if buf_length=15 then message_length:=0
                      else begin
                              message_length:=buf_length-14;

                              (* Begin user data transfer *)

                              for i:=1 to message_length do
                                  msg_string[i]:=data_field[i-1];

                              (* End user data transfer*)

                           end;
```

```
                    msg_string[0]:=chr(message_length);
                    case buf_trans of  (* peer protocol conversion *)
                         8:      begin
                                    trans_signal:=t_open_ind;
                                    sess_signal:=s_open_ind;
                                 end;
                        12:      begin
                                    trans_signal:=t_info_ind;
                                    sess_signal:=s_info_ind;
                                    seq_no:=XLN_seq_no[SA_trans_cct];
                                 end;
                        13:         trans_signal:=t_close_ind;
                         6:         trans_signal:=t_open_acc;
                        10:      begin
                                    trans_signal:=t_open_con_ind;
                                    sess_signal:=s_open_con_ind;
                                 end;
                         9:      begin
                                    trans_signal:=t_open_rej_ind;
                                    sess_signal:=s_open_rej_ind;
                                 end;
                        22:      begin
                                    trans_signal:=t_confirm;
                                    seq_no:=IBM_seq_no[DA_trans_cct];
                                 end
                    end;
                            (* End peer protocol conversion *)

                            (* End frame reformat *)

                    if (FVxlnetbf.flag=1) then
                       begin
                          FVxlnetbf.flag:=0;
                          if (DA[5]=OPEN_PARA_LIST.node_address[5]) then
                          send_msg(network_MB,LLC_MB,mp)
                          else
                            begin
                               ADDR_TEMP:=SA;
                               SA:=OPEN_PARA_LIST.node_address;
                               RDA:=ADDR_TEMP;
                               send_msg(LLC_MB,GATE_MB,mp);
                            end
                       end
                    else
                       begin
                          FDxlnetbf.flag:=0;
                          if (DA[5]=OPEN_PARA_LIST.node_address[5]) then
                          send_msg(network_MB,LLC_MB,mp)
                          else
                            begin
                               ADDR_TEMP:=SA;
                               SA:=OPEN_PARA_LIST.node_address;
                               RDA:=ADDR_TEMP;
                               send_msg(LLC_MB,GATE_MB,mp);
                            end
                       end;
                  end;(*with*)
               end;(*while*)
               inline($FB);
               inline($80/$20/$E6/$20);
end;

procedure set_Dfgate_layer_vector;

 (*Set the interrupt level*)

 var
     Dfgate_layer_vector:^integer;
 begin
     Old_TypeF_vector:=GetIOHandler($F);
     Dfgate_layer_vector:=NewIOProcess(Ofs(Dfgate_layer),1000);
     IOAttach($F,Dfgate_layer_vector);
     port[$21]:=port[$21] and $7F;
     port[$20]:=$20;
     port[$A0]:=$80;
end;
```

REFERENCES

[1] ISO International Standard 7498, Information Processing Systems--Open Systems Interconnection--Basic Reference Model, International Organisation for Standardisation, Geneva, Oct., 15, 1984.

[2] G.J Anido and A.E. Karbowiak, 'Research and Development of a Novel Local Area Network--XLNET, Stage 1, Sec. 1', Unisearch Ltd, Jan., 1985.

[3] IEEE Project 802, Draft IEEE Standards 802.4: Token Bus Access Method and Physical Layer Specifications. Dec., 1982.

[4] ANSI/IEEE Standard 802.4-1985. Token-Passing Bus Access Method and Physical Layer Specifications, 1985.

[5] IEEE Project 802, Local Network Standard, Draft Standard 802.2: Logical Link Control. Nov., 1982.

[6] ANSI/IEEE Standard 802.2-1985. Logical Link Control, 1985.

[7] ISO, Draft Proposal 8473: Information Processing systems Data Communications Protocol for providing the Connectionless Network Service, International Organisation for Standardisation, Geneva 1983.

[8] Open Systems Interconnection: protocol to provide the connectionless mode network service (ISO/DIS 8473) incorporating the connectionless mode service addendum. August, 1985.

[9] ISO, draft International Standard 8073: Information Processing Systems Open System Interconnection for Transport Protocol Specifications, International Organisation for Standardisation, Geneva 1983.

[10] Information Processing Systems--Open Systems Interconnection--Transpport Protocol Specifications, ISO 8073, 1986.

[11] Bux W. A Local Area Communication Network Based on a Reliable Token Ring System, Local Comp. Nets, North-Holland Publishing Co. 1982.

[12] Keller H.J. Meyr H. and Mueller H.R., Transmission Design Criteria for a Synchronous Token Ring, IEEE Journ. Select Areas in Comms, Vol, SAC-1, No.5, Nov., 1983, p.721-733.

[13] Mark J. W., "Welnet": A High Performance Integrated Service Local Area Network, IEEE Globecom 1985, New Orleans, Dec., 1985, p.472-477.

[14] Gary J. Anido, " The Design, Analysis and Implementation of a Fully Distributed Local Area Network for Integrated Voice and Data Communications". PHD Thesis, Chapter 3, University of N.S.W., Australia, Nov. 1987.

[15] Minoli, D. "Issues in Packet Voice Communications", Pro, IEE 126, Aug., 1979, p.729.

[16] Minoli, D. " Packetised Networks-1", Australian Electronics Engineering. Apr., 1979, p.38.

[17] G.J. Anido and A.E. Karbowiak, "Research and Development of a Novel Local Area Network--XLNET" stage 1, section 3, Unisearch Ltd, Jan., 1985.

[18] G.J. Anido and A.E. Karbowiak, "Research and Development of a Novel Local Network--XLNET" stage 1, Section 8, Unisearch Ltd, Jan., 1985.

[19] Gary J. Anido, "The Design, Analysis and Implementation of a Fully Distributed Local Area Network for integrated Voice and Data Communications". PHD Thesis, Appendix D, University of N.S.W., Australia, Nov., 1987.

[20] "TMS-380 Adapter Chipset User's Guide", Texas Instruments.

[21] Ng T.L., "Integration of Voice and Data Communications in the IBM Token Ring", University of New South Wales, Nov.1987.

[22] IEEE Standard 802.5-1985 (ISO DIS 8802/5), Token Ring Access Method and Physical Layer Specifications.

[23] Richard M. Foard, "Multitasking Methods", PC Tech Journal, Mar. 1985 P.51.

[24] Bux, W.,"Local Area Subnetworks: A Performance Comparison.", IEEE Trans Comm., Comm-29, 10 Oct, 1981, P.1465.

[25] G.J.Anido and A.E. Karbowiak, "Research and development of a Novel Local Area Network-XLNET", Stage 1, Section 5, Unisearch Ltd, Jan., 1985.

[26] J.A. Berntsen, "MAC Layer Interconnection of IEEE 802 Local Area Networks". Computer Networks and ISDN systems 10 North-Holland Publishing, 1985. p.259-273.

[27] N.Linge, E. Ball. "A Bridge Protocol for Creating a Spanning Tree topology within an IEEE 802 Extended LAN Environment". Computer Networks and ISDN systems 13 North-Holland Publishing, 1987, p.323-332.

[28] B.Hawe, A.Kirby, "Transparent Interconnection of Local Networks with Bridges". Advances in Local Area Networks. Edit by K.Kummerle, IEEE Press, 1988, p.482-495.

[29] Tanenbaum, A.S. "Computer Networks", Prentice-Hall, Englewood Cliffs, NJ, 1981.

[30] Rybczynski, A., "X.25 Interface and End-to-end Virtual Circuit Service Characteristics". IEEE Trans. 1980, p.500-509.

[31] Folts, H., "X.25 Transaction Oriented Features--datagram and Fast select", IEEE Trans, 1980, p.496-499.

[32] Recommendations X.25. Blue Book, FASCICLE VIII, 2 CCITT, 1988.

[33] Recommendations X.75. Blue Book, FASCICLE VIII, 3 CCITT, 1988.

[34] Hinden, R., Harverty, J., Sheltzer, A., "The DARPA Internet: interconnecting heterogeneous networks with gateways". IEEE Computer 1983, p.38-49.

[35] Douglas Comer, "Internetworking with TCP/IP", Chapter 10, Prentice Hall, 1988.

[36] Douglas Comer, "Internetworking with TCP/IP", Chapter 7, Prentice Hall, 1988.

[37] Green Jr., P.E., "Protocol Conversion", IEEE Trans. Commun., 1986. Vol. COM-34, No.3, p.257-268.

[38] Gien, M. and Zimmerman, H., "Design principles for network interconnection", Proc. 1979. Sixth Data Commun. Symp., p.109-120.

[39] Inge Groenbaek, "Conversion between the TCP and ISO Transport Protocols as a Method of Achieving interoperability Between Data Communications System", 1986, IEEE J. on Selected Areas in Commun., Vol. SAC-4, No.3, p.288-296.

[40] Zoline, K.O. "An approach for Interconnecting SNA and XNS Networks, Comp. Commun. 1985. Rev., Vol. 15, No.4, p.184-198.

[41] Meister, B.W., Janson, P.A. and Svobodova, L. "Connection-oriented versus Connectionless Protocols: A Performance Study, IEEE Trans. Comput., 1985. Vol. C-34, No.12, p.1164-1172.

[42] Dalal, Y.K., Printis, R.S., "48 bit absolute Internet and Ethernet host numbers" Proc. 7th. Data Comms. Symp. ACM Computer Comms. review, Vol. 11, No.4, Oct., 1981, p.240-247.

[43] G.A. Deaton, Jr., and R.O.Hippert, "X.25 and related recommendations in IBM products", IBM Syst. J., vol 22, nos. 1/2, p.11-29, 1983.

[44] Jhitti Chiarawongse, Mandyam M. Srinivasan and Toby J. Teorey, "The performance analysis of a large internetconnected network by decomposition techniques." IEEE Network, July 1988-Vol. 2, no. 4, p.19.

[45] Mischa Schwartz, "Telecommunication Networks Protocols, Modeling and Analysis." Addison-Weslay, 1987, chap 2, p.21-70.

[46] Thomas Fong, "Integration of Voice and Data in the IBM Token Ring.", Bachelor Thesis, Chapter 6, University of NSW, Australia, Oct., 1989.