NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Performance and Fairness Evaluation of the IEEE 802.6 DQDB MAN Supporting Different Communities of Interest

by
Wade Williams

A THESIS
submitted to the School of Graduate Studies and Research
in partial fulfillment of the requirements
for the degree of
MASTER OF APPLIED SCIENCE
in
Electrical Engineering

Ottawa-Carleton Institute for Electrical Engineering
Department of Electrical Engineering
Faculty of Engineering
University of Ottawa

ISBN 0-315-75059-6

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

I hereby declare that I am the sole author of this thesis.

I authorize the University of Ottawa to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Wade Williams

I further authorize the University of Ottawa to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Wade Williams

The University of Ottawa requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

For Meg and John

Contents

List of Figures

List of Tables

Abstract

Metropolitain Area Networks (MANs) are receiving increased interest from the communications industry because of their large bandwidth and ability to support both asynchronous and isochronous data types. The DQDB (Distributed Queue Dual Bus) architecture has been adopted as the IEEE 802.6 MAN standard. The standardization process for the DQDB MAN has been both a lengthy and complex process. The DQDB standard has undergone several major modifications to improve system fairness in dividing the asynchronous bandwidth among active users. This thesis outlines the historical requirement for a MAN and also traces the steps of the IEEE 802.6 standardization process. A detailed performance evaluation of the current version of the DQDB standard to support community of interest based traffic is given. The study shows that the standard in its present form is still unable to equally divide its asynchronous bandwidth among all stations. The problem is caused by two main factors. First, the station's position along the DQDB bus will greatly influence the amount of bandwidth it is able to obtain. The second major reason for unfair bandwidth allocation stems from the dual bus architecure of DQDB. Not all stations enjoy the privilege of using both DQDB buses to send data traffic. This will severely restrict their performance. This situation of being unable to use both DQDB buses will be quite common for DQDB based Public MANs as they service multiple communities of interest. This thesis proposes a solution called Variable Bandwidth Balancing(V_BWB) to correct the unfairness of DQDB. V_BWB is based on the concept of easy access during light network loads and tightly controlled access during peak network loading.

Acknowledgements

I would like to extend my appreciation to my thesis supervisor, Dr. Nicolas D. Georganas for suggesting and guiding this line of research. His encouragement, interest, and guidance has made this document possible.

I would like to thank the Department of National Defence for providing the opportunity for me to obtain my masters degree.

The friendship and help of all my colleagues is also appreciated. I would especially like to express my thanks to Dr. Luis Orozco-Barbosa for his effort in helping me solve many of the obstacles encountered during the research for this thesis.

The invaluable help of the University of Ottawa support staff is also acknowledged.

I would like to thank Mom, Dad, Kelly, Rhonda and Todd for their constant support throughout my academic training during the past 10 years.

I would especially like to thank my wife, Karen, for her constant encourangement and support throughout this endeavor. Without her help I am certain that this thesis would never have been completed. Finally, I would like to thank my son, John, for bringing me back down to earth when I was lost deep in concentration.

Acronyms

ACF	Access Control Field
ANSI	American National Standards Institute
ATM	Asynchronous Transfer Mode
AU	Access Unit
BOM	Beginning of MAC PDU
B-ISDN	Broadband Integrated Services Digital Network
BWB	Bandwidth Balancing
BWB_MOD	Bandwidth Balancing Modulus
BWB_CTR	Bandwidth Balancing Counter
CATV	Cable Television
CCITT	Comité Consultatif International Télégraphique et Téléphonique
COM	Continuation of MAC PDU
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CRC	Cycle Redundancy Checks
DMPDU	Derived MAC PDU
DQDB	Distributed Queue Dual Buses
DSAP	Destination Service Access Point
D_V_BWB	Dynamic Variable Bandwidth Balancing
EOM	End of MAC PDU
FCS	Frame Check Sequence
FIFO	First In First Out
HCS	Header Checksum
HDLC	High Level Data Link Control
IEEE	Institute of Electrical and Electronics Engineers
IMPDU	Impliciet Mac Protocol Data Unit
ISDN	Integrated Services Digital Network
ISO	International Organisation for Standardization
ISU	Isochronous Service User
IVD	Integrated Voice/Data
IVDWS	Integrated Voice/Data Workstation
I-frame	Information frame
LAN	Local Area Network

LLC	Logical Link Control
MAC	Media Access Control
MAC PDU	Media Access Control Protocol Data Unit
MAN	Metroplitan Area Network
MCP	MAC Convergence Protocol
MSAP	MAC Service Access Point
MSDU	MAC Service Data Unit
N(R)	Receive Sequence Number
N(S)	Send Sequence Number
OSI	Open System Interconnection
PA	Pre-Arbitrated
PCI	Protocol Control Information
PDU	Protocol Data Unit
PSR BIT	Previous Seqment Read Bit
QA	Queue Arbitrated
QNAP	Queue Network Analysis Package
QPSX	Queue Packet and Synchronous Circuit Exchange
REQ	Request bit
REJ-frame	Reject supervisory frame
RNR	Receive Not Ready
RR-frame	Ready to Receive supervisory frame
SAP	Service Access Point
SH	Service Header
SONET	Synchronous Optical Network
SSAP	Sender Service Access Point
SSM	Single Segment MAC PDU
S-frame	Supervisory frame
THT	Token Holding Timer
TRT	Token Rotation Timer
TRT	Token Rotation Timer
U-Frame	Unumbered Frame
V_BWB	Variable Bandwidth Balancing
V_BWB_MOD	Variable Bandwidth Balancing Modulus

Chapter 1

Introduction

1.1 An Overview of Metropolitan Area Networks

Local area networks(LANs) have been used extensively in the past to allow a group of users to communicate and share common data and peripheral equipment. This sharing occurs over a common communications medium. To date the most common forms of LANs are Ethernets and Token rings.

The cost savings from communicating and sharing resources over LANs has pushed the networking industry to produce faster systems with a wider bandwidth. Local area networks are currently available which operate in the 16Mbps range over several Kms. It is thought that these LANs will not provide the required performance to satisfy new applications such as high speed voice and data transmissions originating from a multimedia workstation. Organizations which are using LANs are attempting to allow more users over wider distances to communicate by interconnecting these LANs via bridges and gateways.

It is clear that a new high capacity network must be made available to organizations to interconnect these LANs. This movement towards interconnecting LANs suggests that the geographical area covered by this new interconnection network will be much larger than is currently covered by LANs. The growing demand for larger bandwidths and wider geographical coverage has pushed the industry beyond the capabilities of LANs. Standard token rings

will have too great of a token rotation time and ethernets will have too many packet collisions due to large propagation delays, to efficiently act as the interconnection backbone for a group of LANs. This paves the way for a new "concept" of networking with capabilities which will fall between those of LANs and those of very high speed wide area networks(WANs) used by the carriers.

1.2 The IEEE 802.6 Standard

In 1981 IEEE recognized a need for "in between" networks which bridge the gap between privately owned and operated LANs, and the public carriers' very high speed and large distance WANs. IEEE formed the 802.6 Metropolitan area network (MAN) committee to standardize this new network which has one foot in the LAN world and one foot in the public carriers WAN domain.

IEEE set up some early guidelines to help with the standardization process. The IEEE MAN will be required to support both synchronous (voice and video) and asynchronous (packet) type traffic. The IEEE standards committee decided that the MAN architecture will have to operate efficiently over a diameter of about 50 km, which is the size of a typical city. The committee first limited the speed of the MAN to 20 Mbps or less but this restriction was later removed. The network capacity will be dictated by the capabilities of the communications media. It is commonly believed that 43 Mbps (DS3 based) and 155 Mbps (SONET based) seem to be prime candidates for bit rates to support most current applications.

1.3 Perceived Uses of MAN'S - Private and public

The IEEE 802.6 standardization committee is attempting to define a standard for the MAN architecture. This standard is required because the new network will possibly interconnect equipment belonging to many different organizations, each with a different brand of computer equipment and communications protocols. There are two principal MAN configurations. First

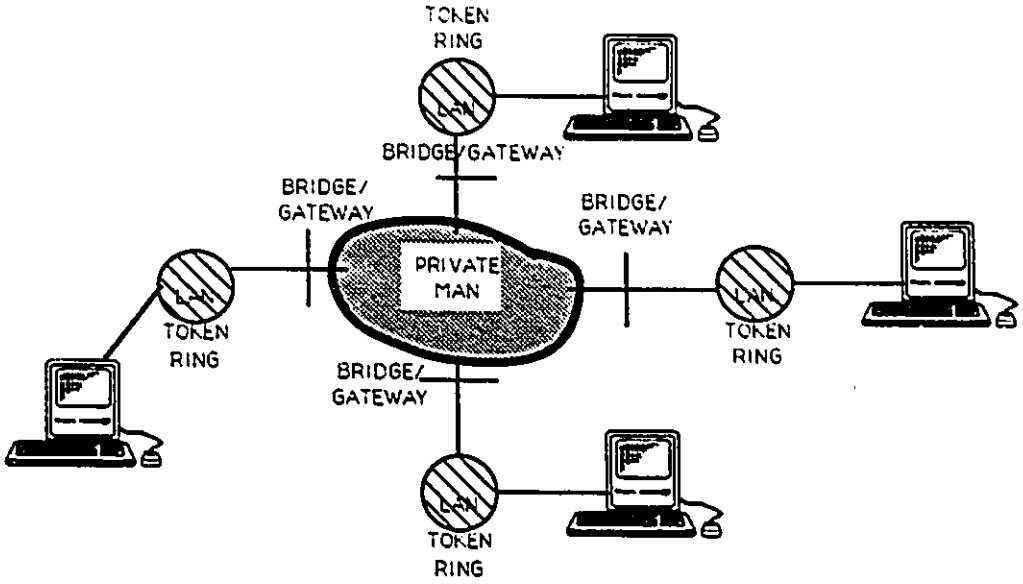


Figure 1.1: IEEE 802.6 Private MAN used to inter-connect 4 Token Ring LANs Via Bridges.

is the private MAN which is owned and operated by a single organization. Second is the public MAN, which is owned and controlled by the carriers. The public users will lease bandwidth from the public MAN to service their requirements.

1.3.1 Private MAN

Figure 1.1 shows how a private MAN would fit into a typical large organization's communication system. We can see that the private organization's LANs could be interconnected using the IEEE MAN. The private MAN could also be used to connect high speed workstations which require more bandwidth than is typically available on a LAN.These high speed workstations could be multimedia based, thus requiring both the synchronous and asynchronous capabilities.The synchronous component of the multimedia transmissions could be voice and/or video and the asynchronous component could comprise of data files or images. The synchronous capabilities of the MAN could also provide a private telephone system internal to the organization owning the private MAN.

The LANs used by a private company would most likely be compatible with each other (eg. all LANs are IEEE 802.5 Token Rings) and thus the required bridges to the MANs would be quite simple and usually no protocol conversion would be necessary between user LANs. The private MAN would most likely reside on the user premises and thus not interfere with the "Public Right Of Way" issues when laying cable in public areas as it is very difficult for private companies to gain access to the carriers public cable domain.

We can see that this private MAN can be looked at as a large and very powerful LAN which is wholly controlled by the private organization and could be used to completely integrate all the communication facilities of the organization. The high cost of such a system would strictly limit the number of organizations which would want or need a private MAN.

1.3.2 Public MAN With Many Communities Of Interest

I suspect the most popular use for the IEEE MAN will be the Public MAN which is owned and operated by the public carriers. Bandwidth will be leased to desiring organizations. Figure 1.2 displays a public MAN supporting several organizations (community of interests) with a single public MAN. The leasing organizations would not have to incur the large cost of installing the MAN, nor would they have to worry about the "Public Right Of Way" problems of laying cable in public areas.

The types of traffic travelling across a Public MAN would be similar to that of the private MAN, however, the traffic statistics would vary greatly from those of the private MAN. For example, the traffic generated within a single community of interest will likely remain within the sending community and will not be sent to others on the MAN. A major bank which is sharing a public MAN with a university will exchange little or no traffic. Thus each will have a great deal of intra-community traffic and very little inter-community traffic on the MAN.

The public MAN will support many types of LANs with different hardware platforms and different communication protocols. The carriers are strongly supporting the concept of

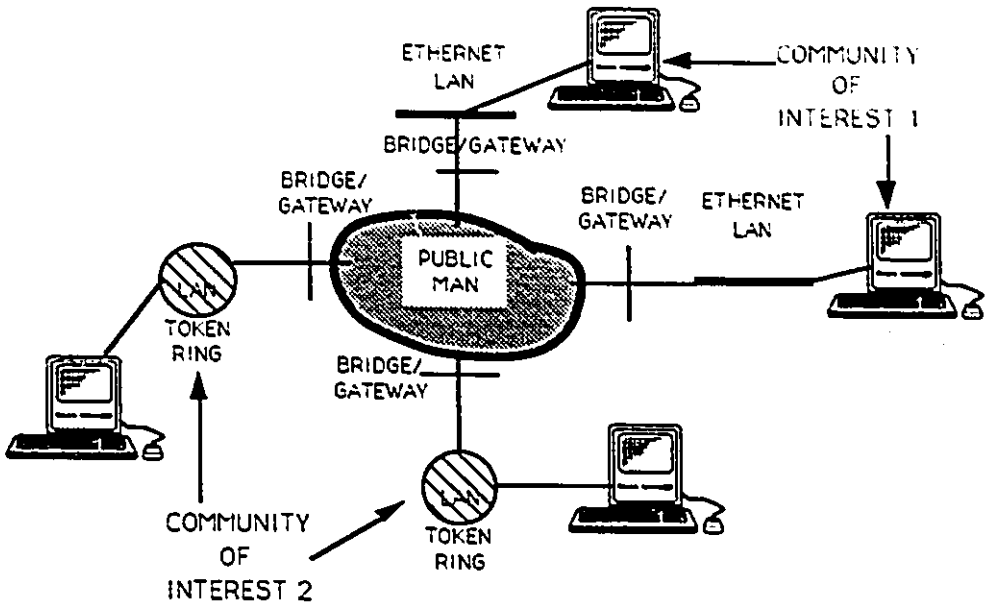


Figure 1.2: IEEE 802.6 MAN configured as a public network supporting two separate communities of interest (one Token Ring Based and the other Ethernet based).

a single MAN standard to keep the economies of scale reduced by only producing a single hardware chip set and one version of the protocol software to run the MAN[MOL88A].

The proposed IEEE MAN architecture will also be acting as a traffic collector for the Asynchronous Transfer Mode (ATM) networks of the future. ATM is the transport protocol which is presently proposed by the CCITT (Comité Consultatif International Télégraphique et Téléphonique) Study Group XVIII [CCI88]. This group is responsible for standardizing the B-ISDN (Broadband Integrated Services Digital Network).The slot size for the IEEE MAN has been made compatible to the proposed ATM cell size. This traffic collector function will fall within the domain of the public MAN[MAT89, RYD89]. Figure 1.3 shows how an ATM based WAN of the future will include both private and public MANs as well as the ATM switches. It is clear that the IEEE MAN architecture will play a vital role in future high speed networks and thus the IEEE 802.6 MAN will be the focus of this thesis.

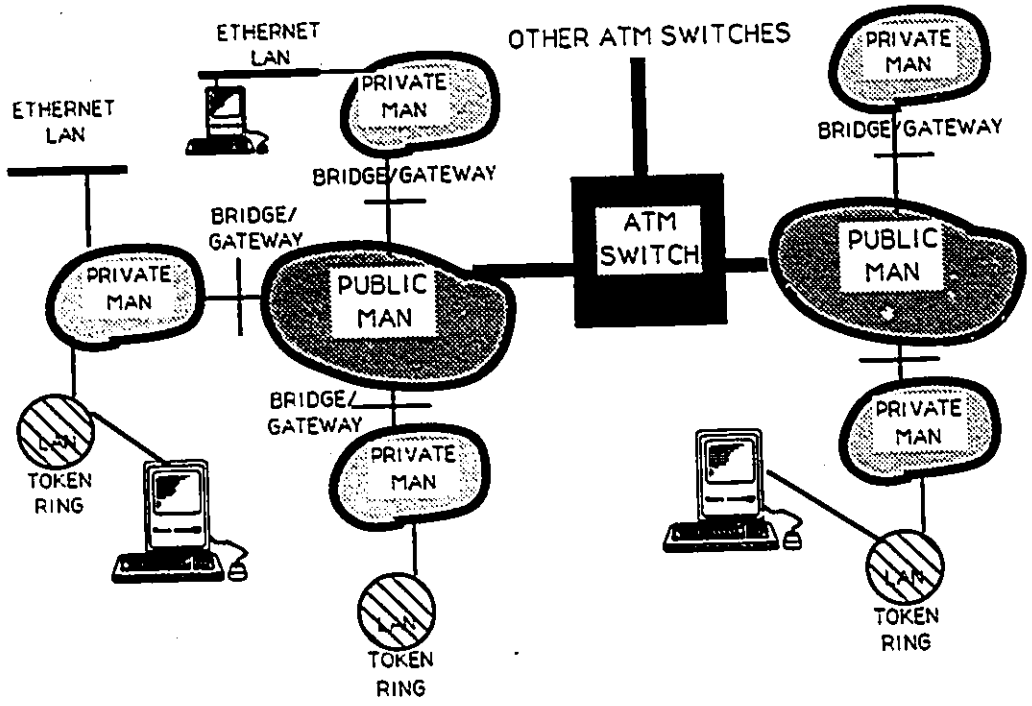


Figure 1.3: IEEE MAN incorporated within an ATM network.

1.4 Outline of the thesis

This thesis will present a detailed study of the IEEE 802.6 MAN standard. The study will focus on the MAN's ability to support the requirements of multiple users and more importantly to evenly divide the MAN resources between all active stations on the network. Two major areas are investigated during the study: first the efficiency and fairness of the IEEE MAN to support a 3 token ring based community of interest which uses the IEEE 802.2 Logical Link Protocol (LLC) as the end-to-end protocol. The second major area was to expand this study from a single community of interest to a more general, multiple community of interest based architecture. In order to evaluate the performance of the MAN to support these applications, queuing models have been developed using the QNAP2 (Queue Network Analysis Package) software package[QNA84] which is briefly described in appendix A. QNAP2 is used for queuing system modelling and performance evaluation. It incorporates many of the known analytical solutions of queuing networks, and also a discrete-event simulator which has been used for the performance evaluations presented in this thesis.

This thesis is composed of six chapters. Chapter one just presented an overview of the Metropolitan Area Network. It also discussed the IEEE 802.6 committee's work in the standardization process and gave an overview as to where the MAN architecture will be used within private, public and ATM based networks.

Chapter 2 will give the MAN historical progression, as to where it fits relative to the other IEEE 802 standards. The goals and objectives of the 802.6 committee will be outlined as will the functional requirements of a MAN network.

Chapter 3 will describe the IEEE 802.6 standard in detail. The first proposal (QPSX) for the standard will be discussed as will the network architecture and timing requirements. The unfairness of the early versions of the standards will be detailed and finally a performance evaluation of the current version of the standard will be given.

Chapter 4 will present a detailed study of the standard to act as the high speed backbone

network to interconnect a three token ring based communication system which uses LLC as its end-to-end protocol. Particular importance will be focused on the standards ability to properly divide its asynchronous bandwidth between all active stations. This system can be thought of as a private MAN because all three LANs attempt to communicate with each other to form a single community of interest using a MAN. Unfairness was observed with the current version of the standard and this unfairness was overcome using a modification to the standard which more evenly divides the MAN's bandwidth. This modification to the standard will be discussed in detail, as it provides a tool for correcting the current unfairness within the standard.

Chapter 5 will extend the study performed in chapter 4 by increasing the number of communities of interest which are supported by the MAN. This will effectively produce a public MAN with several communities of interest. As in chapter 4, unfairness will be observed and the modification introduced in chapter 4 will be used as the basis to achieve optimal MAN system performance. This optimal performance approach effectively eliminates the unfairness observed with the current version of IEEE MAN standard.

Finally, chapter 6 will review the interesting observations and conclusions made during the research and will focus particular attention to the fact that the current version of the MAN standard is still unfair in bandwidth distribution but the situation is correctable if a modification developed in this thesis is made to the current version of the IEEE MAN standard. Furthur, chapter 6 will suggest possible future research.

In summary, the contributions of this thesis are:

1. The determination that the IEEE 802.6 DQDB MAN standard as it now stands is unfair in asynchronous bandwidth sharing between active stations. It will be shown that this problem is extreme when public MANs support multiple communities of interest.

2. The unfairness was found to stem from both the station's position on each bus and also how well each station splits its traffic between the two DQDB buses.

3. To correct the unfairness in bandwidth sharing a Variable bandwidth balancing technique was developed to more evenly divide bandwidth. This type of modification to the current DQDB standard will be essential to provide the network managers the required flexibility to efficiently control their MANs.

Chapter 2

MAN Historical Progression

2.1 IEEE 802 Family of Standards

When studying the history the IEEE 802.6 MAN we must first look at the OSI layered architecture for computer communications to determine where the MAN will fit into this internationally defined standard for heterogeneous computer communications. We are particularly interested in determining what need there is for a MAN architecture within the OSI framework. The information gained from determining the need for a MAN architecture will lead us to defining the functional requirements the MAN must take on to meet these needs.

2.1.1 OSI Model

The International Organization for Standards (ISO) created the Open Systems Interconnection (OSI) subcommittee in 1977 to standardize computer network architectures[ZIMS0]. This standardized architecture would define the basic building blocks of a layered architecture used to create computer networks. The OSI committee approved the final 7 layer standard in 1983[ZIMS3]. Figure 2.1 displays the 7 layers of the OSI model.

The application, session, and transport layers provide the upper layer host functions which generate data and monitor the end-to-end communications between users. The lower layers, Network, datalink and physical layers, provide the network services required to move the data

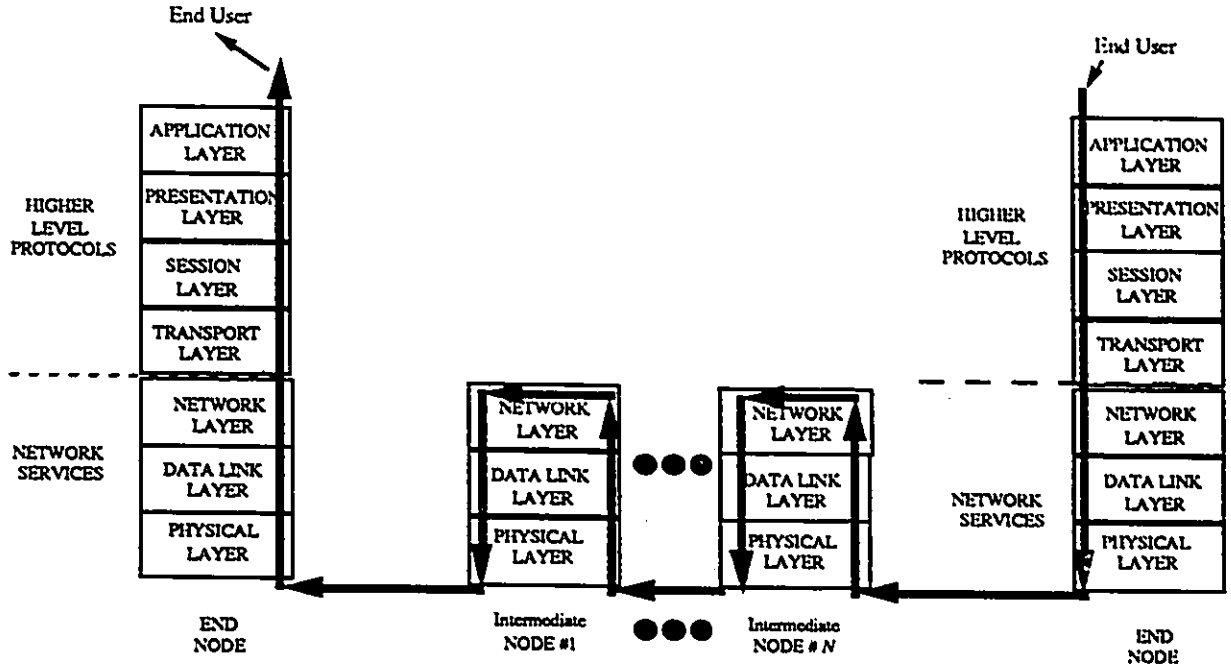


Figure 2.1: OSI Reference Model.

between end users. Each layer is intended to provide services to the layer directly above itself and use services of the layer directly below.

Protocol Data Units (PDUs) are the principal data blocks which are exchanged between peer entities during communications. Peer entities are simply similar layers within the OSI stack situated on opposite ends of a communication link. All layers set up virtual links with their peer entities at communication setup time and keep this virtual link active until communications are disconnected.

The PDUs of each layer are created by the addition of its own protocol control information (PCI) to the data units received from the layer above. The application layer first creates the data unit which is to be sent to its peer application at the far end of the communication link. The application layer will pass its PDU to the presentation layer where it will be encapsulated by the presentation layer control information. This control information is referred to as the service header(SH). This process of adding the SH of each layer to the data unit will continue

down through the OSI stack(see Fig. 2.2). At the destination station the SHs of each layer will be stripped as the data unit is passed up the stack. The SHs are used to validate the data and to determine to which virtual circuit it must be forwarded.

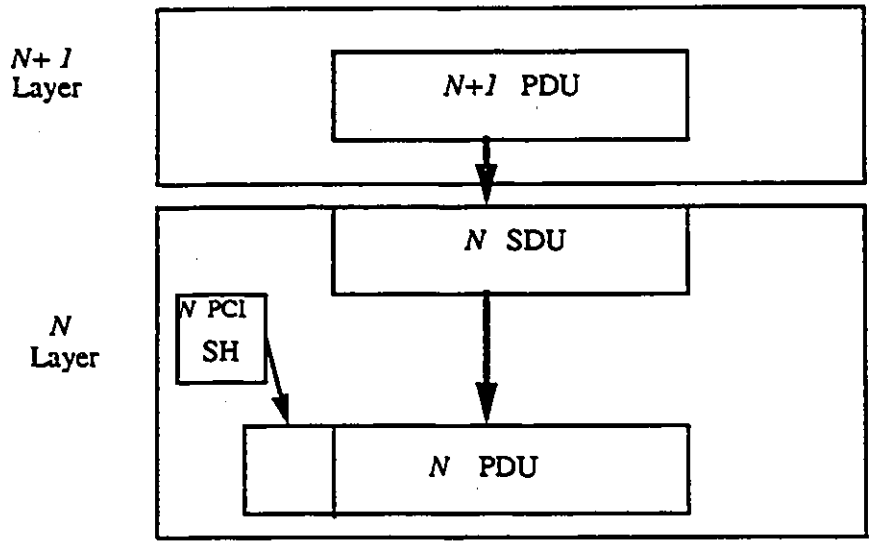


Figure 2.2: Flow of protocol data units (PDUs) through the OSI layers.

Before describing the function of each layer within the OSI model, I will briefly outline the operations of a generic layer within the model. This generic layer description will form a template from which all layers will operate within the OSI model. As the IEEE 802.6 MAN will have to fit into this OSI model it will also have to be defined within the bounds set out by this general description.

Each layer of the OSI model is intended to provide services to the layer directly above (the service user). Each layer will provide these required services by combining the special functions within its own layer and also calling the layer directly below for functions which it cannot perform. The specific requirements and services differ from layer to layer but all the concepts are similar. For example the data link layer will primarily deal with bit errors or out of sequence, frame arrivals, where as the transport layer will have to deal with network failures. Thus similar types of events must be dealt with at each layer but the scope and

magnitude of the events will vary depending on where we are located within the stack.

The following basic concepts remain constant throughout the OSI stack. Each layer N in the architecture provides services to layer N+1 above [SCHS7][ZIMS3] [LINS3]. Figure 2.3 provides a visual representation for this interaction between generic layers. It is shown that a single entity within a layer may service several entities in the layer above. These mulitplexing (at the sender) and demultiplexing (at the receiver) operations may appear throughout the OSI stack. The N layer will rely upon the N-1 layer for the services it is unable to provide. The services which are requested by the N+1 layer of the N layer are passed via the N-service access points(SAP). These SAPs can be thought of as function calls or procedure calls within a standard computer program.

These function calls have been standardized by the ISO and are referred to as primitives. There are only four acceptable primitives which may be used within the OSI architecture: *request, indication, response, and confirm*[LINS3]. Figure 2.4 shows how these basic service primitives provide the interaction between the service user at one level and the service provider at the layer below.

The request is issued by a service user within the N+1 layer in system A to invoke a procedure of the service provider in the layer N below. This request will cause the N layer to send one (or more) N layer PDUs to its peer layer in system B. The receipt of the N-PDU in system B will force an indication to be issued by the N layer in system B to the N+1 layer in system B. The N+1 layer of system B will complete the desired operation and then issue the response primitive to layer N below. The response primitive will force a N-PDU (or more) to be sent from system B back to A. When layer N of system A receives the N-PDU from B it will issue a confirm primitive to layer N+1 above to inform it that the request has been completed. In summary, we can see that the layers will interact via the four defined primitives of the OSI model and the peer entities will interact by exchanging the required PDUs. This separation of the two interaction mechanisms simplifies the OSI architecture. It is obvious that depending on the traffic and network configurations the interactions between stations

could become quite complicated and thus a standardized model for interaction is imperative.

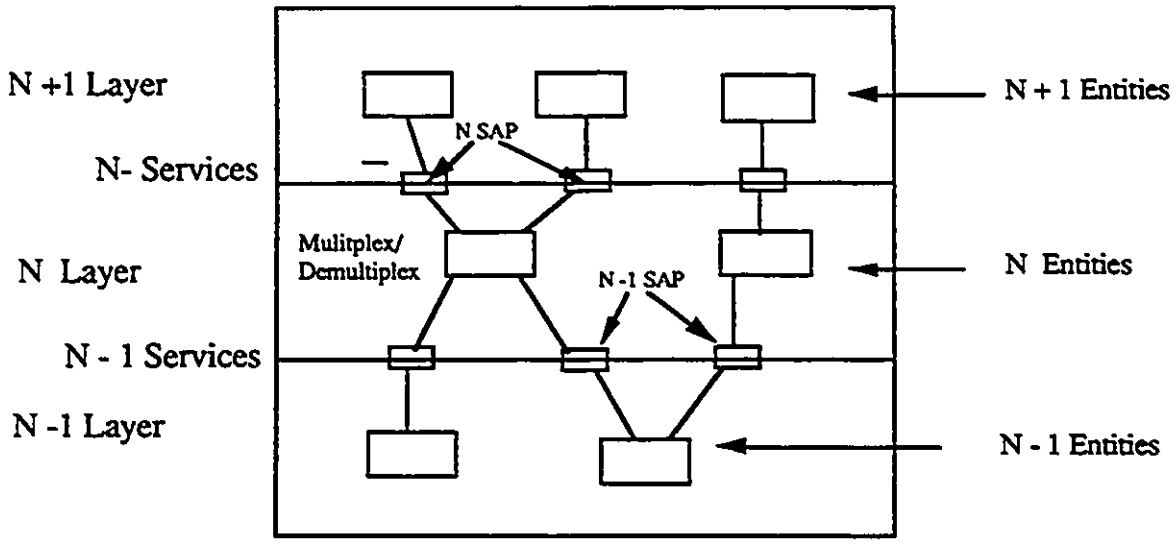


Figure 2.3: Relations between layers, entities and service access points (SAPs).

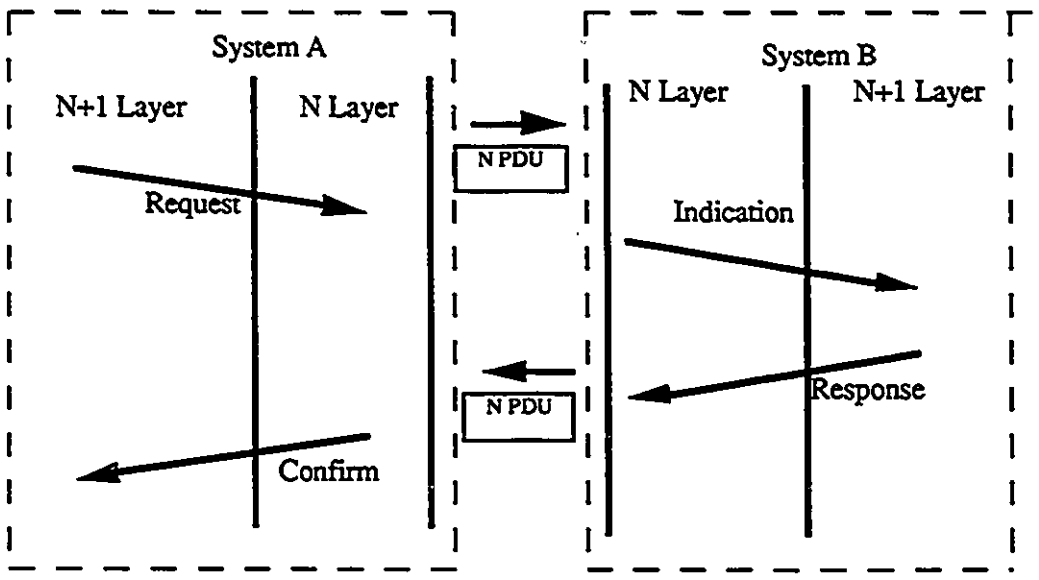


Figure 2.4: Four Basic primitives within OSI structure.

2.1.2 Seven Layers of OSI

This section will briefly detail each of the seven layers of the OSI model. These seven layers can be divided into two groups. The upper four layers reside within the end stations and control both the generation of data as well as the monitoring of the end-to-end communications. The second major group are the bottom 3 layers,which deal with forwarding the data packets successfully through the network to their respective destinations.

The **application layer** is intended to directly serve the end user by providing distributed information services appropriate to an application, and to its management. This layer is also responsible for the meaning of the information being exchanged between the application entities and as such the content of the application layer is defined by the user. There has been some effort to standardize some of the common applications which most distributed users will encounter. Examples of such applications are virtual terminals, file transfers, and finally, job transfer and manipulation services[SCH87][TAN88].

The **presentation layer** is responsible for the negotiation of the transfer syntax and format of the data generated by or addressed to the application layer entities. If the syntax is different between the sender and receiver, it is the responsibility of the presentation layer to map one syntax into the other. The presentation layer's major functions could include data encryption for network security and privacy purposes, data compression for reducing the number of bits to be transmitted, and code conversion for interconnecting incompatible platforms with different terminals, printers, and file servers[HOL83].

The **session layer** is intended to provide the management and control of the dialogue between end users. This consists of setting up a session and assuming a synchronization service to overcome any error detected. This will involve establishing certain synchronization points from which communications may be resumed if a break in communication occurs. This layer also allows remote users to identify themselves in establishing communications with other users[EMM83].

The **transport layer's** purpose is to provide the session layer with a standard access mechanism to the underlining networks. This shielding process will require the transport layer to provide the session layer with the means of achieving reliable and cost-effective transfer of data regardless of the type of network used to support communications. The transport layer could be using a LAN, MAN or WAN for the end-to-end links and thus the particular details of supporting network will be shielded from the session layer. The degree of the reliability and other such characteristics will be detailed by the quality-of-service parameters set up to support each transport layer link between peer transport entities. These quality of service parameters set out the desired throughput, transit delay, error rate, and link failure rate during the data transmission[KNIS3].

The complexity of the transport layer depends on the design of the lower layers. OSI has defined five separate classes for the transport layer[STUS3]. The class chosen for a particular network will depend on the services offered by the network layer below and the quality of service parameters desired for the link. If the lower layers provide virtual circuit services guaranteeing that messages are delivered in order and without loss from sender to receiver, the transport layer protocols become relatively simple. However, if a datagram service is provided by the lower layers and the quality of service parameters desired are far beyond the capabilities of the underlying network then the functionality of the transport layer could become quite complex to guarantee the desired quality of service parameters.

The four layers just discussed all reside within the end stations and are responsible for the data generation and end-to-end control of the communication session. The following three layers are also located within the end station but are also found at all intermediate switching nodes to ensure the data is delivered to the proper destination error free. The IEEE MAN will fall into the domain of one of these next layers as it provides a mechanism to transmit information within a metropolitan area.

The **network layer** carries out the routing and congestion control procedures to ensure adequate delivery of packets from one end of the network to the other. However, contrary to

the other layers in which entities are defined to work in pairs, the network layer comprises entities working in each network node which implement proper routing and congestion control. Similar to the transport layer, the network layer can also provide virtual circuit or datagram services, error detection and recovery. The network layer also uses quality of service parameters to set out the acceptable performance objectives for the link.[WARS3].

The **data link layer** (DLL) has been defined to provide link-level service to the network layer entities. It guarantees error-free, sequential transmission of data units on a link between network nodes. Like the two previous layers, the DLL can provide for data error and loss recovery. The DLL may use a windowed flow control mechanism. The DLL provides services, such as frame error detection (checksum), frame delimiting (by using special bits), and addressing in order to route the frames through the network[TANS8].

Two principal data link applications have arisen in the real world of networks. First we have the WAN type of network which consists of a group of nodes interconnected with multiple links. The link protocol is carried out on each link within the network and thus the network layer may assume that the packets will arrive at the destination as desired because the link layer will guarantee reliable and sequenced data transmission on each link of the network. The second major type of real world application of the DLL is that associated with LANs. A LAN only has a single link for all users on the network. This provides for simple routing and in many cases a null network layer. The LAN architecture is referred to as shared medium because all users on the network will share a common data bus[SCH87]. The IEEE 802.6 committee decided that this shared medium approach would form the basis for the MAN standard.[IEE85]

The shared medium based DLL must deal with the problems of media contention as well as providing reliable and sequenced data transmission. Examples of shared media networks are the star, bus, or ring type topologies. The DLL is usually divided into two sublayers for these types of networks:first the **logical link control** sublayer (LLC), and second the **medium access control** (MAC) sublayer[ANS85A, ANS85B, ANS85C, ANS85D].

The LLC is concerned with the link level flow control, delivery without loss, duplication or misordering and the error recovery of PDUs. These PDUs are exchanged between peer entities of the LLC sublayer. These services are very similar to the services provided by the standard DLL layer within a dedicated circuit switched WAN.

Serving the LLC entities, the MAC sublayer is intended to provide core services and management of the multi-access link. This will ensure that the MAC PDUs can be sent by each node without constant interference from other nodes. The MAC protocols differ considerably depending on the method used to access the network[SCHS7]. This is the level within the OSI model where the IEEE 802.6 MAN standard falls (within the MAC sublayer). This thesis focuses most of its efforts here and will deal with multiple users sharing a common medium within a metropolitan area. The next section will present an overview of various MAC protocols defined by the IEEE 802 committees to date. The review is intended to give the reader a basic understanding of the various architectures and performance characteristics of the members of the IEEE 802 family.

Finally, at the bottom of the OSI stack is the **physical layer**. This layer provides mechanical, electrical, functional, and procedural characteristics for establishing, maintaining and releasing physical connections between data link entities. It transmits, bit by bit, the PDUs generated by the DLL entities. Again aspects of this layer will be discussed within this thesis as it deals with the physical make up of the MAN and stipulates the architectural and timing constraints of the MAN.

The OSI Reference Model serves as the basic model to design future computer networks. It was pointed out in chapter 1 that standards are important to provide for uniform interconnections between different systems and to keep the economy of scale reduced to limit the cost of new networks. The IEEE 802.6 MAN is one of these new networks which must be standardised for the reasons outlined.

2.2 Family of IEEE 802 Standards

Figure 2.5 details the 802 family of standards which deal with the physical and data link layers of the OSI model for the various shared media based systems. The uppermost layer in the figure shows the 802.1 specification which is a companion document to the other 802 standards[ANS85A]. The 802.1 specification details the relationship of the 802.x standards to the ISO model. This companion document will be used to specify network management standards and information on internetworking[IEE90].

Below the 802.1 specification, there is the IEEE 802.2 Logical Link Control (LLC) sublayer which is responsible for the efficient transfer of each data unit on the link. Below the LLC layer we have our five different MAC standard definitions. These MAC access standards define five types of medium access technologies and associated physical media. Each of the five standards is appropriate for particular applications or system objectives. Only three of the five standards have been completed to date: the IEEE 802.3 (CSMA/CD), the IEEE 802.4 (token bus), and finally the IEEE 802.5 (token ring). The emerging standard for the IEEE 802.6 DQDB MAN will soon be completed and the IEEE 802.9 IVD (Integrated Voice/Data) LAN interface definition is currently in progress. A brief overview of the 802.2 LLC and all five MAC standards will be given in the next sections.

2.2.1 Logical Link Control (802.2)

The IEEE 802.2 Logical Link Control (LLC) is the common component to all of the 802.x architectures. The exact details of the protocol are outlined in the ANSI/IEEE standards publication *"Logical Link Control"* which sets out the operating guidelines for peer to peer data transfer over the 802 architectures [ANS85A]. This upper half of the OSI data link layer provides the standard interface to the data link layer from the network layer. This standard interface layer(LLC) will accept data at its SAPs and pass this data on to the network dependent MAC sublayer below.

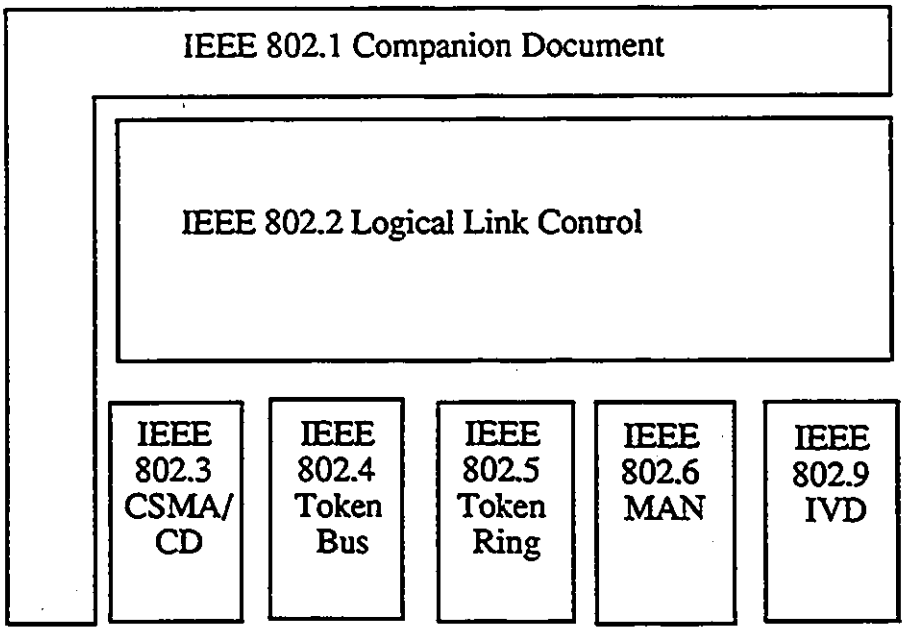


Figure 2.5: IEEE 802 family of standards.

The logical link control carries out the data link control functions which are medium independent[SCH87]. The IEEE MAN architecture falls into this category and thus will rely upon the LLC for its lowest level peer-to-peer communications. These shared media architectures do not usually require a network layer as there is no routing involved. This will cause the LLC layer to deal directly with the transport layer although a network layer may be involved when gateways will be used to move data over or through other networks.

It is clear that the LLC protocol may take on various roles depending on the complexity of the network and transport layers it is supporting. With these various roles in mind the IEEE 802.2 has developed three different classes of the LLC protocol. The first class referred to as type 1 or *connectionless service*, provides for service across a link which will have minimal overhead complexity. This first type of LLC provides for no error recovery. The second type of LLC is the type 2 LLC or more often referred to as the *connection service*. This second type of link is considerably more complex and provides for error recovery. This type 2 LLC is based on the asynchronous balanced mode of the High Level Data Link Control (HDLC)

protocol[SCH87, RYB80]. The third type of LLC which is currently under consideration by ISO is called type 3 and will provide for an acknowledgement based Connectionless Service. This 3rd type of LLC is intended to provide for a compromise between the type 1 and 2 LLC protocols. The type 3 LLC will have little overhead and complexity because of the connectionless service but still provide for a basic form of data acknowledgement[ISO86]

LLC Type 2 frame format

This thesis will only deal with the type 2 LLC protocol to support MAN communications. More detail will be given about this class 2 LLC protocol to help in understanding the basic operations of the protocol. As stated earlier this LLC type 2 is based on the HDLC standard and provides for several frame formats. Figure 2.6 gives the basic frame structure for the LLC PDUs which are exchanged between peer entities on a communication link. The source address as well as the destination address are included within the frame. The control portion of the frame will allow the LLC protocol to transmit 3 different types of frames. The I-frame is used for data exchange, the S-frame is used for control and finally the U-frame is used for connecting and disconnecting circuits. The type of LLC frame is determined by the contents of the control field. Figure 2.7 outlines the 3 different options for the control field.

The I-frame is the basic data exchange unit for the LLC. It contains a send sequence number N(S) as well as a receive sequence number N(R). These numbers are used to label each frame for accounting purposes. The S(N) is the label placed on a frame before transmission and the N(R) is the next expected frame to be received by this station. This process is called **piggy-backing** the acknowledgement. This piggy backing mechanism is used to reduce the number of acknowledgement packets which must be sent as the acknowledgement is incorporated into a standard data I-frame[SCH87].

The S-frame functions in a supervisory role and has 3 different formats. The first format is that of *Ready to receive (RR)* where the N(R) acknowledges all frames up to N(R) -1. The RR frame will be used if a station must acknowledge a frame that has just been received and

has no I-frame to send. The second form of the S-frame is that of *Receive Not Ready (RNR)* and is used to temporally limit the flow of data. This type of S-frame might be sent if the receiving station is having buffering problems and wants to stop the flow of data to it. The N(R) value contained within the RNR S-frame will indicate the next expected frame. Data transmission will only resume after receiving a RR frame from the station which had sent the RNR frame. The receipt of the RR S-frame would indicate that the problems have been cleared up. The last of the three S-frames is the *Reject(REJ)* frame which provides for a negative acknowledgement signal to be sent. The N(R) value within the REJ frame provides for a negative acknowledgement of frame number N(R) plus all additional frames sent after this value.

The final aspect of the LLC frame format, which is of interest to us, is the Poll/Final bit within the control portion of the frame. This bit is used to check the peer entity on the far end of a link to see if it is still active. This portion of the format traces its roots back to the Polling version of the HDLC protocol but is used within type 2 of the LLC as a heart beat check to see if the link is still operating properly.

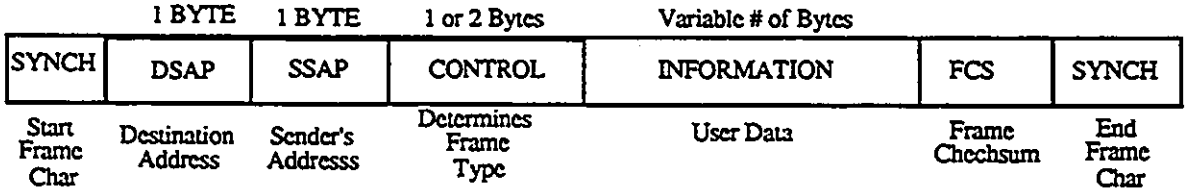
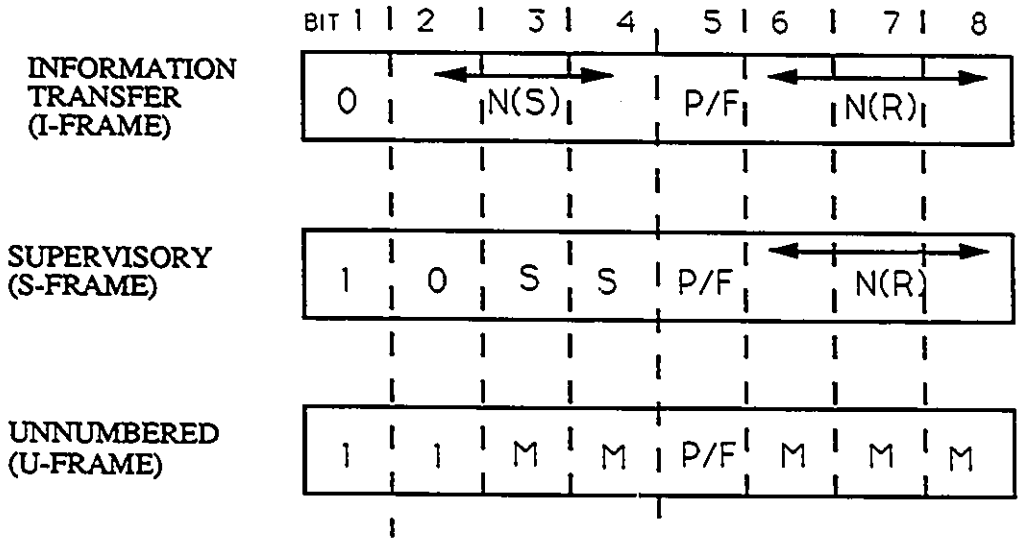


Figure 2.6: Logical Link Control Protocol Data Unit (LLC PDU) frame format.

LLC Type 2 operation

The **LLC type 2** protocol provides for connection-oriented information transfer between two LLC entities[FIES6]. A logical link must be pre-established between peer SAPs before exchanging data frames. The data must be transferred in sequence and without loss or duplication. The principal tools used to ensure the proper flow of data are the labelling

Control Field(1 Byte) Options To Determine LLC Frame Type



N(S): SEND SEQUENCE NUMBER
N(R): RECEIVE SEQUENCE NUMBER
S: SUPERVISORY BIT
M: MODIFIER FUNCTION BIT
X: RESERVED AND SET TO ZERO
P/F: POLL FINAL BIT

Figure 2.7: LLC PDU control field bit description.

mechanisms, N(R) and N(S). These two values will keep constant track of which frames have been sent and properly acknowledged and which ones are still outstanding. The S-frames provide control information regarding which frames to send and which ones are expected. The S-frame also provides a flow control mechanism (RNR and RR) to assist if a receiver gets into trouble.

The previous paragraph indicated that it is possible to have several I-frames sent but their acknowledgments still outstanding. This phenomenon is a direct result of the "Go back n" protocol [BERS7] which is embedded within the LLC protocol. This aspect of the LLC may dramatically improve the throughput of the protocol because it allows a station to send several I-frames before an ACK is received. The number of frames which may be sent is one of the parameters of the protocol which must be determined. A study by BUX et al. showed that this value should be a dynamic one. Its value should be dictated by the traffic conditions[BUX85].

The "Go back n" protocol uses a windowing mechanism to limit the number of outstanding frames allowed by the LLC. The send sequence numbers N(S) used to label I-frames must have an upper bound. Usually this upper bound is set by the number of bits used for the label. The LLC allows 7 bits for sequence numbers which sets the upper limit to 128.

In order to have the "Go back n" protocol function properly the maximum number of outstanding frames must be less than the upper bound of N(S) (128 if we use 7 bits). If we allowed more outstanding frames than possible labels we would have multiple frames with the same sequence number. When an acknowledgement for the sequence number with multiple frames is received the sender of the frames would not know which frame is being acknowledged and thus the protocol would fail. The term window is used to describe a variable which will determine the valid label numbers for future I-frames to be sent.

Figure. 2.8 gives a visual indication as to the functionality of this windowing effect. For simplicity's sake we assume a maximum sequence number of 8. This window variable which will determine the N(S) label value will initially be set to 1 for the first frame to be sent. This

value will increment with each successive frame and roll back over to 1 after using 8. Thus the labels will be modulo 8 values. Figure. 2.8 part A gives a typical case where 5 frames have been sent and are unacknowledged. The window of permissible N(S) labels has been reduced to only 6,7 and 8. Part B shows how the window variable has increased to include frames 1 and 2 as we assume a frame has just been received with N(R) = 3 . The N(R) = 3 indicates the next expected frame is 3 and thus 1 and 2 have arrived intact. This process of allowing multiple frames to be outstanding can greatly increase the throughput capabilities of a network. If the maximum number of frames which are allowed to be outstanding is restricted to one we would have the famous stop and wait protocol.[SCH87]

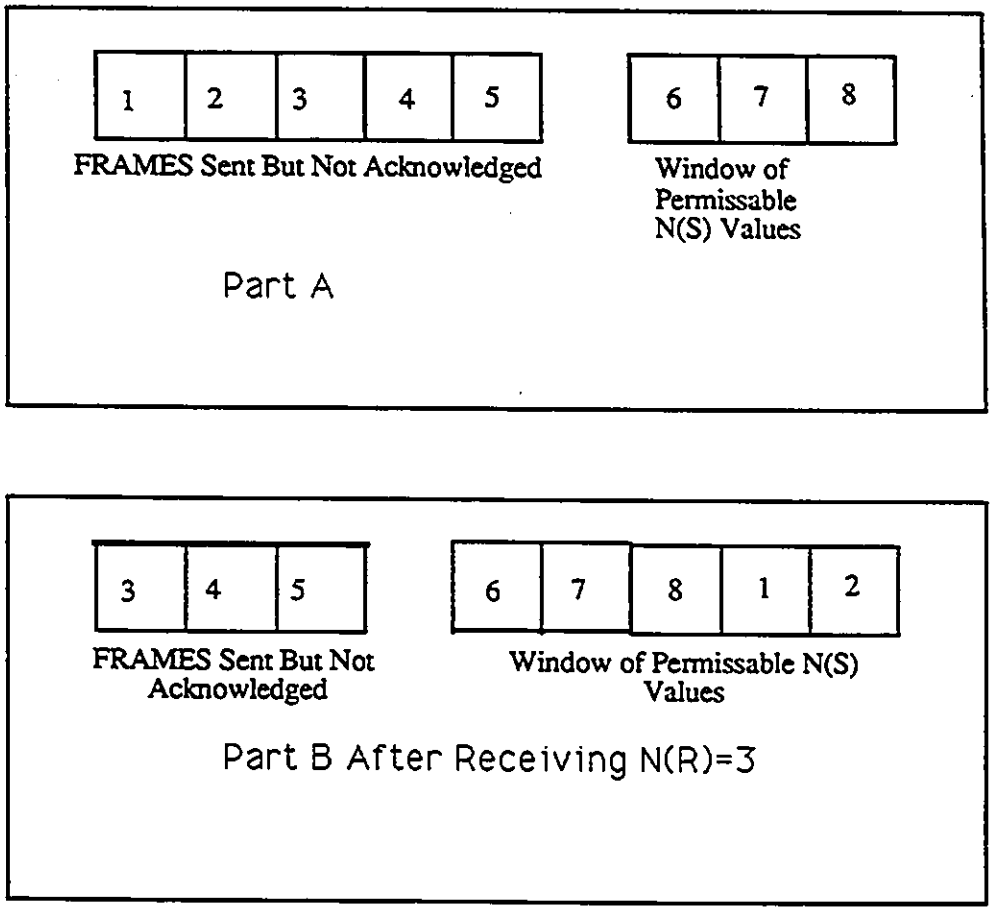


Figure 2.8: Example of the "Go back n" protocol with $n = 8$.

Along with the GO back N mechanism the LLC type 2 relies heavily upon the use of timers to ensure proper synchronization between stations. It is obvious that a station cannot wait forever to receive the proper acknowledgements after sending its frames. The timers are used to determine: when to retransmit a frame, how much time to wait for a poll response, how much time to wait after issuing a REJ frame and finally, how much time to wait for a station to correct a trouble condition which was indicated by a RNR frame. All of these time intervals are important and must be monitored to avoid a system deadlock (stations waiting for an event which will never occur). The timers will also be used to inform the higher layers if the link goes down and communications may no longer continue. The four timers of LLC are as follows:

- *acknowledge timer*: used to define the time interval during which the LLC will wait for an acknowledgement of an I-frame.
- *P-bit timer*: used to define the time interval during which the LLC will wait for a reply to a previous poll command.
- *REJ-timer*: used to define the time interval during which the LLC will expect to receive a reply to the sent REJ-frame.
- *busy-state timer*: used to define the time interval during which the LLC will wait for an indication of the clearance of a busy condition at the remote LLC (used when the LLC has received a RNR-frame).

2.2.2 802 MAC Standards

The 802.2 LLC description given in the previous section provides for the media independent interface to the DLL layer. Below the LLC sublayer we will find the media dependent portion of the DLL. This portion is referred to as the media access control (MAC) sublayer. The 802 standards body has completed the 802.3 (CSMA/CD), 802.4 (Token bus), and 802.5 (Token Ring) MAC standards. The finishing touches are currently been put on the 802.6 (DQDB)

standard and finally work is continuing on the 802.9 (Integrated Voice/Data (IVD)) LAN interface. In order to introduce you to the types of architectures and relative performance of the 802 MAC family, the CSMA/CD, Token Bus, and Token Ring standards will now be described. A brief comment on the performance of each will also be given. By looking at the architectures and performance limitations of the existing 802 standards we might understand the need for the 802.6 DQDB MAN. The 802.6 DQDB protocol will be discussed in detail in the next chapter of this thesis. The 802.9 IVD standard will simply be introduced as its definition is ongoing.

CSMA/CD IEEE 802.3

The CSMA/CD 802.3 standard[ANS85B] provides for a common bus to be shared by all users. The CSMA/CD standard details signal rates from 1 to 20 Mbps using several different media types. Figure. 2.9 displays stations attempting to share the common CSMA/CD bus. The key aspects to this standard are, first a station must listen for a carrier and transmit only when there is no energy detected on the bus. Second, while transmitting a station must monitor the bus for a collision. If a collision is detected the station will send a short collision burst to inform all other stations of the collision and then back off for a random period before attempting to retransmit.

The retransmissions of I-frames uses a truncated binary-backoff algorithm which states that before retransmitting the colliding stations will wait a random number r slot times. The value r will be a uniformly distributed random variable between 0 and 2^n. n equals the number of retransmission attempts. This will provide for an increasing pool of random wait times as collisions continue to occur. However the n value will not increase above 10. The random wait time possibilities then remain at 2^{10} until n reaches 15. At this point the collision event is reported as an error to the higher layers where appropriate action must be taken.

The slot time within the CSMA/CD standard provides for the synchronisation required to avoid the Aloha type of collision problems. The fixed slot size will stipulate the maximum

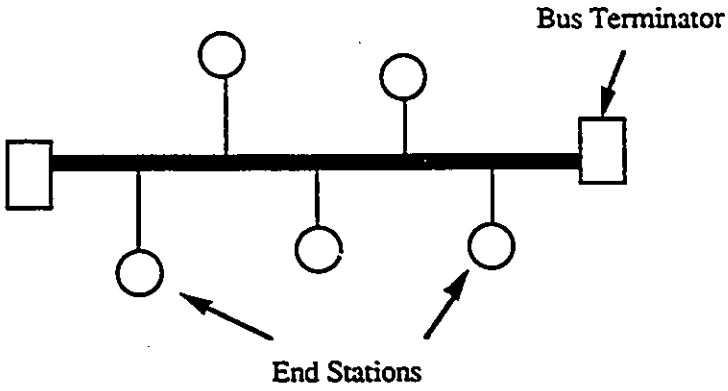


Figure 2.9: 802.3 CSMA/CD Bus Architecture.

frame size and the scheduling quantum used for retransmissions. This information will provide the lower boundry for bus access delay[BUX81].

The performance of the CSMA/CD bus has certain limitations which must be pointed out. Frame collisions is an obvious problem. To limit these collisions the end-to-end propagation delay must be limited, which will force short bus lengths. The frame length should be relatively small to avoid numerous collisions. As an example the Ethernet which is a coaxial cable based CSMA/CD application has its speed fixed at 10Mbps with a 1500 meter maximum bus length and slot size set to 512 bits[SCH87]. The positive aspects of the CSMA/CD performance comes to the forefront when the network is operating with a light load and thus relatively few collisions occur. This will provide the users with very short bus access times (in the order of one half of a slot) and with very little protocol overhead.

Token passing bus IEEE 802.4

The Token passing bus (802.4) uses a similar medium architecture to the CSMA/CD bus. Both use a common bus which is shared by all stations[ANS85C]. The access protocol to the bus is quite different from that of the CSMA/CD. The Token passing bus avoids the random access approach of CSMA/CD by introducing a control mechanism which governs

access to the bus. The control mechanism is the token frame which in effect is "permission to transmit". All stations will have a turn accessing the bus, the stations turn occurs when it receives the token frame. An obvious problem with this method is determining how much time each station must be given to transmits I-frames after receiving the token. The standards body has introduced a Parameter called the Token Holding Time(THT) which stipulates the maximum time any station may hold the token and transfer frames. The standard calls for bus capacities of 1,5 and 10 Mbps.

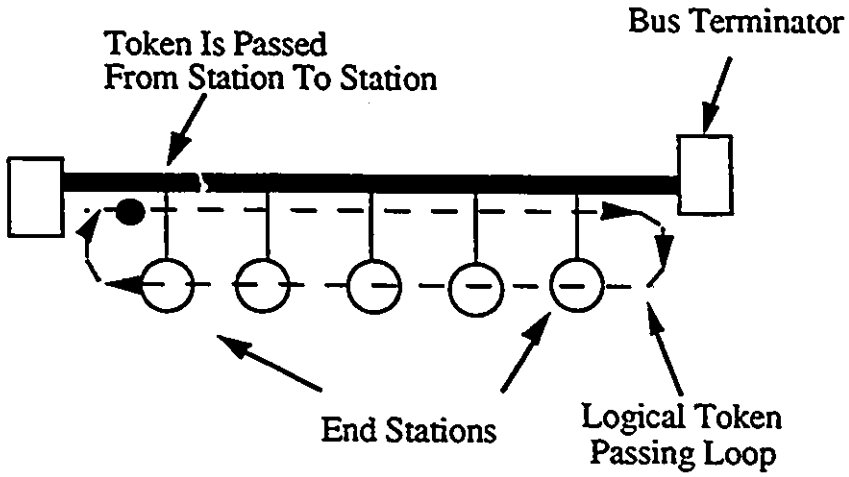


Figure 2.10: IEEE 802.4 Token Passing Bus Architecture.

The stations will pass the token frame from one to another on the bus in a logical order. Each station is assigned a logical number and will pass the token to its successor (station with next lowest logical number) when either it has sent all its data or the THT has expired. When the THT fires, the station is forced to pass the token on to the next station. The logical order obtained by the successive passing of the token forms a logical ring (see Figure. 2.10) on the bus. A problem with this type of protocol is that new stations joining the bus will never be handed the token because they are not already known by the others on the bus. The standard has corrected this problem by periodically generating a "Solicit Successor" frame which invites new stations to join the bus. Other special control frames which have been added to help administer the bus are frames to re-establish a lost token and also frames to

delete disconnected stations from the ring.

802.4 standard includes a priority mechanism with eight separate classes of priority. This portion of the standard was added to overcome the problem of a station holding on to the token for a relatively long period of time and transmitting only low priority traffic. The stations will transmit their high priority traffic first then proceed to the lower levels of priority. The standard has introduced the Token Rotation Time (TRT) parameter to help divide the bus bandwidth between high and low priority traffic. Basically the TRT is the acceptable amount of time a station should wait between turns in transmitting high priority frames. If the token completes its rotation to all stations faster than this value the stations will be able to transmit some of the lower priority traffic. This type of bandwidth division will ensure that the high priority traffic will always be promptly serviced within a specified time frame and the lower priority traffic will obtain the remaining bandwidth after the higher levels are serviced.

Performance is a key issue with these types of networks. A principal goal of this standard is to provide predictable performance behaviour for its users during all traffic conditions. Unlike the CSMA/CD network which is entirely probabilistic and virtually impossible to guarantee a minimum I-frame access time, the token passing bus has gone through great pains using the THT and TRT to guarantee predictable network access times. Indeed by setting strict THTs the standard does provide for very predictable access delays. It must be pointed out that several recent studies have shown that fixed THTs and TRTs have lead to network oscillations in performance and has suggested that a dynamically determined TRT parameter may improve performance [PANS8][PANS9]. This point of predictable access delays and dynamically determined network parameters will be a key topic in future portions of this thesis.

Token Passing Ring IEEE 802.5

The final IEEE 802 standard that has been completed is the 802.5 Token Passing Ring[ANS85D]. This standard is based primarily on the work of the IBM Research Laboratory in Zurich,

Switzerland[BUX81]. The operating concepts of this ring are very similar to the 802.4 standard. All stations will share a common bus, in this case the bus is looped to form a physical ring. The stations will wait until they receive the permission token before sending their I-frames. All stations will be in either a transmit or repeat state. While in the repeat state a station will simply forward any frame to the next station on the ring bit by bit as it is received. The station may copy the frame as it is forwarding it, if the station detects its own address as the I-frame destination. A station will enter the transmit state after receiving the transmit token. The station will continue to transmit all I-frames currently in its queue until either the station empties its queue or the THT timer fires forcing the station to relinquish the token.

Figure. 2.11 depicts a typical Token ring structure. Each station is physically connected to the ring and is capable of both transmitting on to, and receiving from the ring. A bypass mechanism is usually included in the network hardware to ensure that if a single station fails, the bypass can be engaged to work around the faulty station[SCH87].

The priority mechanism which is built into the 802.5 standard is quite different from that of the IEEE 802.4 token bus. The token itself sets the priority. When a station receives a token it may only send a frame with an equal or greater priority level than the value indicated by the token. The token also contains several bits which may be set by stations which have higher priority traffic to send. The setting of one of these higher priority bits will force the station responsible for issuing the next token to increase the priority level. The higher priority data will be serviced by this higher priority token. A station which increases the priority level of the token will be responsible for lowering it to its previous value for normal operation.

The token passing ring relies on a distributed access protocol for data transmission. However, a supervisory role must be maintained by one of the active stations to correct network problems[BUX81]. For example, a frame may circulate the ring indefinitely. It is usually the responsibility of the sending station to remove its frames, but a station could fail after transmission and not have the capability to remove its own frame. A second possible network

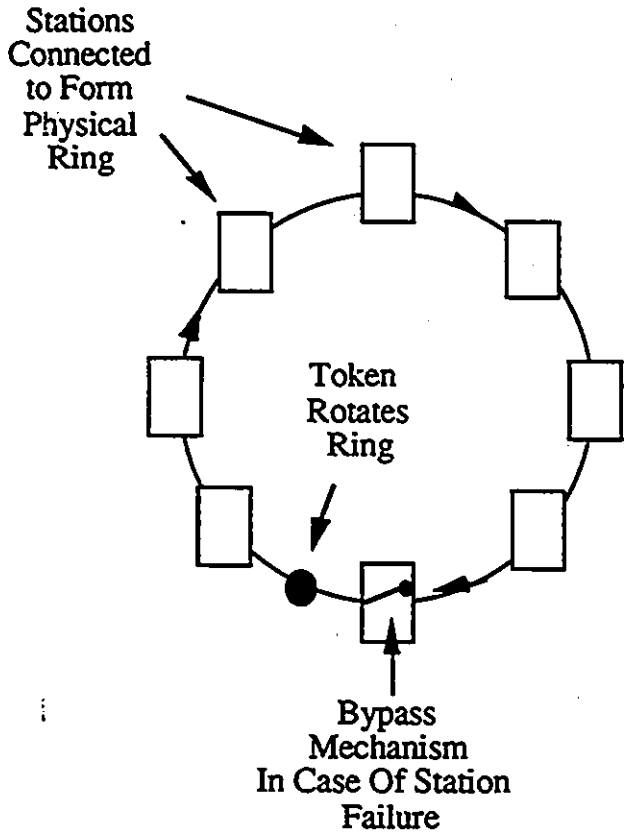


Figure 2.11: IEEE 802.5 Token Passing Ring.

problem is token loss. This will occur when a station does not regenerate the required token as required. The supervisor will have to determine that this has happened and generate a new token[SCHS7].

The performance of the Token Passing Ring is comparable to that of the token passing bus although a new version of the standard allows speeds up 16Mbps[IBMS8]. We obtain predictable access delays which are directly related to the THT value. A study by Bux in 1981 has indicated that the token ring does display improved performance over the CSMA/CD protocol mainly due to the poor performance of CSMA/CD during periods of heavy bus load[BUXS1]. The CSMA/CD network as expected does provide for fast access delays at low network utilization.

Again we see the debate of predictable performance versus fast network access. Minimal overhead collision based networks, like CSMA/CD display very low throughput with high delays during heavy network utilization. Networks which are tightly controlled require a large control overhead but display predictable network performance. It seems the ideal network would be a hybrid of these two approaches which would allow easy access to the shared media during periods of light load and enforce tight operating procedures during peak load periods.This concludes the review of the 802 MAC standards which have been completed to date. The 802.6 MAN standard which is in the finishing stages of standardization will be the focus of the next chapter of this thesis.

Integrated Voice/Data 802.9

The 802.9 IVD standards committee is still formulating this standard and from experience gained by following the 802.6 standard, major changes are possible and this section will provide general information only[AND84].

The 802.9 IVD (Integrated Voice/Data) standard is not a typical member of the 802 family because it is not purely a MAC protocol, but rather an interfacing standard. The goal of the

802.9 committee is to standardize the interface between an integrated voice/data workstation (IVDWS) and an integrated communications network which provides both ISDN and LAN services[IEE89, MOU88]. Although not directly related to this thesis, the interface standard is of interest to me because the 802.6 MAN will be one of the integrated communications networks which may be called upon to support the IVDWS.

The interface is intended to provide the workstation user with high speed packet switching capabilities as well as circuit switching functions. The IVDWS will be physically attached to a network access unit (AU). The AU will then be the access mechanism to the integrated network. The standard is calling for four different types of channels (2B+D+P+C) to join the AU to the IVDWS. Each of the channels will provide a logical link for the various types of services which will be supported by the IVDWS of the future (see Figure. 2.12)[JUL86]. The channel descriptions are as follows:

- *B-channel*: 64 kbps channel intended to support digitized voice and other subscribed ISDN services.
- *D-channel*: 64 kbps for the provision of call control via the Q.93x protocols defined in the ISDN standards [KAN86].
- *P-channel*: packet channel that provides IEEE 802 MAC services for data.
- *C-channel*: isochronous channel in which the bandwidth is a multiple of 64 kbps and intended to carry wide-band circuit switched data.

The work is still ongoing to define the bandwidth allocation mechanisms and other out standing issues. It is interesting to note that with this much effort spent towards defining an interface standard to high speed integrated networks, it is clear that there will be a major requirement for these types of networks in the not too distant future. In fact the highspeed IVDWS users will want to communicate over fairly large distances which are beyond LAN capabilities. It is clear that a new concept in networking will be required to fill the gap left by the limited capabilities of the current LAN architectures.

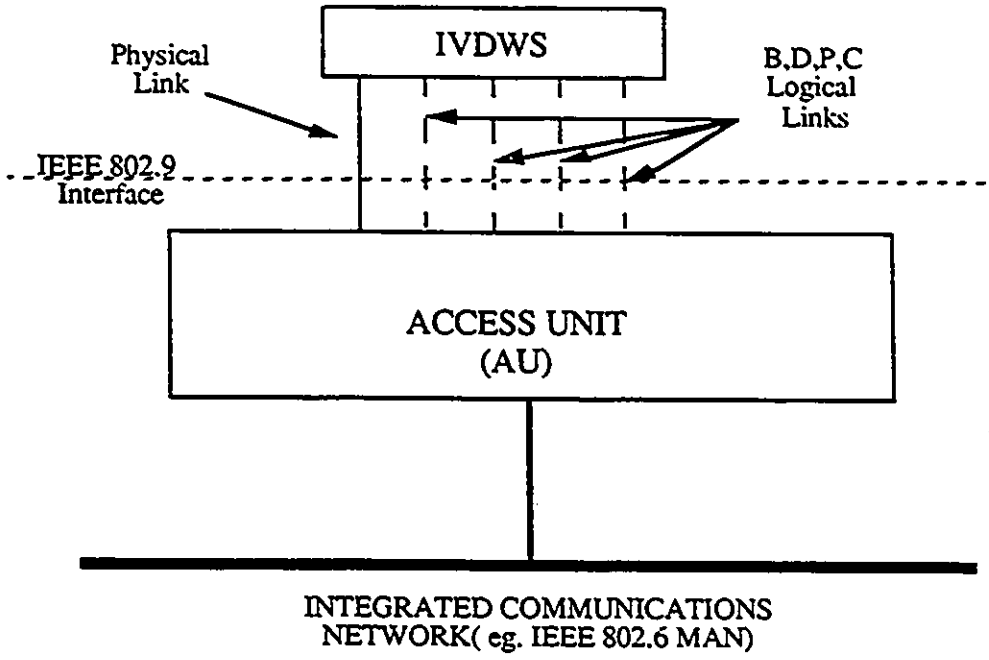


Figure 2.12: IVDWS AU Logical Channel Links.

2.3 LAN - WAN interconnection requirement

Metropolitan area networks are a relatively new idea in the computer network industry. The vast majority of work dealing with computer networks has dealt with wide area networks (WANs) which operate over the vast telephone network. As the industry was limited to using the existing cabling provided by the telephone industry the very familiar RS-232 interface has become a fixed point through which most computer communications has taken place. The telephone companies have a big advantage in the communications game because they have sole access to lay cable in a public domain. This "Public Right Of Way"is very difficult for corporations to obtain as permission of the carrier is required to cross into thier domain. For these reasons any large scale computer network will normally involve the carriers.

The local area network(LAN) has departed from this telephone based concept and has adopted the shared media approach. The shared medium is usually serial and may operate

at sufficient speeds to accommodate many users which will contend for the shared medium. The LAN's ancestry is quite different from that of the WAN. The LAN is a concept which has been driven by the computer industry and not by the public telephone system. LANs are privately owned and operated and thus avoid user fees, security concerns, and the "public right of way" problems discussed earlier.

The IEEE 802.6 MAN is required to try and bridge the gap between these two concepts. Because of its bridging role the MAN will have one foot in each camp. The computer industry first thought it would extend the LAN. It was soon realized that with the size and scope of the job to be done, a central body is needed to standardize the new MAN network rather than allowing one of the carriers or CATV companies set the standard[MOLS8A]. The need for a central standardization body for MAN's was increased when it was decided to make the MAN compatible with the asynchronous transfer mode (ATM) based WANs of the future. MANs will be functioning as the bridge linking LANs to these new ATM based WANs. The standards body chosen to define the MAN should have a working knowledge of both ATM principals as well as LANs.

A major driving force for the development of the MAN architecture is the emerging trend toward distributed computing. Distributed database management, distributed resource sharing, and other various client server applications are pushing for a new concept of computer networking. Part of this new concept of highspeed distributed systems is that the network will be thought to be part of the computer rather than just wires connecting different resources [DES90] [LOMS9]. The network will almost be considered as the bus of a computer. The high speed network will be considered an importatnt part of the computer rather than an extension. Standard LANs will not provide the required bandwidth to support these high speed workstations especially if realtime multimedia communications will be involved.

It is clear that a central body which has experience and knowledge of both LANs and WANs must be used to establish a clear standard for this new MAN architecture. The preferred choice for this body is the IEEE organization which is currently used for the standardization of the

other 802 family members for computer networks. The need for an IEEE MAN standardization body was suggested in 1981 by the IEEE 802 project[MOL88A].

2.4 IEEE 802.6 Standard Committee Goals and Objectives

The IEEE 802.6 committee was formed in 1982 to establish a standard for this new type of network which would bridge the gap between the LAN and WAN. The first things which had to be decided were the committee's goals and objectives for the MAN. The following goals and objectives would form the basis for defining the requirements for the MAN architecture:

1. Distance of coverage: City and Suburbs (50 km).
2. Lan connectivity: User observes no differences between communications within a room or within the city.
3. Data Type: Both synchronous and asynchronous (Voice and Packet).
4. Speed: High Speed to support real time applications.
5. Security: Maximum security of information transported on the MAN.
6. Reliability: Have fault detection and self-healing capabilities.
7. Price for service: Public MAN should have Competitive pricing.
8. Support: Service and Support should be available from the carrier.

The above list gives a general outline of the goals that the IEEE 802.6 committee had set out for itself to meet concerning the architecture of the MAN. [CRA90][MOL88A, MOL88B]

2.4.1 Functional Requirements of MAN

It is clear that the functionality of the MAN will be greater than that of the LAN and more oriented towards that of a public network. The MAN architecture must be much less sensitive to network size than the other IEEE 802.6 LANs. The IEEE committee stipulated that the typical diameter of a MAN will be in the order of 50 kms[IEE90]. As discussed in earlier sections, the existing 802 MAC standards are very limited in bus length. The large token rotation times of long token rings and high collision rates of large CSMA/CD networks would make them unacceptable for a MAN architecture of 50 Kms[SCH87].

A major concern to the 802.6 committee are the capacity requirements of this new network. The MAN will be used to support both LAN interconnection as well as supporting high speed work stations. This will stipulate that the MAN capacity be considerably larger than that of the LANs and work stations which it services. The committee first restricted the capacity of the network to 20 Mbps but this constraint was later removed. Now the MAN capacity will be dictated by the capabilities of the communications media of the MAN[IEE90].

The MAN will be used to transport both "time sensitive" and "non-time sensitive" type traffic better known as isochronous and non-isochronous. Each of these types of traffic have their own special requirements which must be fulfilled.

The time sensitive isochronous traffic could be that of voice , video or any media which requires bandwidth at regular intervals. These time sensitive types of traffic must be delivered to their destinations within very strict time constraints. If the time constraints are not achieved the voice message received will not be understandable. The same could be said for video signals which use a predefined screen refresh rate. If the data is not received to keep the refresh rate updated the picture will become distorted. I believe that after this standard is proven to be effective many more applications will be developed which have strict time constraints which must be met.

The other major category of traffic for the MAN will be data communication between

computers. This non-isochronous traffic is also known as packet oriented. The most important requirements of the MAN which must be met to support this type of traffic are reliability and very low data loss rates. A typical user of this type of traffic could be that of a banking instant teller machine. This machine is required to have the network available 24 hours a day 365 days a year. This type of machine also requires error free operation. If a network error moves a decimal point several spots to the right, it could cause a bank a great deal of lost revenue. A second typical user of this packet oriented traffic could be an image file transfer application where the amount of data transferred will no longer be in the order of several packets with a mean of 1000 bits. A typical session could involve the transfer of tens of thousands of packets each with several thousand bits[VALS9]. We could almost put this type of data transmission into the "time sensitive" category because a user waiting to view an image does not want to wait an extended period of time.

2.5 Conclusion

This chapter has presented an overview of the OSI 7 layer model as well as a review of the relevant 802 standards which are related to the 802.6 MAN. The review had brought up the debate of minimal overhead with unpredictable performance versus complex protocols with predictable results. This will be a topic of interest carried throughout the thesis. The positive and negative performance characteristics of the other 802 protocols are highlighted to get an insight into the possible performance of the 802.6 MAN. The final portion of the chapter displayed a clear need for a standardized MAN architecture. It also set out the goals of the IEEE 802.6 standards committee. These goals lead to the functional requirements the MAN must provide. The next chapter will describe the 802.6 MAN standardization process based on the functional requirements set-out in this chapter.

Chapter 3

IEEE 802.6 DQDB MAN Standard Progression

The previous chapter described the functional requirements for a standardized metropolitan area network. It also gave a background description of the other members within the IEEE 802 network family.

It is clear that this new MAN architecture will require capabilities far beyond those of its other 802 family members. The integration of both isochronous and asynchronous data traffic on the same network will require a completely new concept in computer networking. The specific requirements of both types of traffic must be fulfilled. The isochronous traffic is primarily concerned that the timing abilities of the network are sufficient to avoid distortion. The asynchronous traffic is more concerned with data integrity. To achieve data integrity a network must display very low data error and loss rates.

This new architecture will attempt to address the concerns of these two separate data types. But at the same time try to keep the architecture simple enough so that it can be produced by the industry at a reasonable cost. We will see that efforts were made to keep the standard complexity to an absolute minimum. This lead to unpredictable performance in the early versions of the MAN standard. Issues brought up in the previous chapter dealing with minimal overhead with unpredictable results versus very complex predictable protocols will resurface in this chapter.

This chapter will track the IEEE 802.6 DQDB MAN standard from the QPSX based initial proposal to its present day form. A detailed description of the standard's architecture will be given, as will an overview of the DQDB MAC protocol. The chapter will then focus on the performance issues of the standard. The early version of the standard was found to behave unfairly under certain traffic conditions. This unfairness and modifications to the standard to correct the problems will be discussed.

The final portion of the chapter will give a performance evaluation of the latest version of the MAN standard. The performance evaluation was based on a QNAP2[QNA84] queueing model of the final version of the DQDB MAN standard.

3.1 Evolution of DQDB

The IEEE 802 committee reached an agreement on the architecture for the MAN standard in 1986. The winning proposal was called QPSX (Queue Packet and Synchronous Circuit Exchange)[BUD86, NEW86, NEW88B]. The proposal was submitted by a company called QPSX Communications Pty. Ltd.. This company was formed to develop and market QPSX products and is supported by Telecom Australia. The QPSX proposal was developed by a research group from the University of Western Australia. The work is based on earlier developments of an integrated packet/circuit switch by Budrikis[BUD84] and work on the Fastnet network by Limb and Flores[LIM82].

3.1.1 QPSX: Early Proposal For 802.6 MAN Renamed DQDB

The IEEE 802.6 committee decided to rename the proposed architecture from QPSX to DQDB (Distributed Queue Dual Bus)[IEE87A] in order to avoid confusion with the QPSX company. Keeping the QPSX name would seem like an endorsement for the company, and there would be legal copywrite problems concerning the name QPSX.

The DQDB proposal was intended to meet all of the functional requirements set out in the

last chapter for the metropolitan area network. The proposal would allow for both synchronous and asynchronous based traffic. The protocol was unique in that it would allocate the required synchronous bandwidth on a demand basis and then evenly divide the remaining bandwidth using a distributed queuing mechanism. This distributed queuing mechanism was intended to allow an orderly formation of a network wide access queue which would provide for packet access to the network in the same order as they were generated system wide[NEWS6].

The predominant limitation of the other 802 family LANs is the network size. Large size networks display long propagation delays which cause either packet collision problems for random access protocols, or excessive delays for control information on the controlled access protocols. The DQDB protocol was thought to be the ideal solution which would allow immediate access during low network utilization and minimal overhead to control access during heavy load periods. All this control would be independent of both network speed and size[BUDS6][NEWS6] [NEWS8B][IEES7A]. Indeed it was first thought that this proposed architecture was very promising and would solve the major problems of shared media based networks. It was soon learned that this was not the case and the protocol would require some modifications. The next sections will describe the architecture and timing specifications of the DQDB standard.

3.1.2 DQDB Architecture and Timing

The proposed IEEE 802.6 MAN standard is based on the QPSX proposal which consists of a pair of contra flowing unidirectional buses. These buses are referred to in the standard as bus "A" and bus "B". These buses support communications between a multiplicity of nodes situated on the dual bus structure. Figure 3.1 gives a visual depiction of the bus structure. Full duplex communication links may exist between any pair of nodes on the network. All stations have the capacity to both transmit and receive on both buses simultaneously.

The selection of which bus is used for transmission and which for reception is made entirely on the basis of the direction of the destination address from the sender. Since each bus only

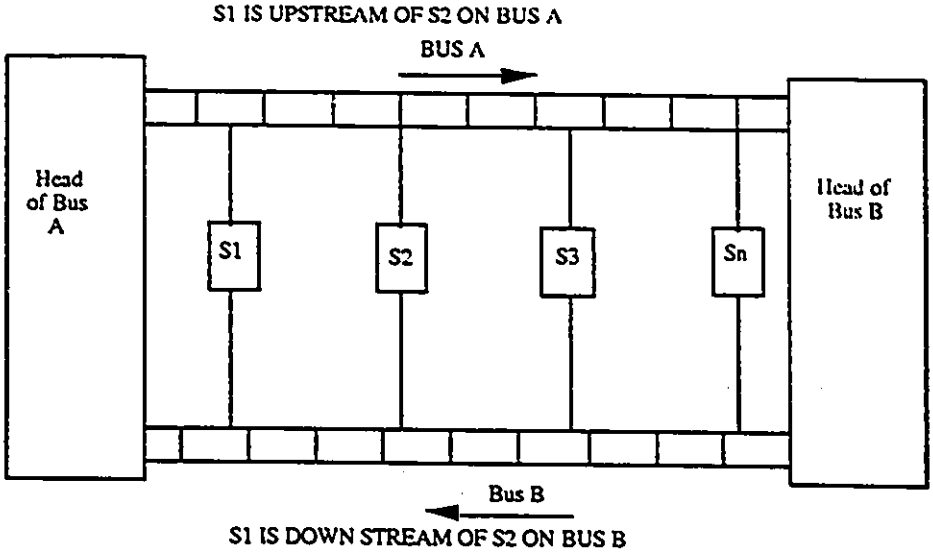


Figure 3.1: IEEE 802.6 Dual Bus MAN Architecture.

flows in one direction the choice of a bus becomes obvious. The terms "upstream" and "downstream" are used throughout the current literature when referring to direction. An upstream station will see a slot as it travels along a bus before a downstream station will see this same slot. The head end station which is used to generate the slots for a particular bus will be the most upstream station on that bus. This same station will be the most down stream station on the opposite bus as it will be the last station to see any slots on the opposite bus before they are destroyed.

The dual bus architecture has been chosen for both the high bandwidth capacity and also for the reliability which the dual bus structure provides. The high bandwidth is caused by both buses being operational at all times. This will provide for twice the capacity of a single link. The bandwidth of DQDB will be first allocated to the stations to fulfil their synchronous bandwidth requirements on a demand basis. The remaining bandwidth will be shared between all active stations on the buses based on the distributed queueing protocol (to be discussed in later sections).

Several factors give the DQDB MAN its high reliability. First, the architecture calls for the

buses to pass directly by each station but not through the stations as in many networks (e.g. token ring). The nodes are physically attached to each bus by a read and write connector. The read connector is physically placed ahead of the write connector on each bus. The node will read the incoming data and when the node has determined that it is its turn to transmit, it will logically "OR" its data onto the bus as an empty slot moves past.

Figure 3.2 shows how the station is physically attached to the dual bus architecture. The concept that the bus does not physically pass through the station provides for no detrimental effects if the station fails passively. A passive failure occurs when the station does not attempt to flood the network with garbage. The second major reliability aspect of this network is that the dual bus architecture can be manipulated into a loop structure. This looped dual bus configuration is capable of reorganizing itself around a major fault in the network (see figure 3.3). If a disaster occurs, such as a backhoe tractor breaking the physical link between stations, the functions associated with the head end of each bus may be reassigned to the stations physically located on each side of the break. After a brief interruption of service the system will return to normal operation with little or no side effects to the user. Figure 3.3 shows the self healing mechanism of DQDB to reorganize itself around a major fault. This fault might be either a break in the physical media or a station which has failed in a manner which floods the network with unwanted traffic or garbage.

The DQDB architecture calls for the end stations to provide the global timing signals for the MAN. All synchronization from bit through to frame timing will be controlled by the head end of each bus. The basic timing will consist of frames which contain a number of fixed size slots (53 bytes). The frame timing will be based on implicit frames of 125 μs. The 125 μs cycle time for the frame will provide for each byte within the frame to support a 64 kbps channel. These 64 Kbps channels will form the basic unit of allocation for the synchronous channels[IEE90]. The DQDB standard has not set any strict timing considerations for the standard except for the frame time. This allows the users to choose the operating speeds of their DQDB based system based on the available resources. The preferred choice for the DQDB bus medium will be fiber optic because of its very high bandwidth capabilities and

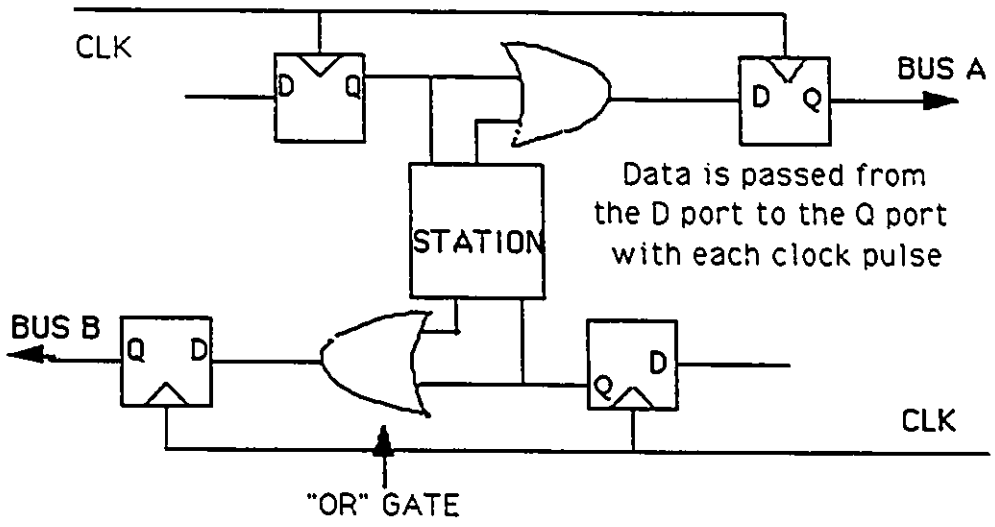


Figure 3.2: Stations Physical Connection to IEEE 802.6 Bus Structure.

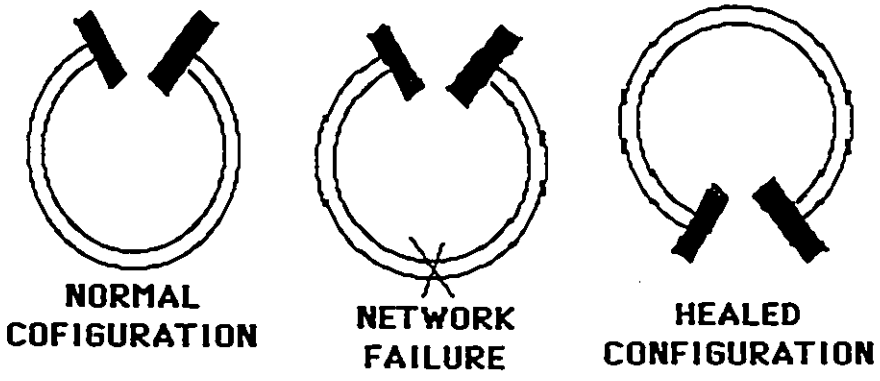


Figure 3.3: DQDB's Self Healing Mechamism.

low bit error rates. The likely choice for the DQDB capacity is 43 Mbps (DS3 compatible) or 155 Mbps (compatible with SONET based systems).[IEES7A]

3.1.3 DQDB'S Media Access Protocol

As stated earlier all communications on the DQDB MAN are based on implicit frames with a 125 μs cycle time. These frames are comprised of a given number of fixed sized slots. These slots are shared between the isochronous and packet oriented traffic sources (see figure 3.4). The 802.6 committee has decided to keep the DQDB slot size compatible with the proposed cell size of ATM at 53 bytes. The ATM cell calls for 5 bytes for overhead and 48 for user data. The 802.6 DQDB committee has defined its slot format the same as the ATM cell to make the two networks compatible[IEE88B].

Figure 3.5 outlines the general structure of the MAC and physical layers of this standard. We can see that the MAC layer and DQDB layer are in fact the same layer. This terminology will be interchanged throughout this thesis. The DQDB layer provides two basic services to the layers above. First the pre-arbitrated services will provide the required bandwidth for isochronous type data services. The second service type is the queued arbitrated service which will provide the asynchronous data services to support the LLC layer directly above. Both the DQDB and Physical layers have a management portion to control layer activities. The management functions will include task monitoring within each layer and communication with other management entities for effective network control.

Pre-Arbitrated Data Services

The Isochronous Service supports the transfer of isochronous service octets(bytes) between two Isochronous Service Users (ISUs) over an already established isochronous connection. The service is called isochronous because there is a constant inter-arrival time for the service octets[IEE90].

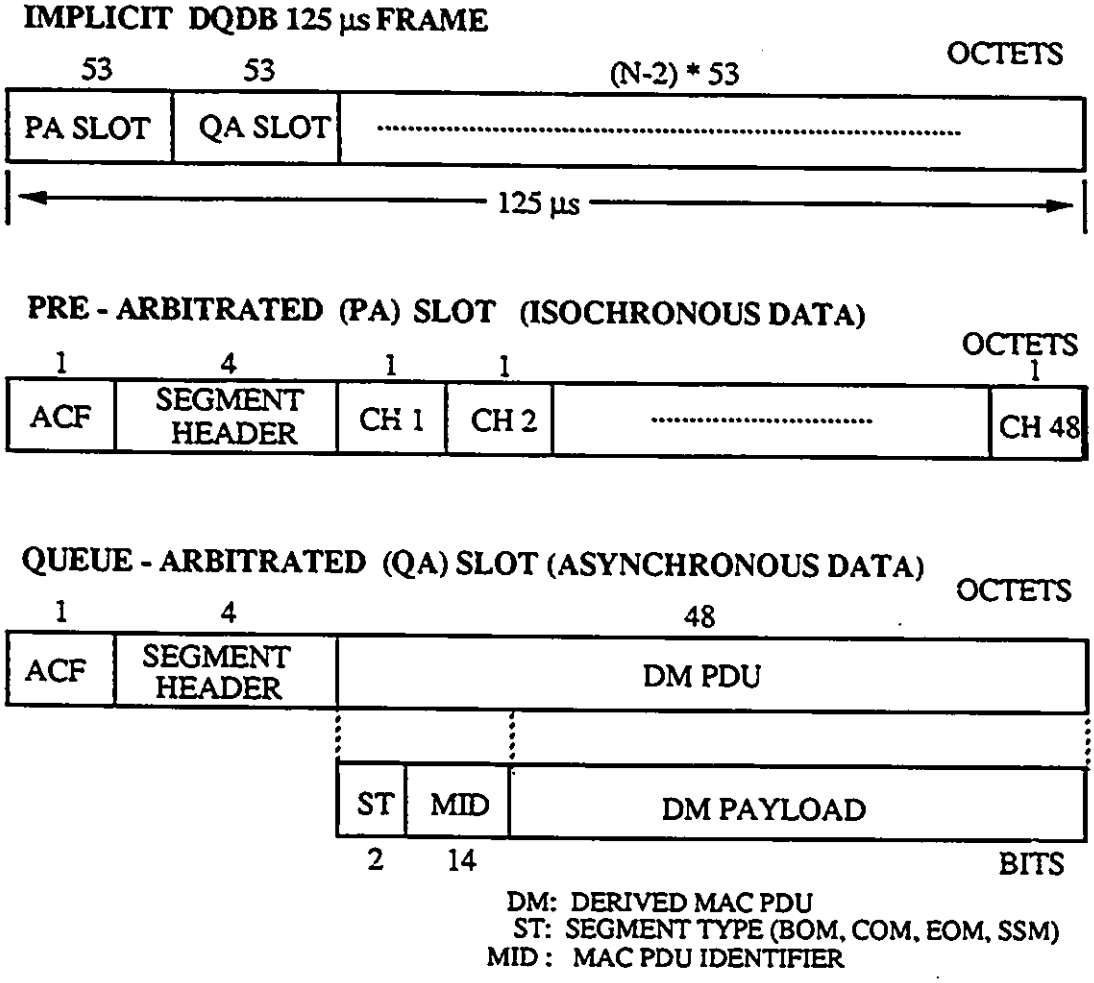


Figure 3.4: IEEE 802.6 Implicit Frame and Slot Formats.

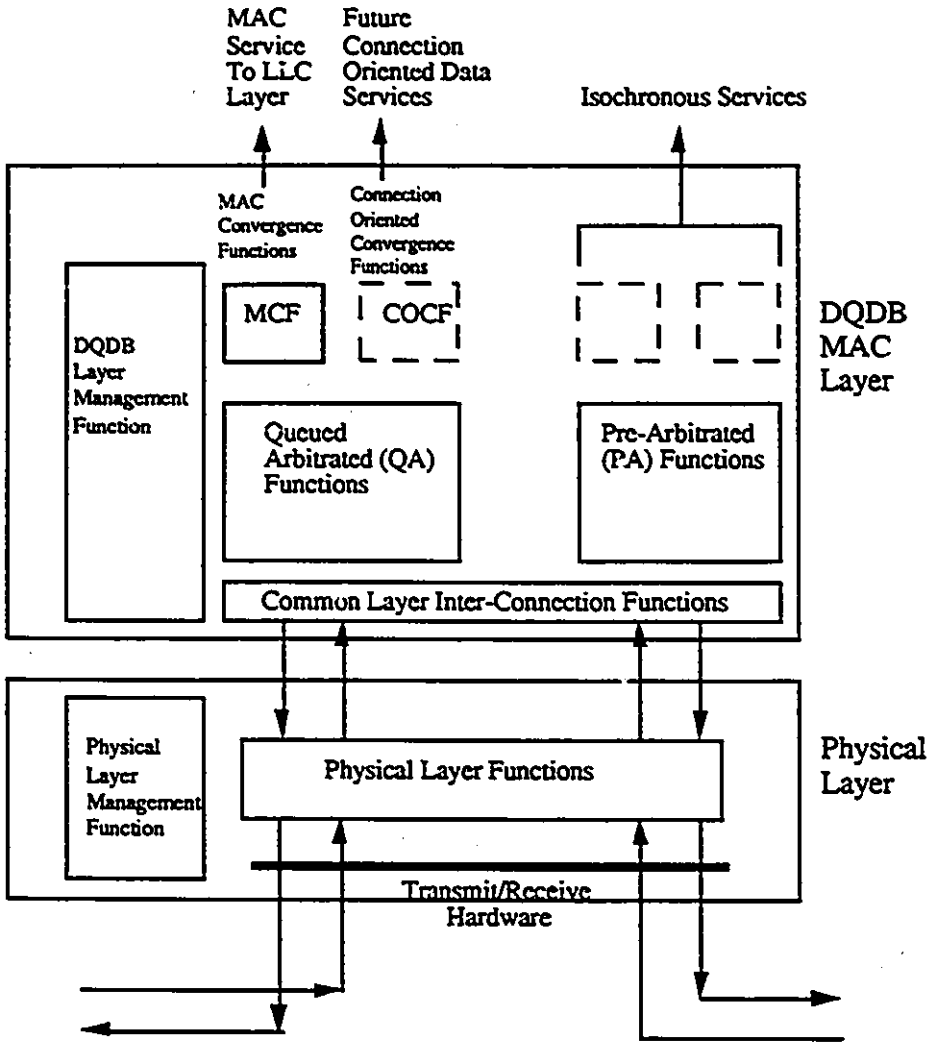


Figure 3.5: IEEE 802.6 MAC and Physical Layer Block Diagram.

The isochronous traffic sources will be dynamically allocated slots from the frame on a "demand" basis to meet their strict time constraints. These slots which are assigned to the isochronous sources are termed pre-arbitrated slots (PA). The PA slots cannot be used for the queue-arbitrated (QA) packet type traffic. The terms QA and PA are the terms used by the DQDB MAN standard and refer to asynchronous and isochronous types of traffic slots generated by the stations on the DQDB network.

The stations will make a request to the head end of the required bus for its required number of 64 Kbps channels to meet its isochronous bandwidth needs. The head end will allocate the desired channels by informing the requesting station which bytes of which slots within the implicit frame it has gained control of. Each byte within each slot of the frame will provide for a 64 Kbps channel because of the 125 μs frame cycle time (see figure 3.4). The 53 byte slot size will allow 48 separate channels to be active within each slot of the frame. The access control field (to be described later) and the header information take up the other 5 bytes of bandwidth within a PA slot.

The allocation of PA slots is carried out at slot generation time using a centralized protocol at the head end of each bus. The proposed standard for DQDB does not impose an allocation scheme as it is assumed that this function will be performed by a higher layer of the communication system. The QPSX proposal currently uses the "first fit algorithm" [NEW88B] but the 802.6 committee did not incorporate this mechanism into the standard as it is out of the domain of the 802.6 standard. The procedure for establishing, maintaining and releasing an isochronous connection is also outside the scope of the DQDB standard, but will be specified by forthcoming IEEE standards[IEE90].

3.1.4 Queued Arbitrated Data Service

This section will describe the original Queued Arbitrated(QA) access mechanism of the QPSX proposal which was renamed DQDB. DQDB was accepted as the prime candidate for the 802.6 DQDB MAN standard[IEE87A]. This original QA access mechanism has undergone several

recent modifications in the standard because of unpredictable performance characteristics displayed during specific heavy load conditions. These modifications will be discussed later in this chapter.

The DQDB MAC layer will be responsibile for supporting the packet oriented data traffic. This packet oriented traffic is referred to as QA traffic within the standard. The term asynchronous is also used throughout the literature when referring to this type of traffic because the inter-arrival time for such data is not at regular intervals.

The QA traffic will be given the use of the remaining bandwidth within the DQDB frame after the PA slot users are assigned their required bandwidth (see figure 3.6). We can see that the QA slot contains 53 bytes of data similar to the PA traffic. 5 Bytes are used for slot control and 48 for MAC layer data.

The DQDB MAC layer will receive the LLC PDU from the Logical Link layer above. This LLC PDU will be passed to the MAC layer as a Medium Access Control (MAC) Service Data Unit (MSDU) within a *MA-UNITDATA request* primitive. This request primitive will be passed to the MAC layer from the LLC layer. These primitives were discussed in the last chapter and are the principal means for interaction between the layers of the OSI model. Figure 3.7 shows a diagramatic overview for the steps taken to convert a MSDU into the QA segments which will be forwarded over the DQDB MAN.

First the MSDU is converted into an Initial MAC Protocol Data Unit (IMPDU). This is accomplished by adding protocol control information to the MSDU. This control information will include: the Common PDU Header, Mac Conversion Protocol Header(MCP), Common PDU trailer and finally, pad information in order to keep the IMPDU size at an integer multiple of 4 bytes (see figure 3.8).

The common PDU header will contain the start of IMPDU delimiter (1 byte) as well as IMPDU size (2 bytes) information to inform the receiver of buffer requirements. The MCP Header will contain the destination and receiver addresses (8 bytes each) as well as 1 byte

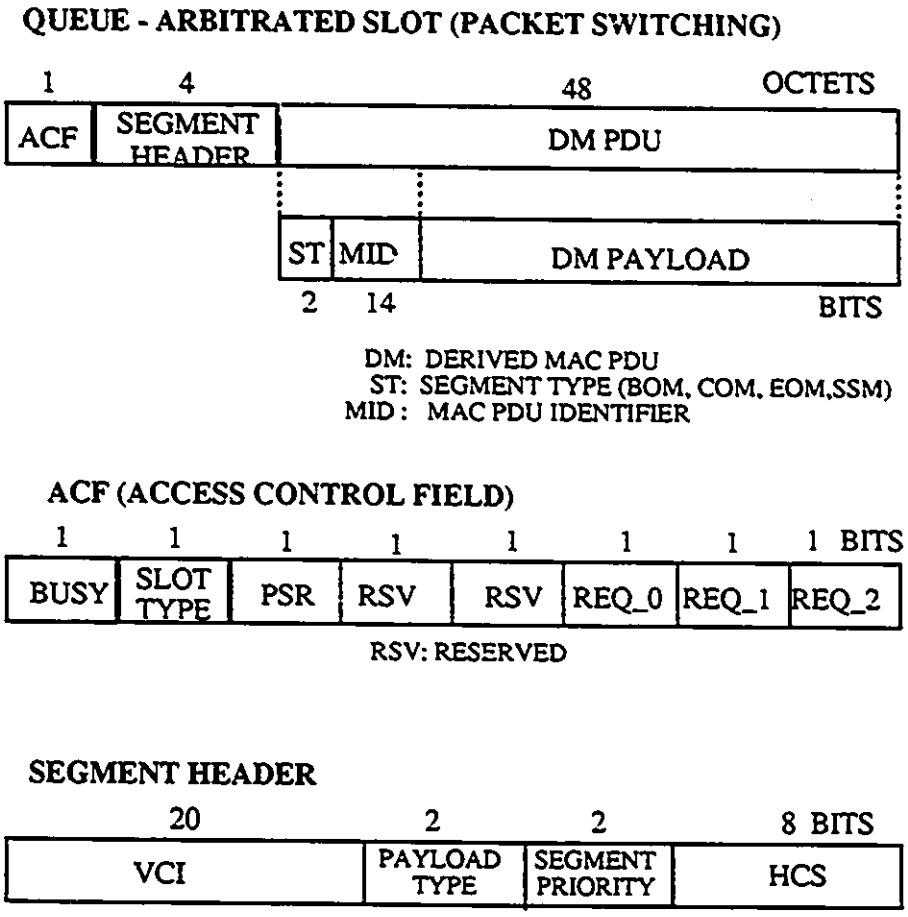


Figure 3.6: QA Slot format including the Segment Header and ACF.

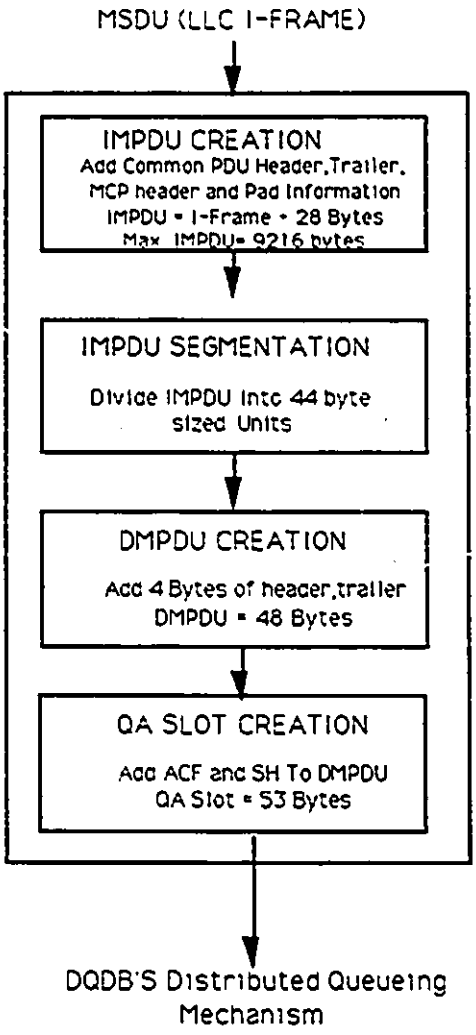


Figure 3.7: DQDB Layer Functions to convert MSDU into QA Slots

to describe the type of LLC protocol used. The MCP header will also be used to detail the quality of service required by the link. The final 2 bytes of the MCP header are set aside for future bridging requirements of the DQDB MAN. The Common PDU trailer will contain an end of IMPDU delimiter which will have the same value as the start of IMPDU delimiter. This will act as a check to ensure the start and end flags are for the same IMPDU. As a second check the size value for the IMPDU sent in the common header will be repeated in the common PDU trailer to ensure data integrity. Padding information will be used to ensure that the IMPDU is an even multiple of 4 bytes to simplify data movement. The length of the IMPDU will vary depending on the length of the LLC INFO field. The maximum length for the MSDU which becomes the INFO field has been set to 9188 bytes which when added to the IMPDU control information will build a maximum IMPDU length of 9216 bytes.

IMPDU- 9216 BYTES (0 Byte Header Extension)

IMPDU Header		Header Extension	I-Frame (Info-field)	PAD	Common PDU Trailer
Common PDU Header	MCP Header				
4 Bytes	20 Bytes	MAX 20 Bytes	MAX 9188 Bytes	MAX 3 Bytes	4 Bytes

Figure 3.8: IMPDU Frame Format.

Now that we have created a proper IMPDU, we must now segment this data unit in-order for it to be forwarded over the DQDB MAN. As discussed earlier the DQDB slot is 53 bytes long, of which 5 are used by the DQDB layer. The remaining 48 bytes are available for the Derived MAC Protocol Data Units (DMPDUs) which will be created from the IMPDU. Each DMPDU will be 48 bytes long of which 44 will be a portion of the IMPDU and the remaining 4 bytes are used for the header and trailer information (see figure 3.9). The header information will label the DMPDU as either the beginning, continuation or end slot of a IMPDU. The header also contains a sequence number and message identification label for reassembly purposes. The DMPDU trailer will contain a length indicator to detail how many

bytes of the IMPDU are contained within this DMPDU (between 1 and 44). The trailer also contains a cyclic redundancy check (CRC) for error detection.

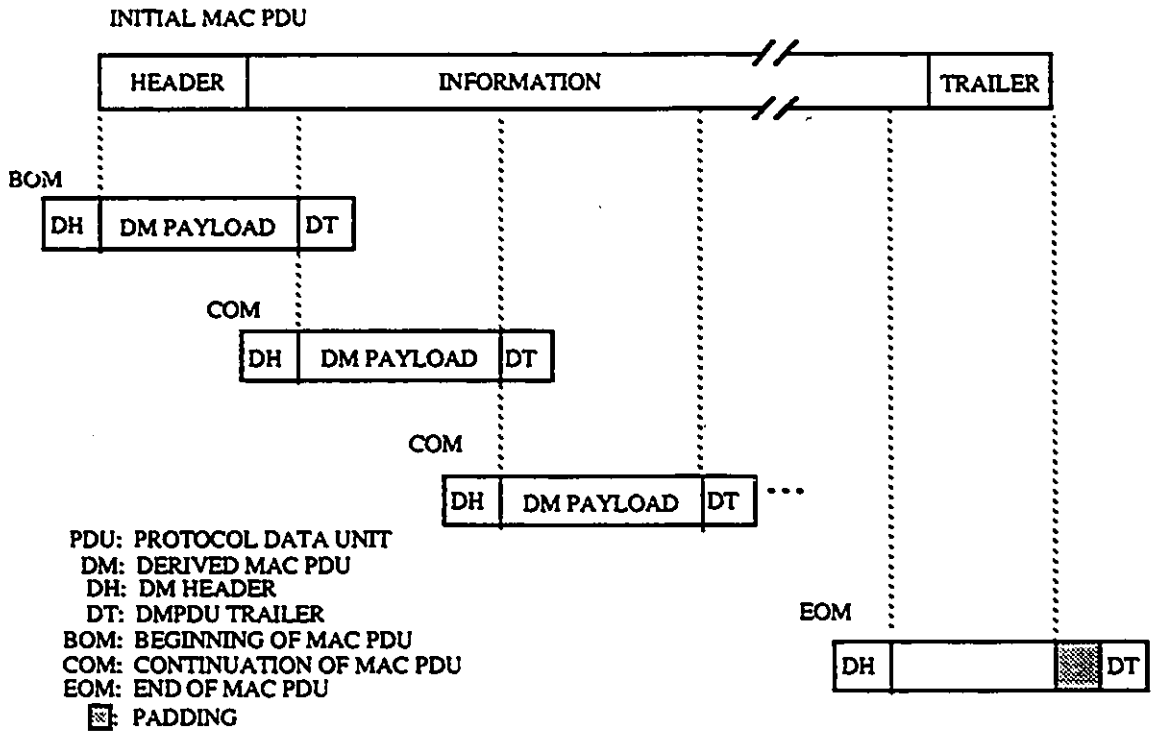


Figure 3.9: Segmentation of IMPDU into DMPDUs.

The final phase of asynchronous QA slot preparation involves the addition of the Access Control Field (ACF) and the Segment Header (SH) to the DMPDU. This will produce the QA slot for onward transmission over the DQDB MAN (see figure 3.6). The ACF is the first 8 bits of the QA slot which contains a busy bit to indicate if the slot is full,and a type indicator to separate the QA and NA slots. The ACF also contains the Previous Segment Read (PSR) bit which will be used to indicate that the slot has passed its destination and maybe reused (if the network has this capability). The final three bits of the ACF are used for the three separate priority levels of requests. One request bit is used for each priority level and the setting of one of these bits indicates that a QA slot is requested at this priority level.

The Segment Header contains a Virtual Channel Identifier (VCI) which provides for a 20

bit label to identify the VCI for each QA slot. There is a single VCI space which is shared by all services on the MAN. The Payload Type and Segment Priority field have been reserved for future use. The final field is the Segment Header Check Sequence (HCS) which is a CRC calculation to verify the SH.

Now that the QA slot has been completed it is ready for transmission onto the DQDB MAN. The Access mechanism involves a distributed queueing protocol which will be described in the following section.

QA Access Protocol of QPSX

After the DMPDU is produced by the MCF it will be forwarded to the QA functions portion of the DQDB layer. This QA functions entity will convert the DMPDU into a QA slot. Once the QA slot is complete it will be passed to the FIFO queuing mechanism. Each station will have control of its own local FIFO queue of QA slots. As mentioned earlier, the DQDB architecture has two separate unidirectional buses. Each station will have to keep FIFO queues for each of the two DQDB buses. The DQDB protocol has incorporated three separate levels of priority into the standard. Each level of priority has its own request bit within the ACF of the DQDB slot. This again will cause division of the QA slots not only between buses but also between priority levels. The end result will be six separate access queues for each station's QA slots. Three queues on each bus at each priority level.

The station is responsible for properly ordering its own FIFO queues. Each station will then be allowed to place one slot from each of its queues into the distributed queuing mechanism of DQDB. The following paragraghs will describe this distributed queuing mechanism for bus A using a single priority level. The description will be identical for all priority levels on both buses.

Figure 3.10 reminds us of the dual bus architecture of the DQDB MAN. The draft DQDB standard relies on 4 bits (1 BUSY and 3 REQUEST BITS) in the ACF to control the dis-

tributed queuing mechanism. A BUSY bit set to one will indicate to all down stream stations on bus A that the slot is being used. A BUSY bit set to zero will indicate that this slot is available for use. The 3 remaining bits of importance to the distributed queueing mechanism are the 3 request bits. One bit for each of the supported priority levels. If a station has a slot to transmit, it will set the appropriate REQUEST bit corresponding to the desired priority level. This request bit will be set in a slot passing upstream on the opposite bus B. This bit which is set to one will inform all upstream stations to allow a free slot to pass on bus A unused.

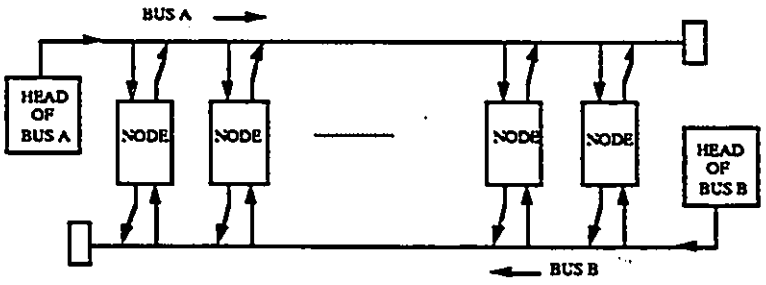


Figure 3.10: Dual Bus Architecture.

Each station's distributed queueing mechanism is driven by a state machine with three possible states (idle, standby, and countdown). A station also requires 3 separate counters (request, countdown, and outstanding requests) to determine the station's current position within the distributed queue.

A station will originally be in the idle state, and while in this state it will increment its request counter for each request bit it sees passing on bus B. This request count is the total number of empty slots which must be allowed to pass unused on bus A to the down stream stations. This request counter will be decremented for each empty slot which is allowed to

pass on bus A. If a slot becomes available for transmission the station will leave the idle state and move to the standby state if the request counter = 0, otherwise it will enter the countdown state (see figure 3.11). As the station moves into the countdown state from the idle state it will set a request bit on the opposite bus B to inform the upstream stations of the need for a slot. The station will also transfer the current contents of the request counter into the countdown counter as it transfers from the idle to the countdown states.

The value which is transferred to the countdown counter is very important because it is the number of empty slots which must be allowed to pass unused before the station may transmit its own slot on to the DQDB bus. In other words this countdown value is the current position of this station in DQDB's distributed queue. After the station transfers the request value to the countdown counter the request counter will reset to zero and begin to count new requests which pass by. The new request information must be obtained as it will be needed to determine the position in the distributed queue for the station's next slot.

If the request counter had been zero when the slot became available for transmission the station would have moved to the standby state instead of the countdown state. In the standby state the station will examine the next slot which passes on bus A. If this slot is free the station will transmit its slot and then immediately return to the idle state. If the next slot is busy the station will move to the countdown state with the countdown counter set to zero. It will also send a request up stream. A third possible event which may occur while in the standby state is that a request is received on the opposite bus B. This again will force the transition to the countdown state and the setting of a request bit. In this case the countdown counter will again be zero (implies this station has queued itself ahead of the request just received).

The general idea behind this distributed queue is to allow immediate access during periods of low network utilization (from the standby state) and to have a single ordered distributed queue for controlled access during times of heavy utilization. This would provide for easy access during low loads and strict operating procedures during busier times.

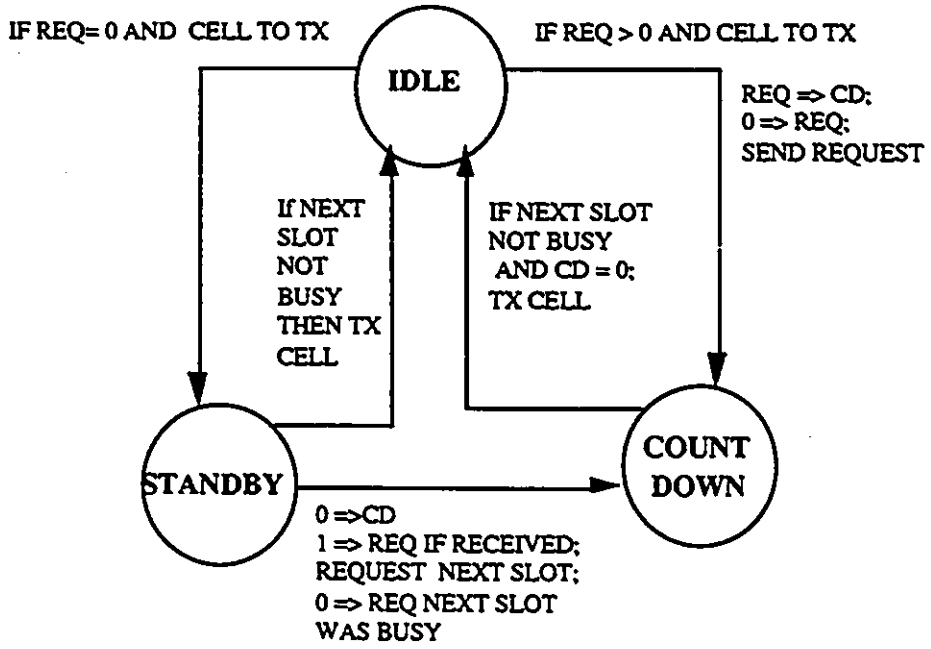


Figure 3.11: State Machine for early DQDB standard. Note: CD = countdown counter; REQ = request counter.

3.2 Early Performance Evaluations of DQDB

The DQDB standard had been accepted in principle as the 802.6 MAN standard. It will be essential to prove DQDB's capabilities to the industry before it could be given final approval as the IEEE MAN standard. Proving these capabilities would involve performance evaluations consisting of simulations, analytical evaluations and possibily test bed results of working versions of the standard.

3.2.1 Ideal Network Performance

Earlier papers which discussed the performance capabilities of this new architecture and protocol showed it to be very promising indeed. Many authors stated quite bluntly that the distributed queueing mechanism would solve all of the perceived problems regarding bandwidth sharing and still achieve 100% utilization of the media. Such comments as "average packet access equal to that of the perfect scheduler"[NEW88B], "predictable queueing under heavy load"[MOL88A], "minimum access delay at all levels of loading up to 100% utilization"[NEW86], and "Packet delays are generally independent of network sizes"[KIM90], all delighted the networking industry.

From these early reports on DQDB's (QPSX) performance capabilities we certainly get the impression that the industry finally had the "all singing and dancing" protocol with little or no delay at light load and the perfect sharing of bandwidth during heavy load conditions. The access protocol would perform like the perfect M/D/1 scheduler over the distributed queueing system. It truly looked as if the networking industry had a major breakthrough, in that a fair shared media access protocol had been devised which would offer 100% bus utilization independent of network speed, size or offered load.

3.2.2 Unfair Bandwidth Distribution during Heavy Loads

As more and more organizations learned that the DQDB proposal was inching closer to becoming the IEEE MAN standard they started to take a closer look at the protocol. It was soon ascertained that, much to the disappointment of the QPSX people, this protocol is not as perfect as it once seemed.

The problems occur when the network is in a heavy load state. This heavy load state occurs when active stations always have slots in the distributed queue and are trying to gain control of the majority of the DQDB bus bandwidth. According to the early performance predictions the DQDB protocol should have divided the bandwidth evenly between all active users. But it has been shown that during periods of heavy load the throughput of each station is very unpredictable and depends greatly on the separation distance between stations and also on the traffic conditions just before the overload occurs[BIS90][FDI90][KAU90][DAVS9][WONS9][VAN90A].

Standby State Removed

The first attempt to improve the performance was to remove the standby state from the DQDB state machine[IEES8A, IEES8B]. The standby state was allowing a single station to obtain a disproportioally large amount of the bandwidth by transmitting slots without issuing a request. This lead to insufficient control information within the network which caused fairness problems.

The problem becomes clear if we consider only two active stations both attempting to gain control of the DQDB bus A (see figure 3.12). The final bandwidth allocation to each station will depend on the initial traffic conditions as well as the separation distance between the two active stations. Let us assume that the stations are 4 slots apart and station 1 has been transmitting for some time when station two becomes active. Station 1 will be transmitting from the standby state without issuing requests. The space-time diagram of figure 3.13 shows

that station 2 will only be able to obtain about 1/10 th of the DQDB bandwidth that station 1 will obtain. The unfairness arises from the fact that the downstream station will have to send a request to the upstream station for each slot it wants to transmit. While the request is propagating up stream, station 1 has control of the bus and transmits slots from the standby state. After station 1 receives the request from station 2, station 1 will enter the countdown state with its countdown counter set to 0 and its request counter set to 1. This means that station 1 will take the next free slot before finally allowing one to pass for station 2. Once station 1 has allowed 1 free slot past it will again resume transmitting from the standby state as the free slot propagates down to station 2. Station 2 will receive the empty slot, send its QA slot and immediately issue another request, and the whole process then repeats itself.

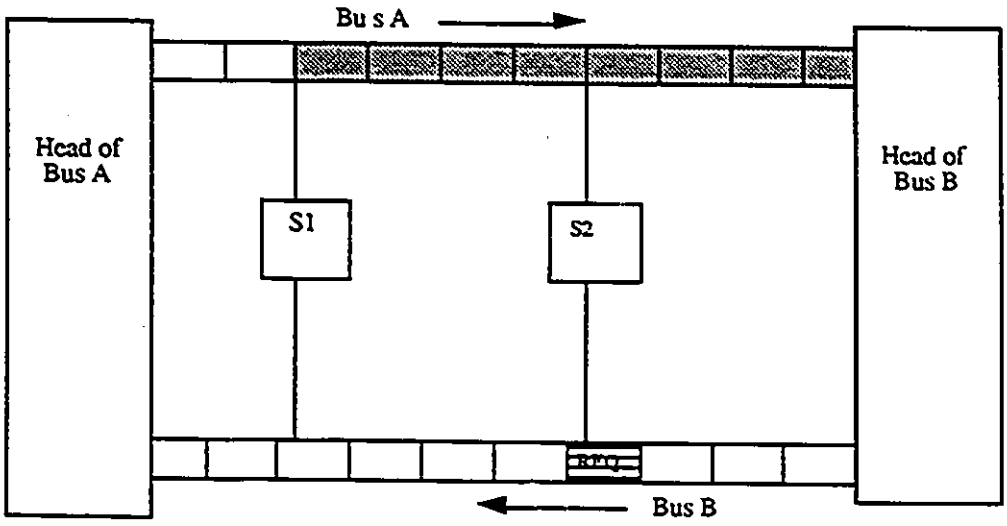


Figure 3.12: Two Heavy User Stations Fighting For Control Of DQDB Bandwidth.

The throughput statistics are easily determined for this example because of the simple initial conditions chosen. Station 1 receives over 90% of bus A's bandwidth while station 2 receives less than 10%. If this was a communication service offered by one of the carriers, station 2 would not be a happy customer. It should be noted that station 2 would get equally bad treatment if it was the station which was initially transmitting from the standby state. Station 2 would be transmitting from the standby state and thus no requests would be received

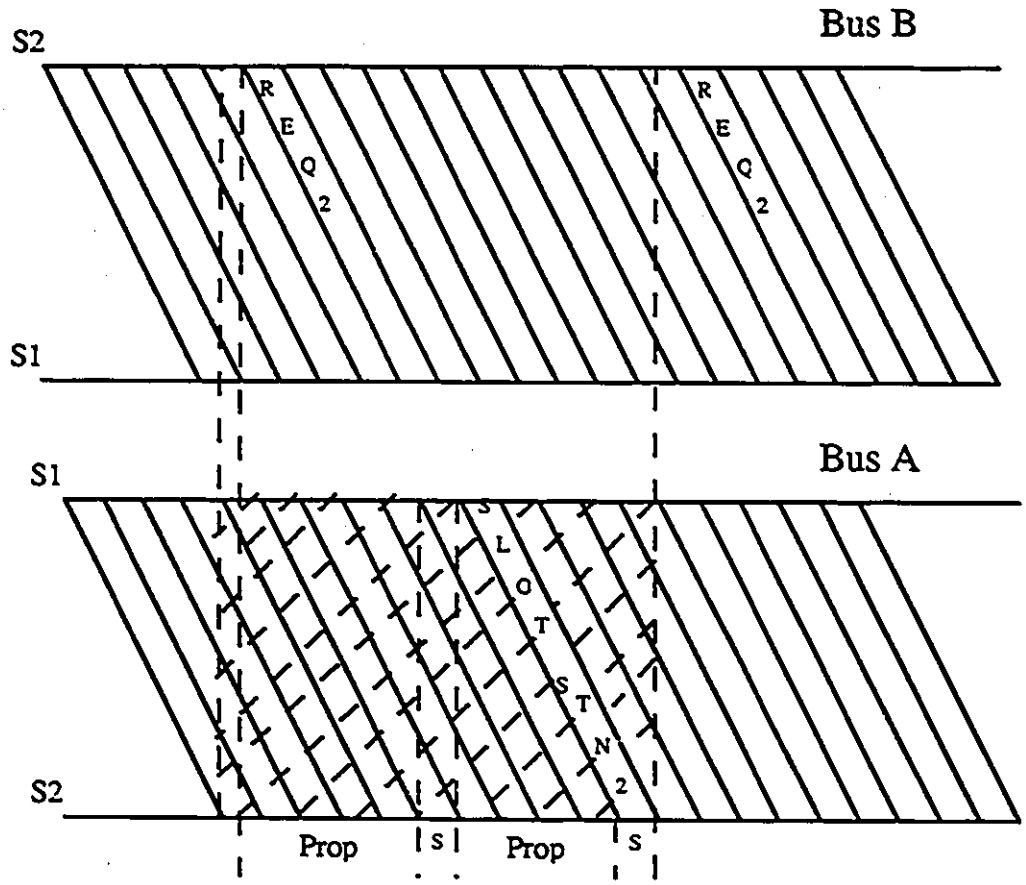


Figure 3.13: Space Time Diagram To Explain Unfair Bandwidth Allocation.

by station 1 to increment its request counter. When station 1 begins transmitting it would be unaware of station 2 and immediately obtain over 90% of the bus bandwidth.

The unfairness gets more complicated as we increase the number of stations and vary the number of slots between stations and also vary the ratio of busy to empty slots for initial conditions. The end result is still very unpredictable behaviour[WON89]. To correct the situation the IEEE 802.6 committee decided in the spring of 1989 to remove the standby state and force all stations to send a request with each slot transmission[IEE89]. This would provide for more control information to be spread throughout the network and thus network stability.

Performance Problem Persists

The removal of the standby state from the standard would prevent a station from cycling from the idle state to the standby state, sending its slot, and then returning back to the idle state. All of this would occur without the station sending any request information upstream. The station must now move directly to the countdown state and issue a request with each slot transmission. It was promptly shown that this modification only managed to make the unfairness more complex and in certain cases more extreme[VAN90A].

It was shown in Van As's paper [VAN90A] that drastic unfairness within the DQDB standard endures. The source of the problem is that the individual access queues at each station, which co-operate to form the single distributed queue across the entire MAN, must be updated simultaneously in-order to achieve the fairness desired. The simultaneous updating of control information must occur within 1 slot time network wide. This implies that the maximum bus length of any DQDB network must be no longer than 1 slot time. The distance of this slot time will depend on the signal speed within the particular media. For a fiber optic system at 155 Mbps the slot distance will only be 546 meters. If the size of our network becomes longer than 546 meters the access queues at each of the stations will not be consistent. This inconsistency leads to a very complex network behaviour which is based on the network

speed-size product and the bus load profile at the instant overload occurs.

An interesting observation which was made in Van As's paper, displayed how the modified DQDB protocol may exhibit an inverse priority behaviour. A high priority transmission can actually result in a station having less throughput. The authors concluded that the DQDB standard as it then stood was very unfair in bandwidth allocation and was very unpredictable during heavy load situations. The DQDB system loading would have to be reduced to below 60% utilization in order to avoid these undesirable properties. If this was accomplished it would display little or no advantage over a slotted bus structure with no reservation system operating at a 60% utilization[VAN90A]. These observations bring into question the validity of such a complex system.

Bandwidth Balancing Introduced

The final major modification to the standard was based on a paper by Hahne et al. This paper introduced the concept of bandwidth balancing[HAH90]. It has been determined that the large propagation times of the control information within DQDB networks cause the distributed queue to perform unfairly for certain stations on the bus. The problem becomes extreme as the bus utilization approaches the DQDB maximum capacity.

The 802 .6 standardization committee accepted a proposed modification to the standard which introduced the bandwidth balancing mechanism. This mechanism is solely intended to equally distribute the DQDB bandwidth[IEES9B] between active stations. It defines a bandwidth balancing counter (BWB_CT) and a bandwidth balancing modulus (BWB_MOD). Each station has two BWB_CT's, one for each bus. These counters will keep track of the number of slots sent on each bus by incrementing its value for each slot sent. The BWB_MOD is a system parameter which states the maximum number of slots a station may send on each bus before the station is forced to let a slot pass by unused. While the BWB_CT is incremented with each slot transmission its value is compared to the BWB_MOD, when the two are equal the BWB_CT is reset to zero and the next slot which the station would normally take for

itself is permitted to pass unused. The idea being that other stations will use the slots which were permitted to pass unused and in turn generate more requests to more evenly distribute the DQDB bandwidth.

The BWB_MOD has been incorporated as a system wide parameter within the DQDB standard [IEE90]. Its value may range from 0-64. If the value is 0, this indicates that the bandwidth balancing mechanism is disabled, otherwise the value is chosen between 1 and 64. This value once chosen will be the value used by all stations on both buses. The standard calls for the default value of the BWB_MOD = 8.

An obvious concern with this change to the standard is the wasted bandwidth which is caused when stations are forced to let slots past unused. It will be shown that in reality this wastage is not extreme . The most wasteful situation is when only one station is active on a bus. This station is forced to let one slot pass free for every eight it uses. This will cause $\frac{1}{9}$'th of all slots (11%) to be wasted. Each active station should receive $\frac{M}{M+1}$ of the slots not used by the other stations. The steady state average throughput per station will be :

$$THRPT = \frac{1}{N + \frac{1}{M}} \tag{3.1}$$

which will provide for a DQDB average bus utilization of:

$$THRPT = \frac{N}{N + \frac{1}{M}} \tag{3.2}$$

where:

M = System wide Bandwidth Balancing Modulus

N = Total Number Of Active Stations

From these equations we can see that the wastage will not be too severe, but 100% bus utilization is not possible for DQDB MANs with bandwidth balancing enabled. The bus utilization will improve with larger values for the BWB_MOD and as the number of active stations increases. It is important to note that the convergence towards fairness is not immediate and it takes a period of time for the bandwidth balancing mechanism to provide sufficient free slots to be forced from the active stations to evenly divide the bandwidth. A lower BWB_MOD converges the fairness faster but causes a higher percentage of wasted bandwidth.

It will be shown later in this chapter that by assigning all stations on both buses the same modulus without considering the station's traffic arrival rate nor the traffic destination distribution, may cause more damage to the fairness of DQDB. The BWB_MOD must not be a system wide parameter treating all stations on both buses equally. The station's position and traffic distribution should be considered when choosing the bandwidth balancing modulus for each station.

3.3 Performance Evaluations of Final DQDB Standard

After the DQDB proposal was accepted as the prime candidate for the IEEE MAN, many organizations performed performance evaluations on the standard as it evolved. These performance evaluations by Wong,Van As and Hahne were instrumental in forcing changes to the standard to improve its weaknesses.

Some early performance statistics were obtained analytically [ZUK87][ZUK88A] [ZUK88C] but these were early attempts which resulted in M/D/1 solutions for the DQDB standard. An analytical result for two users was developed by Wong[WON89] which clarified the problems of the standby state. Bisdikian's work[BIS90] analytically investigated a specific network user not under heavy load conditions. His work displayed how complex and involved an analytical study of the DQDB network can be. Bisdikians's work did not consider the Bandwidth Bal-

ancing modification to the standard which will only complicate the study further. The latest publications seemed to have abandoned the analytical approach. Conti's latest paper[CON91] released in January of 91 stated "The high degree of interactions among a plethora of processes that make the exact analysis of the network almost impossible". I believe that an analytical solution for the DQDB network will eventually be developed but I believe it to be beyond the scope of this thesis. I will perform my study of the DQDB network using the QNAP2 simulation package[QNA84]

3.3.1 Simulation Study of DQDB performance

In this section a performance study of the most recent version of the DQDB standard will be presented. The model for this initial study will be used as the basic platform for more complicated studies given in later chapters. The study will focus on the ability of the DQDB protocol to evenly divide its asynchronous bandwidth between all active stations. The study will also investigate the fairness of the protocol to treat all stations equally when sending QA slots onto the DQDB MAN. The study will consider the isochronous traffic as a background load on the MAN. The DQDB implicit frame will have a fixed portion allocated to the PA traffic. The remainder of the frame should be evenly distributed between all active users for their QA slots.

Model of DQDB MAN

Figure 3.14 gives a visual description of the model used to simulate the DQDB distributed queuing mechanism. The simulation was performed with the QNAP2 simulation package. The QA and PA traffic slots are represented as customers within the queuing model. The slots will be generated according to the 125 μs frame cycle time set out within the standard specifications. The QA and PA slots will be passed down the bus from station to station. The filling of the QA slots with station data will be carried out according to the DQDB protocol standard. This study will only consider a single level of priority as this thesis will primarily

deal with system fairness. All stations will have the same priority when trying to access the DQDB MAN.

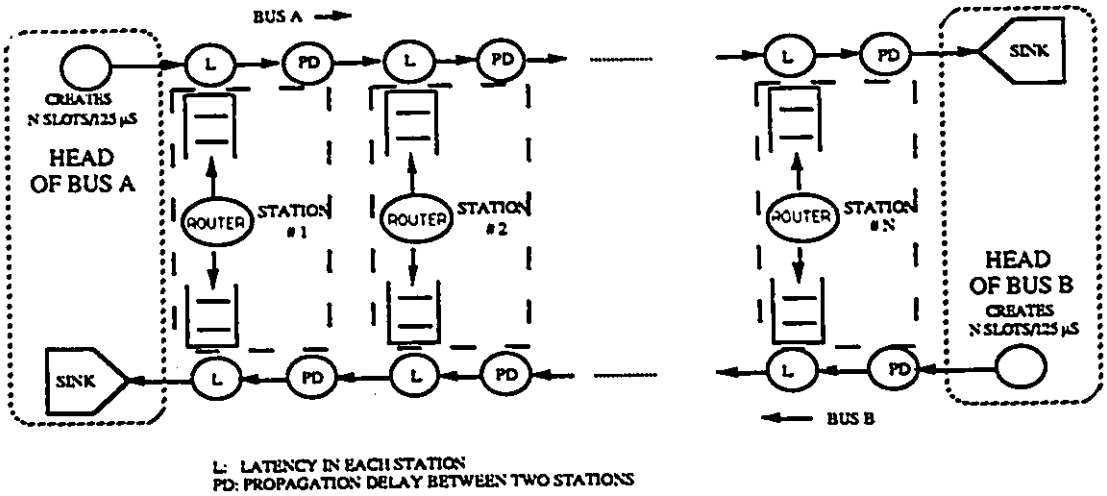


Figure 3.14: Basic Simulation Model for DQDB MAN.

To properly simulate the DQDB distributed access mechanism the slots which are passed from station to station must contain the following information:

REQUEST BIT = To be used by downstream station to indicate a need for an empty slot.

TYPE BIT = To differentiate between PA and QA slots within the 125 μs frame

BUSY BIT = to indicate if the QA slot is available for data transmission

The simulator must maintain the following critical information to determine when each station may place its QA slots on to the DQDB bus.

REQUEST counter = Total number of request bits which have been received from downsteam stations.

COUNTDOWN counter = Total number of slots the station must [illegible] to pass unused before it may access the bus.

BWB_CTR = Total number of slots sent since the last free slot was forced to pass by unused.

BWB_MOD = System parameter which will decide how many slots may be used by a station before a free slot is forced to pass unused.

The stations will act in accordance with the state machine defined in the draft version of the DQDB standard. The standby state has been removed by the standards committee and thus, all stations will cycle from the IDLE to the COUNTDOWN state and back to the IDLE state as it actions the events within the network.

Assumptions

Similar to other performance studies of computer networks, certain assumptions where made for simulation purposes. The values for the model parameters have been chosen by considering the recommendations stated in the draft DQDB standard[IEE90]. The DQDB bit rate is fixed by the number of slots in each implicit frame of 125 μs. Each slot insertion (extra 53 bytes) requires an additional bit rate of 3.392 Mbps from the carrier. Fot this study we use a DQDB frame length of 12 slots with a required bit rate of 41 Mbps (compatible with DS3 circuits). One of these slots is reserved to support PA traffic communications between isochronous traffic generators. The actual PA traffic is not generated but a background load is assumed to use the one PA slot per frame. The PA slots must be produced by the simulator for this study as they carry control information in the form of requests. This control information is required to accurately simulate the QA slot access mechanism of DQDB.

The remaining 11 slots are available for transferring data packets on to the DQDB MAN as QA slots. We assume a dual bus length of 25 km. The 10 access nodes are to be equally distributed across the network and their latency delay is equal to one byte (600 ns). The propagation delay of the medium is assumed to be 5.085 microseconds per km, as the use of optical fibre is assumed.

We are interested in determining the fairness behaviour of the DQDB MAN while supporting these 10 stations, each with equal access priority. The load on the DQDB MAN will be varied from as low as 50% and up to as high as 99% of the DQDB capacity. It is assumed that each station will provide $\frac{1}{10}$th of the DQDB offered load. The traffic arrival rate is exponentially distributed with the mean value chosen to meet a specific network load. All stations should observe roughly the same delays while gaining access to the DQDB MAN. The DQDB standard calls for the bandwidth balancing mechanism to be enabled and the bandwidth balancing modulus to be set to 8 for all stations on both buses. These are the default settings stated within the DQDB standard.

The 10 access nodes are equally distributed along the dual bus architecture. It is assumed that each station will attempt to communicate with all other stations with an equal probability. The amount of traffic destined for each bus will be a function of the station's position along the bus. The stations at each end of the bus will send 100% of their traffic on the downstream bus. Stations located in the centre of each bus will send about half their traffic on each bus. The remaining stations will determine the proportion for each bus based on its relative position on the bus.

This assumption of uniform traffic distribution is one which was simulated by the vast majority of all publications regarding DQDB. I personally do not think that it is realistic to assume a uniform MAN traffic distribution pattern. Later chapters will analyze what I believe to be more realistic traffic conditions.

3.3.2 Analysis and Results

The average access delays displayed by figure 3.15 give an indication of the delays experienced by each of the stations on the DQDB MAN. The goal of this first examination of the standard's performance was an investigation of the fairness of the standard in equal treatment of all stations on the bus.

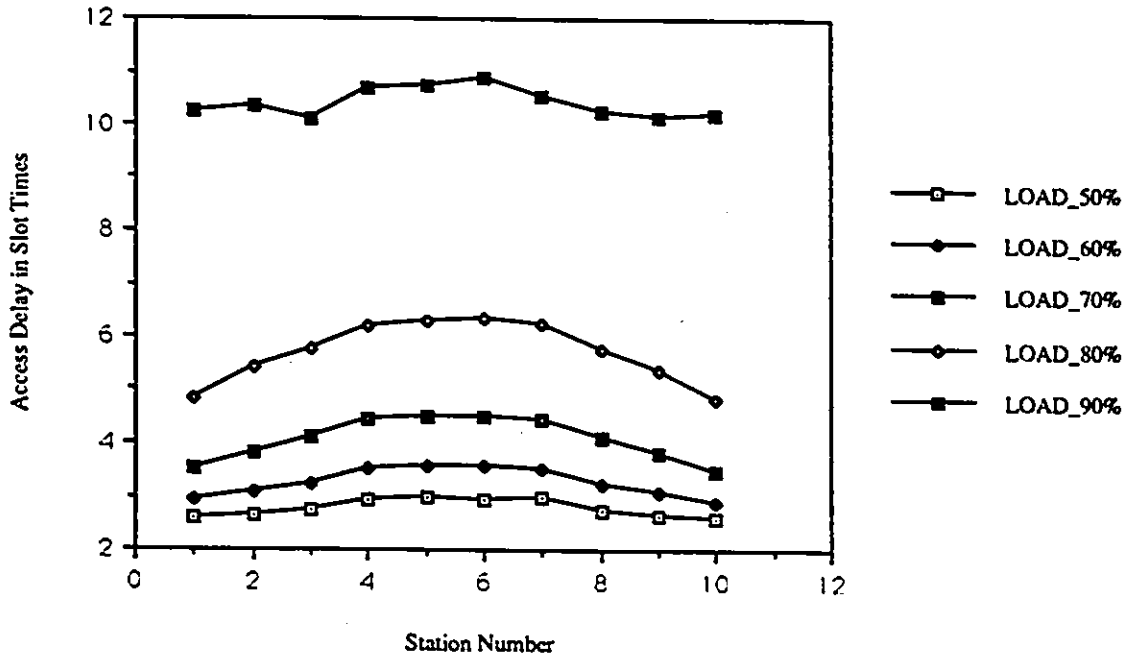


Figure 3.15: Access delay for 10 Stations with a Uniform Traffic Distribution. The load is varied from 50% to 90% of the DQDB capacity..

Figure 3.15 shows that indeed the standard does treat all stations quite fairly during periods of light to medium load (up to 90% DQDB MAN utilization for this experiment). At load levels below this value we observe the end stations enjoying the advantage of faster access times. This observation has been documented throughout the history of the DQDB MAN. This is the result of the end stations in periods of light to medium loads gaining faster access because they do not have to wait for a request to be actioned upstream. They are the most upstream stations. The stations in the middle of the bus must wait for their requests to flow upstream, which will then force all upstream stations to allow free slots to pass. This delay in the requests moving to the head of the bus is referred to as "SKEW". This is the unfair behaviour discussed earlier and is caused when a DQDB bus length is larger than one slot time.

During my study at 80% MAN loading I obtained a network mean access delay of 5.69 slot times. The fastest station (#1) had an average access delay of 4.82 slot times and the slowest station (#6) had 6.32 slot times. These results are compatible to results which have been reported in DQDB performance literature.[NEW88A, CON89, FIL90, FDI90, HAH90, GAR90, KAU90, KIM90, WON89, ZUK88B]

The situation changes dramatically as we load the DQDB MAN network above the 90% value (see figure 3.16). We start to see that the end stations lose their advantaged access delay values and as the load approaches maximum capacity (about 99%) we observe that the end station's access delays rise to a point of being extremely unfair (station #1's 3419 slots compared to station #5's 26 slots). This observation is quite surprising as it has not been reported previously. In fact, I had thought that I had a problem with my simulation program which caused it to malfunction at extreme network loads. After many hours of examining both my results and the simulator I determined that the results were indeed valid. A recent study which has been released on the performance of DQDB [CON91] agrees with the results I have obtained. The study by Conti et. al. noted that the end stations did not enjoy the use of both buses. Conti did not seem to think that it adversely affected the throughput of the end stations, even though he observed a packet loss rate above 40% for the end stations.

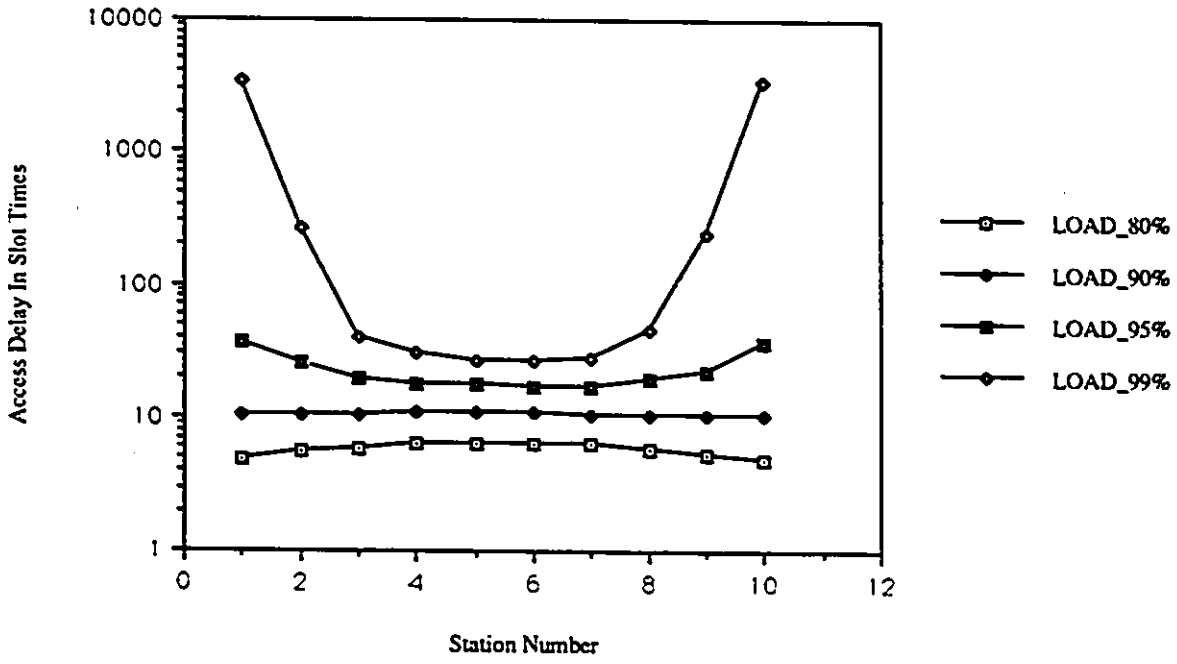


Figure 3.16: Access delay for 10 Stations with Uniform Traffic Distribution With Heavy DQDB MAN Loads.

The end stations dramatic change from most advantaged station at light to medium load to most disadvantaged during periods of heavy load results from several separate aspects of the DQDB architecture. First, the end stations are located at the head end of each bus,and as such they must action every request issued downstream. The very thing which gave it the advantage of not having to wait for the upstream stations to action requests during light traffic is now working against the end station. As the bus load increases so does the number of requests. With the removal of the standby state all stations are forced to send a request with each slot transmission, and also the PA slots may carry requests upstream. The end stations are saturated by all the requests. Each request arrival will force the end station to allow a free slot to pass unused. The end stations spend so much time letting free slots pass they do not obtain sufficient bandwidth to service their own needs.

The second disadvantage which crops up during periods of heavy load for the end stations is the dual bus architecture itself. Figure 3.17 shows how the end stations are forced to send all their traffic on to a single DQDB bus, where as the middle stations enjoy the privilege of using both buses equally. Since each bus is treated completely separately by the standard, the middle stations enjoy what is effectively twice the service rate observed by the end stations. Basic queuing theory will tell us that the middle stations will enjoy very low access delays compared to the end stations during periods of heavy bus utilization.

The bandwidth balancing mechanism which was introduced to correct the unfairness of the DQDB protocol effectively did just the opposite. The standard calls for all stations to have the bandwidth balancing mechanism enabled with the modulus set to 8. This attempts to treat all stations equally. As we have just shown the end stations do not enjoy the luxury of transmitting on both buses. The DQDB dual bus architecture forces the end stations to use only one bus, and the DQDB access protocol is forcing the end stations to surrender $\frac{1}{9}$th of their already unfair bandwidth allocation. This equal treatment of all stations increases the unfairness for the end stations.

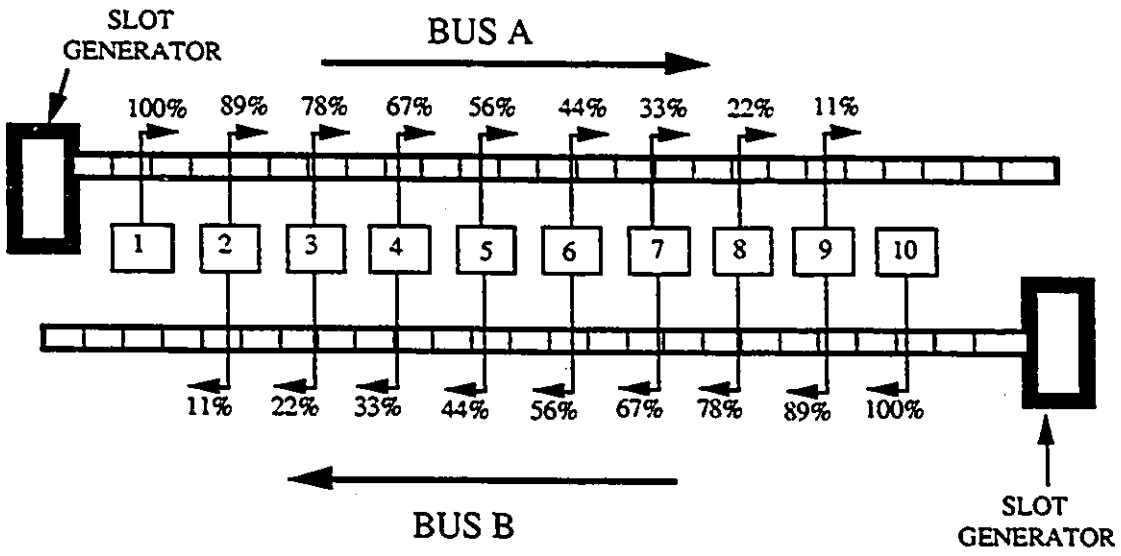


Figure 3.17: Unfair Bus Access For Stations Which Only Utilize One Bus.

3.4 Conclusion

This chapter provided a detailed description of the DQDB architecture and protocol which is in the finishing stages of development. The standization committee had originally thought that the QPSX proposal would meet all the functional requirements set out in the last chapter. It was soon discovered that the original protocol was unfair in properly distributing the DQDB bandwidth between active users during heavy loads, and in general, was very unpredictable.

The DQDB standard has undergone several major modifications to correct the discovered shortcommings. These changes and the reasons for the changes were discussed in this chapter. The question raised in the last chapter of how much control should be kept over a netwok was brought up in this chapter. The protocol in its early forms had too little control over the network. The removal of the standby state (to increase request information) and the introduction of the bandwidth balancing mechanism, added to the control over the network. Both were intended to avoid unpredictable network behaviour. The debate arises again as to how much control do we force on a network before the control mechanisms become to costly.

My performance study in the final portion of the chapter showed that we have some room to improve on the fairness of the DQDB standard. With the introduction of the bandwidth balancing mechanism and the removal of the standby state the end stations do lose their unfair advantage. My study has indicated that the most advantaged stations quickly become the most disadvantaged stations as the traffic on the DQDB MAN approches maximum capacity.

This observation has far reaching implications. The major cause for the unfairness towards the end stations is their unidirectional traffic patterns. The question arises, what if stations other than the end stations have a unidirectional traffic pattern? What will their traffic statistics be? The next two chapters will investigate various community of interest based traffic conditions which are supported by a DQDB MAN. The focus of these chapters will be DQDB's ability to treat all users fairly.

Chapter 4

DQDB Private MAN interconnecting a Three Token Ring Community Of Interest

The previous chapter gave a detailed description of the IEEE 802.6 DQDB MAN's architecture and protocol. A basic performance study was also done to determine if the standard in its present form was fair in dividing the DQDB bandwidth between active stations. It was shown that during light to medium loads the standard is relatively equal in treatment of all stations but still slightly favouring the head end stations. As this favouring of the head end stations occurs during lighter traffic conditions the middle stations still receive adequate treatment. The situation dramatically changes during periods of heavy load as the end stations experience extremely unfair treatment by the DQDB standard which causes extremely large access delays for the end stations QA slots to access the DQDB MAN.

This chapter will investigate the fairness and performance behaviour of a DQDB MAN interconnecting three IEEE 802.5 Token Ring networks (see figure 4.1) to form a metropolitan area communications network. Similar types of studies have been performed previously[BUX85, VAL89] but this study's focus on the fairness and performance aspects of the latest version of the DQDB standard to support community of interest based traffic makes it quite unique.

The focus of this chapter will be to investigate the performance of the LLC protocol working in conjunction with the DQDB MAC sublayer to support the end to end communications of

stations located on IEEE 802.5 token rings. This three token ring system would fall into the category of a private MAN, as it services a single community of interest. This private MAN could be owned by a major government department (eg Department of National Defence) located within the Ottawa area. This government department might use a DQDB based private MAN to interconnect its three separate token ring LANs which are spread across the city.

4.1 Application Overview

When several computer stations are distributed over several large buildings or across a city, it might be more efficient to group these stations into smaller separate networks and to connect these networks by a high-speed backbone network. The Metropolitan Area Network (MAN) is defined to accomplish this task. A MAN will interconnect local area networks (LANs) through bridges which will have to provide various protocol services to ensure proper network inter-working. The IEEE P802.6 Working Group is in the final stages of completing work on the DQDB MAN which will provide these required interconnection services for LANS [IEE90].

I am particularly interested in studying the throughput and fairness characteristics of a communications network using DQDB as the backbone. The DQDB backbone will interconnect, via bridges, three token rings which operate as specified in the ANSI/IEEE 802.5 standard[ANS85D]. The bridges must perform segmentation and reassembly, as the interconnected networks (Token Ring and DQDB) do not use the same data unit sizes (different overhead and maximum packet sizes[SHO79]). In case of congestion, the bridges, having limited buffer size, will discard frames they cannot handle momentarily[WON90, BUX85, GER80, GER88]. To perform the recovery of lost frames, I consider a network architecture in which type 2 of the IEEE 802.2 Logical Link Control (LLC) standard[ANS85A] is employed as the end-to-end protocol. This will reduce the packet processing time in the bridges.

The issue of fairness for DQDB based systems has generated a great deal of interest

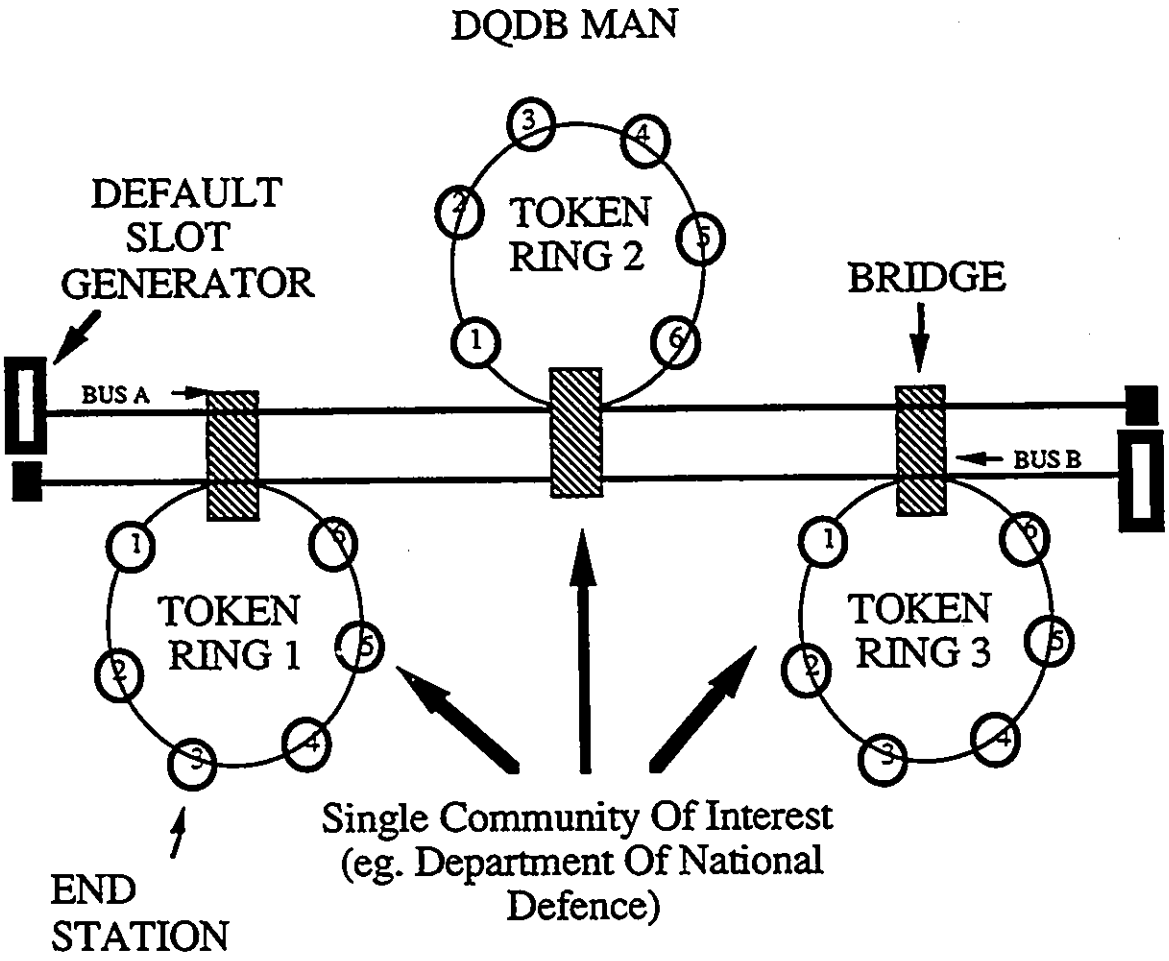


Figure 4.1: Three Token ring LANS of Major Government Headquarters to be interconnected.

within the networking industry and was discussed in the last chapter. This fairness problem arises when one or a group of stations receives a disproportionately large allocation of the DQDB asynchronous bandwidth. This problem was encountered during the study of a private MAN interconnecting three tokenrings and will be discussed. A solution to the unfairness encountered in our study was to use variable bandwidth balancing (V_BWB), which is a modified version of the proposed bandwidth balancing mechanism for DQDB.

In the next section I describe the different elements of the inter-work system. In section 3, I present the performance study of short messages transferred between stations through the proposed interwork system. The emphasis is to measure the system efficiency and fairness for transferring I-frames between stations on different token rings through the DQDB MAN. A proposed solution to the observed unfairness will be presented. The final section will highlight the significant observations of this chapter as well as suggesting how this particular study may be expanded.

4.2 The interconnection System

In this study, I use DQDB as a backbone network interconnecting three token rings comprising six stations each (see figure 4.2). Bridges are used to interconnect the token rings with DQDB. Communications are assumed to be pre-established between pairs of stations situated on different token rings. The traffic destinations of all traffic leaving each ring is split evenly between the other two rings. This will provide for a uniform traffic distribution throughout the entire system. I will now present in detail the elements of the interwork system.

4.2.1 DQDB Architecture

The DQDB architecture and protocol aspects of this study are based on the IEEE 802.6 specifications set out in the DQDB standard[IEE90] and thoroughly discussed in the last chapter. The dual bus architecture and frame timing will be identical to the study performed

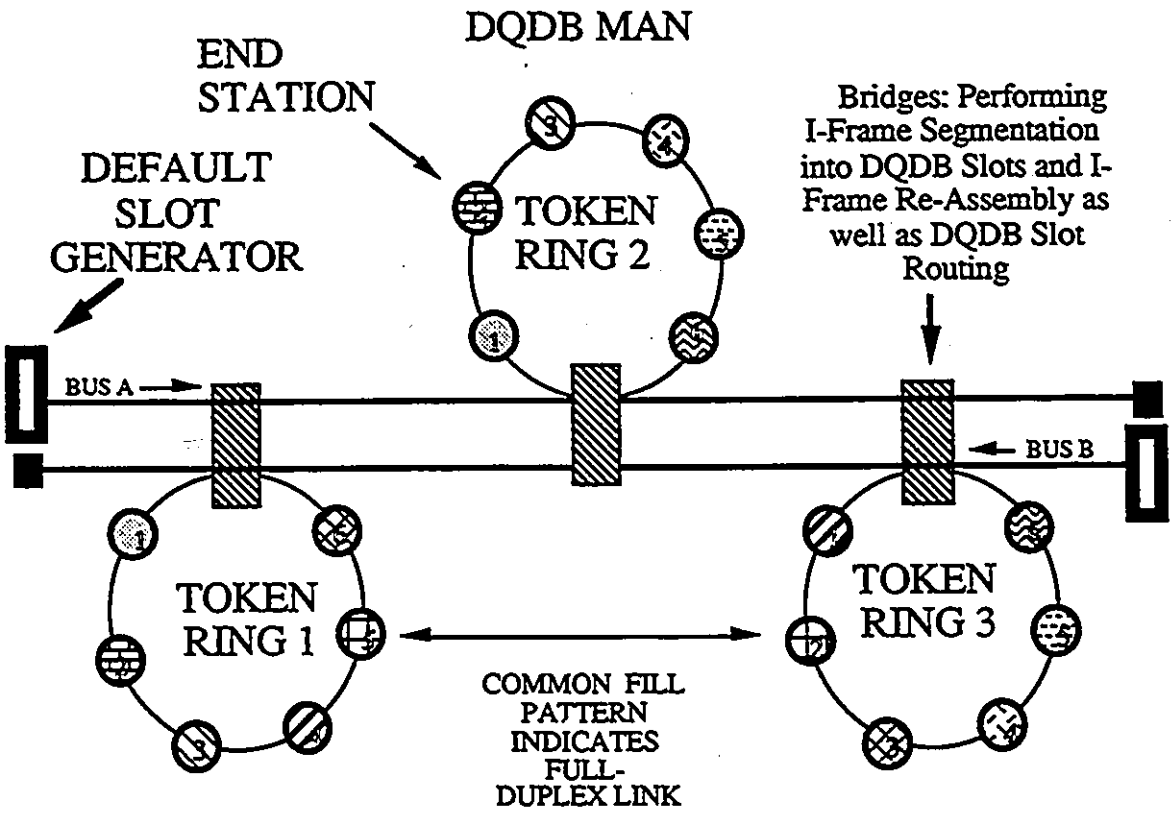


Figure 4.2: The IEEE DQDB MAN configured as a Private MAN supporting Three Token Ring LANS.

in chapter 3. The architecture of DQDB comprises two unidirectional contra flowing buses. Each bridge connecting the LANs to the MAN is attached to both buses which allows for full duplex communication between pairs of stations on the token rings.

The DQDB bandwidth is dynamically allocated to meet the demands of the pre-arbitrated (PA) traffic. Capacity which is not allocated to the PA traffic can be shared by the packet switched queue arbitrated traffic (QA). Non-isochronous (QA) slots are used to transfer asynchronous segments previously generated by a longer message segmentation. The 53 octets of a QA slot provides for 44 octets of user information. The remaining octets are allocated as follows: 1 for the access control field, 2.5 for the virtual channel identifier (VCI) representing a logical link, 1.5 for service and segment header checksum, 2 for segment type and message identification, and 2 for payload length and CRC check.

There are 4 types of QA segments: the beginning of message (BOM) segment, the continuation of message (COM) segment, the end of message (EOM) segment, and the single segment message (SSM), which conveys messages whose length is shorter than the maximum segment payload length(44 bytes).

4.2.2 Token Rings

The IEEE 802.5 standard [ANS85D] was discusses in chapter 2 with the other 802 LANS. The standard defines a priority mechanism which uses eight levels of priorities. I studied two modes of priorities: first the non-priority mode in which all stations and the bridge on the token ring have the same ring-access priority and can only send one frame per token arrival (non exhaustive service). The second priority mode assigns priority to the bridges. Whenever the bridge has a frame queued for transmission, it will either seize the low priority token, or make a reservation in a passing frame for a high priority token. The reservation will force the currently transmitting station, after having sent its frame, to issue a priority token which will be then used by the bridge. After the bridge has completed transmission, the low priority token is granted to the next station downstream from the one which last transmitted.

In this priority mode, bridges are allowed to send all frames queued in their MAC layer on to the token ring, the idea being that the token ring will empty its buffer faster and have less packet losses. The other stations continue to be limited in sending just one frame each time they capture the token (non- exhaustive service). The IEEE 802.5 committee has finalized the standard for the 16 Mbps Token Ring[GLA89]. I shall consider this 16Mbps speed as the capacity of our token rings feeding the bridges.

4.2.3 Bridge

To connect the DQDB MAN with the token rings, I defined bridges (see figure 4.3) which are composed of two independent processes. One process is used to forward the token ring frames on to the DQDB MAN. This will involve converting a token ring frame into a DQDB MAC PDU (Media Access Control Protocol Data Unit) and for segmenting this DQDB MAC PDU into multiple derived mac protocol data units (DMPDUs) which then will be converted into QA slots (see figure 4.4). The final function of this process will be to route the QA slot to the proper DQDB bus depending on the direction of the destination station of either upstream or down stream.

The second process within the bridge performs the reverse procedure. Each token ring frame will be rebuilt from the separate QA slots. After the complete I- frame has been rebuilt it will be then sent to the token ring MAC queue for transmission on to the token ring LAN.

In order to reduce the processing delay in the bridges, I assume that the bridge does not perform any error recovery action nor flow control functions. If an error occurs during the rebuilding of a token ring I-frame (i.e. QA slot loss) the entire frame will be discarded by the bridge. It will be a function of the LLC layer at each station to detect the I-frame loss and re-transmit the frame. The LLC type 2 (connection- oriented) protocol[ANS85A] is used to guarantee the end-to-end link between stations. Finally, the finite buffer size is equally distributed between the two processes (segmentation and reassembly). Hence, token ring I-frames and DQDB QA slots are discarded when they do not find sufficient free memory upon

their arrival at a bridge[BUXS5, GERS8].

In summary the bridges are most concerned with the efficient routing and movement of the data through the DQDB-LAN interfaces. The error detection and recovery is left to the LLC layer of each station on the token rings.

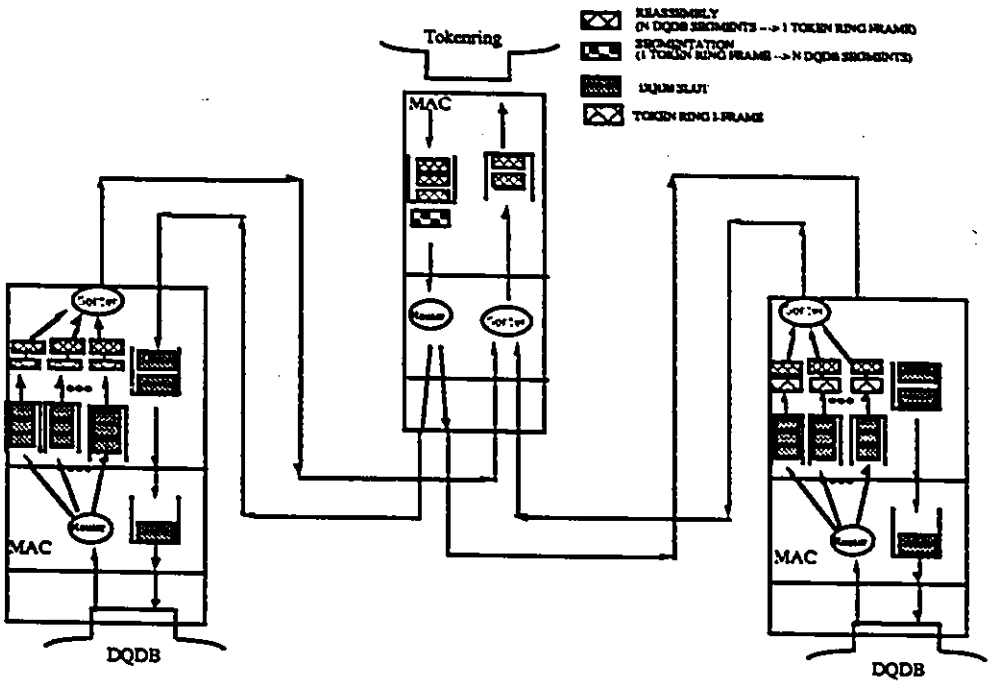


Figure 4.3: The Bridge connecting the DQDB MAN and the Token Ring LANS. The figure displays the DQDB to Token ring Data flow as being two independent processes for clarity. These two functions are actually performed by a single process.

4.2.4 End Stations

I consider 9 pairs of workstations interconnected by full duplex communication links via the token rings and DQDB MAN (see figure 4.2). The communication facilities within each station are based on a layered architecture where the highest layer is the data packet generation application. The LLC layer is located directly below the data generation application. The LLC protocol provides for the end-to-end reliable communication link between end stations.

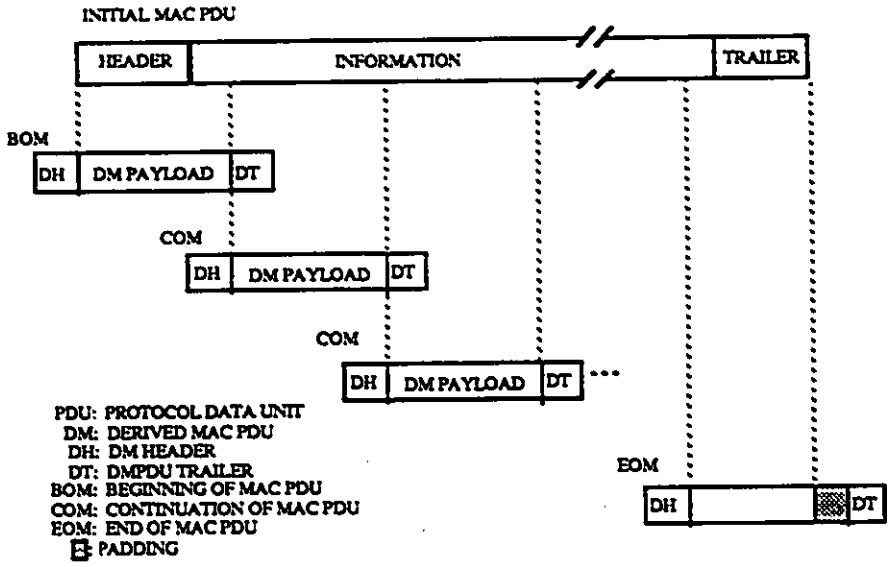


Figure 4.4: The Segmentation of Token Ring I-Frames into DQDB QA Slots within DQDB - LAN bridge.

The LLC type 2 protocol as defined by the IEEE 802.2 committee [ANS85A] and discussed in chapter 2, is based on the use of I-frames for transferring information, S-frames (supervisory frames) for indicating frame acknowledgement (RR-frame) or rejection (REJ-frame), and the P/F (poll final) bit for re-establishing data link connections. Moreover, the LLC protocol assigns a sequence number to each I- frame, which allows the receiver to verify the received frame sequence order and inform the sender if it has correctly received all expected frames. The LLC type 2 protocol uses a limited window size to throttle the generation of packets by the stations in order to reduce congestion in the system. Finally, by using timers, the LLC protocol can re-establish a communication link which has been interrupted. The MAC and physical layers provide core services (frame delimiting and addressing, and detection of transmissions errors) and frames transmissions on to the token rings. Figure 4.5 gives the general architecture of the end station.

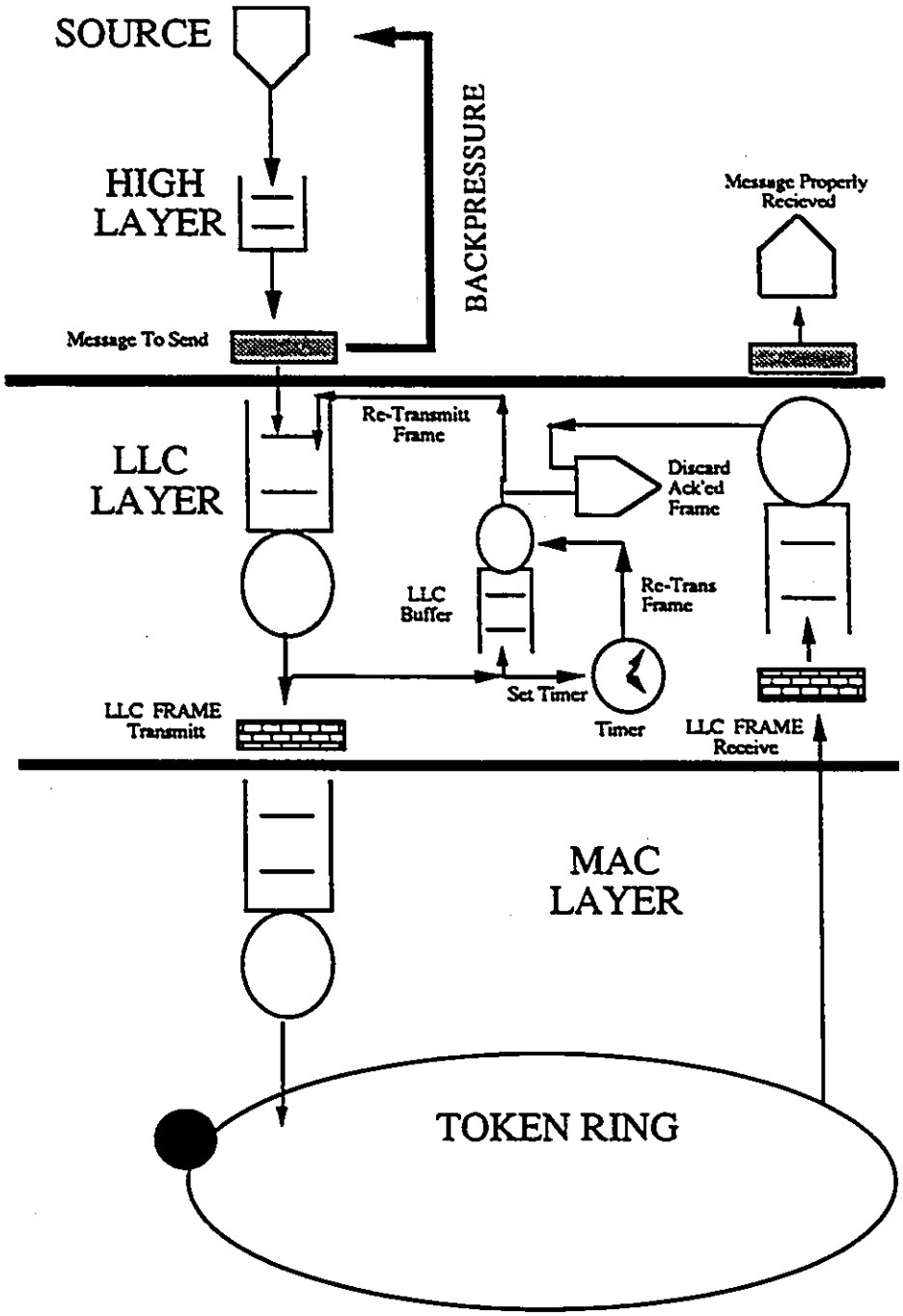


Figure 4.5: The End Station Architecture.

4.3 Performance Study: Packet Transfer Within Private MAN

For this application, I assume full-duplex communication links pre-established between 9 pairs of stations situated on different token rings (see figure 4.2). The 6 stations on each ring are divided into 2 groups, each group will send its traffic to one of the other two token rings. This division will cause 3 stations on each ring to form a duplex connection link with 3 stations on each of the other two rings. Given this type of connection scenario the traffic should be uniformly distributed throughout the DQDB MAN.

The performance study was performed using the discrete-event simulator of the QNAP2 package[QNAS4]. The DQDB portion of the study used two slot generators located at the head end of each bus and produced slots according to the 125 μs timing structure of DQDB. After a slot is generated by the slot generator it would be passed successively down the bus from node to node. The nodes will examine each slot's busy and request bits and based upon this information as well as the information in the request, countdown, and bandwidth balancing counters know when to access a free slot on the bus. I considered the DQDB MAN to be operating over a large area (25 km bus length) and therefore I included the propagation delay between nodes within the study.

The token rings used in our study only allow a single token per ring to circulate at any one time. The token will be passed from one station to the next on the ring. Each station is allowed a single packet transmission with each token arrival. In the case of priority mode, the bridge will receive the token after the current station finishes sending its packet and the bridge will get to keep the token until its buffer is empty (exhaustive service).

The end stations have an application layer which generates fixed length data units with an inter-arrival time assumed to be exponentially distributed. The generation of data units is restricted by a threshold value M which is the maximum number of I-frames and data units being currently handled by LLC or queued to it (this process is called back pressure).

This value, M, encompasses: 1) the number of I-frames already transmitted but not yet acknowledged, 2) the number of I-frames waiting to be transmitted and 3) the number of accepted data units not yet transformed into I-frames.

The back pressure process is achieved using several mechanisms. The generation of data units is first restricted by the number of packets which can queue in the highest layer.The highest layer then sends the data unit packets to the LLC where the packets are formatted as I-frames. The sending of data unit packets by the highest layer is throttled by a resource which represents the window size employed by the LLC. Window units will be only released when the I-frames occupying them are acknowledged by the destination station. Furthermore, a copy of I-frames which have not yet been acknowledged is kept in the station.

The LLC server has to send and receive both I and S-frames (and if necessary to re-transmit I-frames), to slide the window when it receives the acknowledgement for I-frames, and to control the timers. The LLC queue can re-initialize or interrupt the timers at anytime. If a timer expires before being affected by the LLC, it will send a RR-frame to the MAC with the P-bit set to 1. For a more detailed description of the LLC protocol see chapter 2 of this thesis or the IEEE LLC standard[ANS85A].

4.3.1 Assumptions

The values for the model parameters have been chosen by considering the recommendations stated in different standards involved in this study[ANS85A, ANS85D, IEE90] and the values reported from tests done in our laboratory [GEO89, VAL89] and other similar studies[BUX85]. The DQDB bit rate is fixed by the number of slots in each implicit frame of 125 μs. Each slot insertion brings an additional bit rate of 3.392 Mbps. For this study, I used a DQDB frame length of 12 slots with a required bit rate of 41 Mbps (compatible with DS3 circuits). Half of these slots are reserved in order to support synchronous communications between users as PA slots. The remaining are QA slots available for transferring data packets between bridges connecting the token rings and DQDB MAN.

I assume a dual bus length of 25 km. The attached nodes are to be equally distributed across the network and their latency delay is equal to 600 ns (1 byte). The propagation delay of the medium is assumed to be 5.085 μs per km, since the use of optical fibres is assumed. Furthermore, I consider three token rings running at 16 Mbps. The delay for transferring the token between stations is assumed to be negligible.

I have 6 QA slots per frame (20.35 Mbps) available on each of the two buses for a total capacity of 40.7 Mbps for the QA traffic. The QA traffic will be provided by the three 16 Mbps token rings (total 48 Mbps). The DQDB network's utilization will be a function of the traffic arrival rate at each station on the token rings. Our study will examine the throughput and fairness characteristics of DQDB as it's utilization is varied from light to a very heavy load. For the end stations as well as at the bridges, I consider the use of COMPAQ DESKPRO 386/20 computers which are currently employed in our laboratory [GEO89].

The I-field and S-field frames sent on the token rings are assumed to have a fixed length of 1 kbyte and 25 bytes, respectively. The choice of 1 kbyte is justified by the fact that the COMPAQ computer is running on XENIX which allocates memory partitions in multiples of 1 Kbyte. As stated before, DQDB uses QA slots having a length of 53 bytes: 5 for identification and control, and 48 for containing the QA segment payload.The QA segment header itself uses 4 of the 48 bytes for header information which only leaves 44 bytes for user data. Therefore, at the bridges, a token ring I-frame (1025 bytes), after being converted to an INITIAL DQDB MAC PDU (1076 bytes), is segmented into 25 QA slots. The S-frame requires only one QA slot to be transferred.

The processing time for converting the token ring frames into DQDB MAC PDUs (or the reverse process) is fixed at 150 μs and 31 μs to format each QA slot of 53 bytes. These times were derived from tests where it has been calculated that a time of 10 μs is needed to fetch a specific memory location and 0.16 μs for reading each byte from the RAM. To reassemble the whole MAC PDU, a time of 20 μs is assumed for decapsulating each derived MAC PDU. This study also investigated the effect on system performance if bridges of 3 times this speed

were used.

I assumed the bridges contain pairs of buffers where each buffer is 8 Kbytes. One buffer is used for each direction of data flow in the bridge. I also investigated the effects on system performance of a sufficiently large buffer space such that packets and slots will never be lost at the bridges.

Execution of the LLC protocol at the end stations is assumed to require the following processing times: a) transmit I-frame (first-time): 250 μs; b) receive in-sequence I-frame and transmit RR-frame: 300 μs; c) receive RR-frame and delete I-frame(s): 100 μs; d) receive out-of-sequence I-frame and transmit REJ-frame: 125 μs; e) receive REJ- or RR-frame with F-bit and retransmit I-frame(s): 125 μs; f) receive out-of-sequence I-frame and transmit nothing: 75 μs; g) handle timer interrupt and transmit RR-frame: 125 μs; h) receive RR-frame with P-bit and transmit RR-frame with F-bit: 125 μs. The time out value is 250 ms. The back pressure threshold value M is set to the window size plus 4. This gives a source application the flexibility to prepare I-frames for later transmission and also during times when the LLC window is closed. The study compared system throughput for window sizes of 8 and 1.

4.3.2 Results and Analysis

Our study of the Private MAN was divided into three cases. In the first case I looked at five different scenarios for our 3 token rings interconnected using DQDB:

1. No priority to the bridges on to the token rings, LLC window = 8, Buffers = 8Kbytes, Bridge speed = as described in assumptions;
2. Priority to bridges on to token rings, LLC window = 8, Buffers = 8Kbytes, Bridge speed = as described in assumptions;
3. Bridge increased by three times the capacity, Priority to bridges on to token rings, LLC window = 8, Buffers = 8Kbytes;

4. LLC window reduced to 1, Priority to bridges on to token rings, Buffers = 8Kbytes, Bridge speed = as described in assumptions;

5. Unlimited buffer space, Priority to bridges on to token rings, LLC window = 8, Bridge speed = as described in assumptions.

For each scenario I was principally interested in obtaining the system throughput as a function of offered load and the mean end-to-end delays of the I frames. The mean end-to-end delay is defined as the elapsed time between the moment when the data packet is inserted in the LLC window at the source station (converted into an I-frame) and the moment when the packet is accepted by the destination station. To investigate the fairness aspects of DQDB, I tried to determine if any station or group of stations obtained a larger proportion of the DQDB bandwidth than was expected.

In the second case of the study I incorporated a modified version of DQDB's bandwidth balancing technique which I call Variable Bandwidth Balancing (V_BWB). The five scenarios of case 1 were repeated using this V_BWB to determine its effects on system performance and fairness.

In the third case of the study I reduced the amount of DQDB bandwidth allocated to the QA traffic from 50% to 33%. I then observed how this very heavily loaded system behaved with and without V_BWB. The reduction of the number of QA slots would guarantee constant competition for QA slots among the three bridges.

CASE 1 (5 scenarios)

The first scenario of case 1 involved the bridges having equal priority to the other 6 stations on each ring. The LLC window was set to 8, the bridge buffers size was 8 Kbytes. The system throughput increased linearly with offered load until it reached 14.75 Mbps(see figure 4.6), at which time the delay in the DQDB to token ring data flow through the bridge caused the buffers to overflow and lose packets. The lost packets triggered negative acknowledgements

and packet re-transmissions which further reduced the overall system throughput. The back pressure mechanism will eventually provide for a steady state system throughput of 10.25 Mbps (see figure 4.6) with a mean packet delay of 93 msec (see figure 4.7).

The next scenario gave the DQDB to token ring gateway priority on the token rings and provided for exhaustive service which would hopefully eliminate the bottleneck for the traffic leaving the DQDB MAN and entering the token rings. Figure 4.6 shows that the throughput does peak at a higher value than in scenario 1 (16.88 Mbps). Congestion in the token ring to DQDB side of the bridge then occurs and packets are lost. As in the first scenario, packet loss causes packet re-transmissions and NACKs to further reduce the system throughout. The steady state throughput is 14.4 Mbps with an average delay of 55 msec (see figure 4.7).

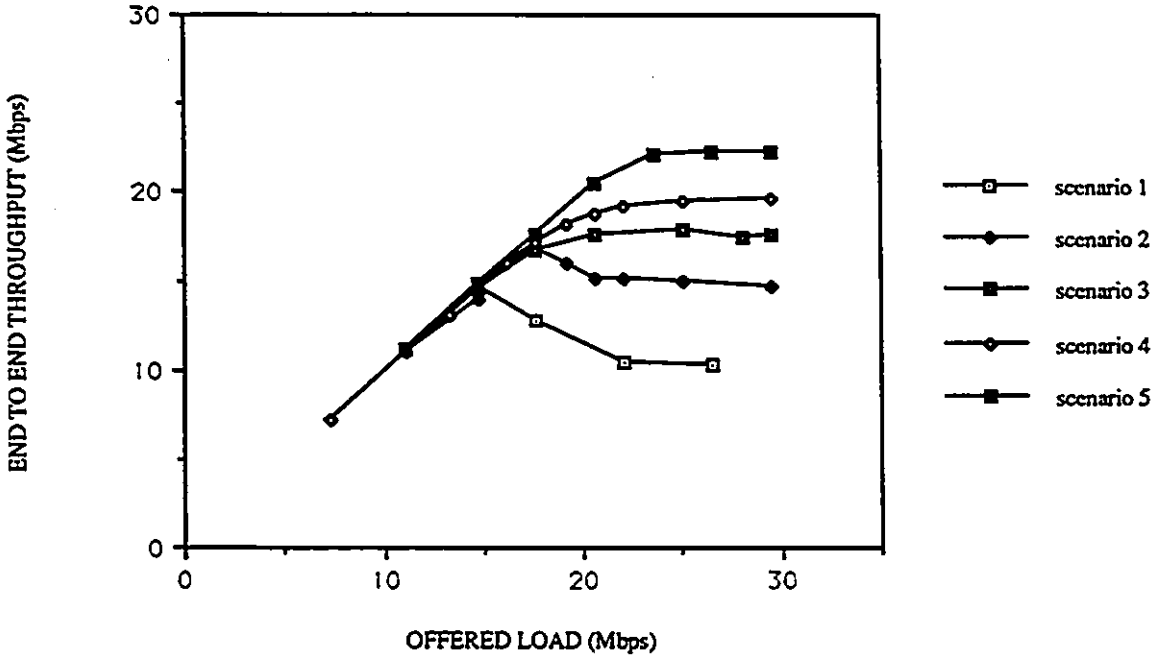


Figure 4.6: Case 1 Throughput Results.

The third scenario was to attempt to avoid the congestion in the token ring to DQDB side of the bridge by increasing the speed of the bridge three times its original capacity. Figure 4.6

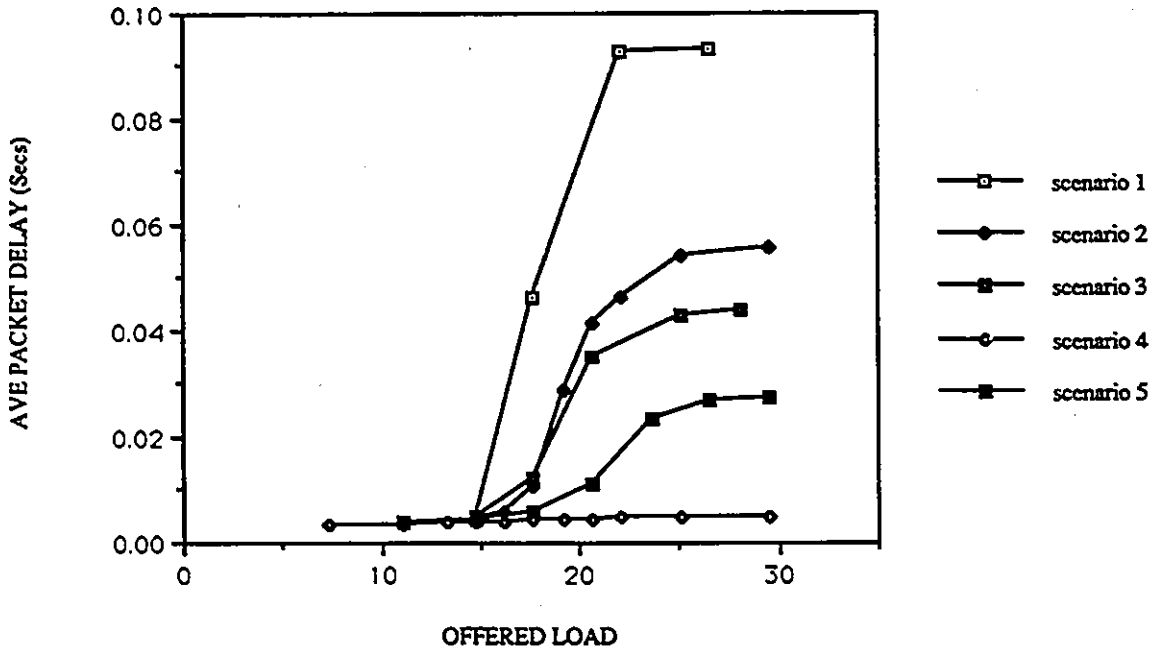


Figure 4.7: CASE 1 Average Packet Delay Results.

shows that the system performance does improve because of the faster bridge (steady state throughput 17.48 Mbps, and delay 44msec). The limiting factor in this scenario is again the buffer space, but here it is not a case of I-frames waiting to be segmented for DQDB as in scenario two, but rather a case where the segmented frames, are waiting for access to DQDB. The 8 Kbytes buffer used at the bridge is the total available space for frames waiting to be segmented as well as segmented slots waiting for access to DQDB.

The fourth scenario again was to avoid the congestion in the bridge, but this time I changed the window size of the LLC protocol from 8 to 1 (Stop and Wait protocol). The bridge priority, bridge speed and buffer size remained the same as in scenario two. The results (see figure 4.6) show that congestion is completely avoided and I am capable of obtaining a system throughput of 19.68 Mbps with an average packet delay of only 5msec (see figure 4.7). We do not have any congestion in this scenario and thus no down turn of performance due to re-transmissions and NACKs. The obvious problem with this scenario is that it is completely bounded in performance due to the nature of the stop and wait protocol[SCH87]. The hard limit in this scenario is the time it requires a packet to work itself through the network and the time it requires the ACK to be returned. The total of these two values completely bounds the system, this is not desirable. The results clearly indicated that reducing the window size of the LLC protocol is a very effective method to correct a congestion problem in the bridge.

During the final scenario of CASE 1 the buffer size restriction on the bridges was removed and this allowed for unlimited buffers on demand. The remainder of the parameters were as in scenario 2. As expected, the results showed that the highest system throughput occurs when we do have a windowing mechanism incorporated into the LLC with unrestricted buffers. The maximum throughput of our system was 22.22 Mbps (see figure 4.6) with an average delay of 27 msec (see figure 4.7).

This final scenario was similar to scenario 4, where we were dealing with a lossless system as no buffers will ever fill to capacity and lose packets. If packet loss was not the limiting factor for system throughput of this final scenario of case 1 my investigation then turned to

the bridge capacity. It was determined that it requires 925 μs to format and segment an I frame in the bridge, as well, it requires 181 μs to format the S-frame acknowledgement for that same I-frame. Thus the bridge requires 1.106 msec per successful packet which limits the system capacity to 22.22 Mbps. It should be noted the assumption of sufficient memory space for scenario 5 is not extreme. The simulation study shows that the steady state buffer requirements in the bridges were only 24Kbytes.

The fairness of DQDB has been a prime concern of the networking industry lately and therefore I investigated the actual number of packets each station managed to successfully transmit (successful means received proper acknowledgement). I wanted to see if any station or group of stations received more than its fair share of the DQDB bandwidth.

Our study for fairness focused on scenario 2 with an offered load of 130 packets/sec/station, which provides for packet loss and congestion problems. I wanted to investigate how heavily loaded token rings using the LLC protocol interacts with a heavily loaded DQDB MAN. Figure 4.8 indicates that the stations on the centre token ring have successfully transmitted a larger number of LLC packets (about 20%) during the simulation, than did the stations on the other two rings. This leads us to believe that they obtained a larger proportion of the DQDB bandwidth than did the others.

Investigation has determined that the unfair advantage enjoyed by the middle token ring is a function of the DQDB dual bus architecture. The situation arises from the fact that the middle ring enjoys the ability of sending half its traffic on each bus, whereas, the other two rings are compelled by the DQDB architecture to only use one bus (see figure 4.9). Token ring 1 for example must send all its traffic along bus A and similarly ring 3 must send all its traffic on bus B. One of the fundamental concepts of DQDB is that all access nodes may only have one DQDB slot in the distributed queue for each bus at any one time. This principle gives the obvious advantage to nodes who have traffic destined for both busses. These nodes may have two slots accessing the DQDB MAN at any one time, one on each bus. The nodes with uni-directional traffic will only have one slot accessing DQDB at any one time. This principle

will lead to unfairness for any station which will always send its traffic in one direction.

DQDB MANs are intended to be used to interconnect LANS of multiple communities of interest within a metropolitan area. Normally these communities of interest will keep their traffic very localized and thus the nodes on the outer edges of each community of interest will be sending their traffic in one direction only. If there are many communities of interest this unfairness could be a wide spread problem which must be dealt with. This situation of multiple communities of interest will be dealt with in the next chapter.

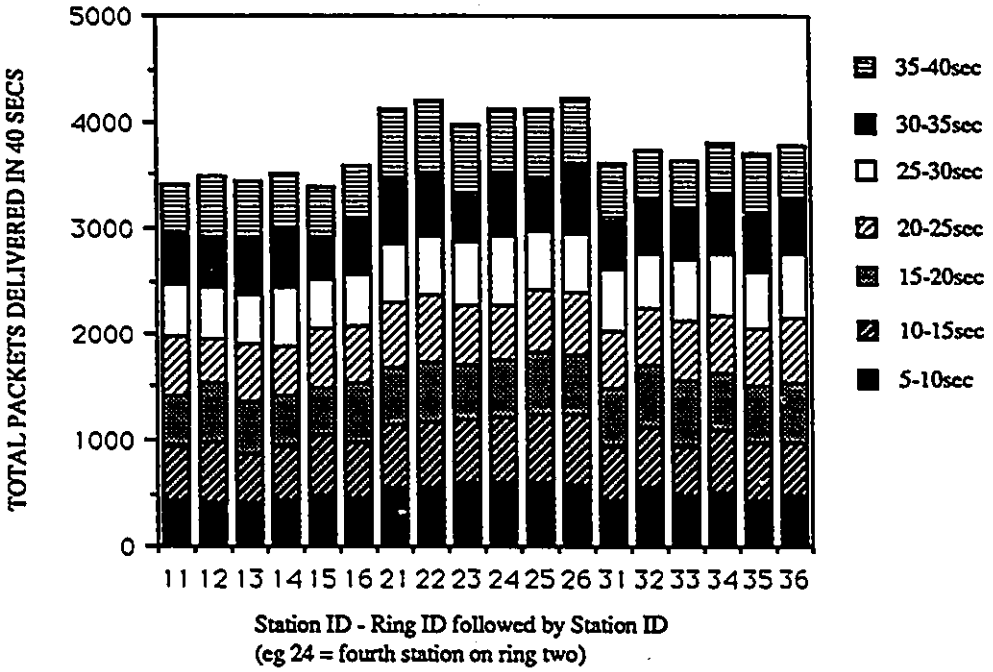


Figure 4.8: The Station To Station Packet Transfer Comparisons.

CASE 2:DQDB Standard With V_BWB (5 Phases)

The next logical step in this study was to look at possible ways to improve the unfairness of the DQDB MAN without having a major impact on the system throughput or delay.

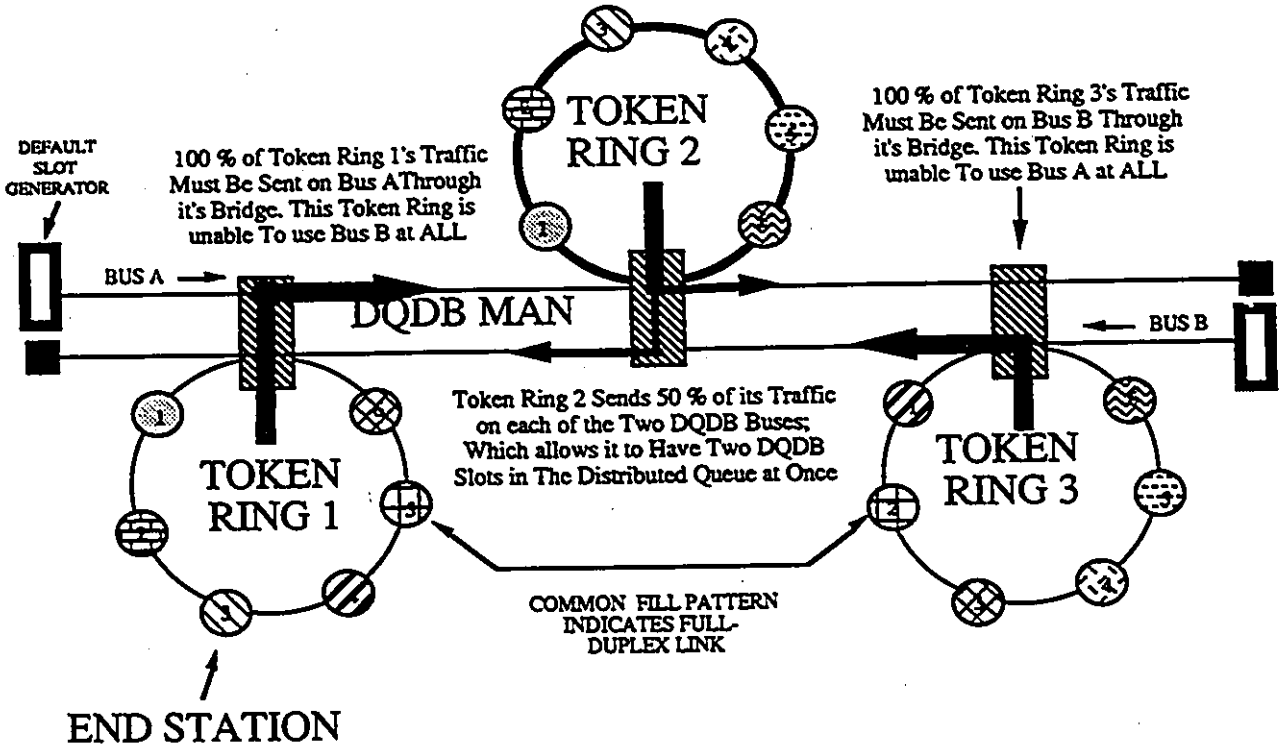


Figure 4.9: Unidirectional Traffic Unfairness

The proposed DQDB standard has incorporated a bandwidth balancing mechanism into the proposed standard. The major draw back to this mechanism is that all access nodes to DQDB are treated equally and all nodes are forced to let free slots go by unused at an equal rate. The proposed standard calls for the bandwidth balancing mechanism to be enabled upon start up and the default value for the BWB modulus is set to S[IEE90].

I am proposing that to improve the fairness of DQDB, the same mechanisms which the standard has already incorporated (BWB_CTR, BWB_MOD) be used in a modified manner. The modification would stipulate that the BWB modulus value associated with each access node be a variable (The standard now defines it as a system wide parameter). The value for each BWB_MOD would be a function of the destination address distribution of the nodal traffic. I refer to this as variable bandwidth balancing (V_BWB) as each station's actual BWB modulus would vary, and be based on the amount of traffic it sends on each bus.

The rational for this modification stems from asking the question, why should we take bandwidth away from an access node which is forced by the DQDB architecture to only use one bus and thus is already working from a disadvantage? The last section showed us that the dual bus architecture of DQDB forced the two token ring bridges on the outer edges of the private MAN to only utilize a single bus for traffic (see figure 4.9). If we enable BWB for these outer stations we will force them to give up needed slots to the middle token ring which will cause a higher degree of unfairness.

The obvious challenge here is to determine the V_BWB modulus for each access node. For this study I decided that the end stations were already working from a disadvantage and thus I should not force them to give up free slots. The standard calls for the BWB mechanism to be either enabled or disabled system wide. As I did not want the end stations to be forced to give up free slots, I set the V_BWB modulus to 100,000 for the end access nodes. This effectively turns off BWB for these two nodes as they only have to give up one free slot for every 100,000 it uses.

The middle access node had its V_BWB modulus varied from 0 to 9. A V_BWB_MOD

set to 0 indicates that the BWB mechanism is disabled. Figure 4.10 displays the average system throughput and delay results for all data packets sent from all stations as a function of the V_BWB moduli chosen for the centre access node. When the centre access node had its V_BWB modulus set to 1, it was forced to let every second slot which it would have used pass by unused. This had a detrimental effect on both throughput (13.76 Mbps) and packet delay (43msec) when comparing to the same system operating with no BWB at all (15.94 Mbps and 29 msec). The middle Token Ring was not permitted to enter it's I frames (segmented into slots) on to the DQDB MAN as fast as before, nor could the acknowledgement packets be sent from the centre ring as quickly to inform the outer rings that they can send their next packet. It is clear that by reducing the V_BWB modulus of station 2 to a value of 1 I am forcing this station to give away an excessive number of free slots which leads to a drastic degradation of system performance.

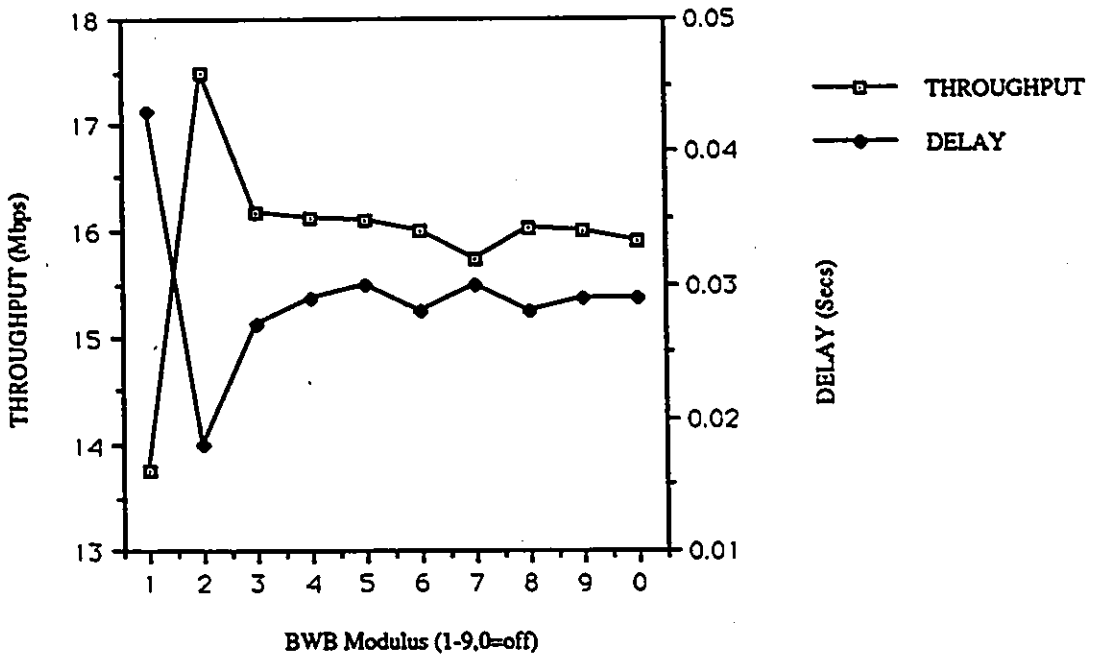


Figure 4.10: V_BWB Throughput and Delay Results for Moduli 1-10.

When the V_BWB modulus was increased to 2 we see a dramatically different situation(see figure 4.10). Here the average system throughput increases to 17.5 Mbps, while the average packet delay reduces dramatically to 18 msec. As the V_BWB modulus is increased beyond 2 for the middle access node we gradually approach the throughput and delay characteristics when BWB is not enabled. This convergence towards the no BWB case as the V_BWB modulus increases is expected as the stations are not being forced to give up as many free slots and the system will operate as if the BWB mechanism is disabled.

The drastic increase in overall throughput is a very surprising result. It must be pointed out that what we are doing is forcing bridge 2 not to use some of its DQDB bandwidth. This unused bandwidth will not be used further down the DQDB bus because the ring located down stream is trying to place all its traffic onto the opposite bus (see figure 4.9). Then the question arises, how could taking away from bridge 2 possibly provide a higher system throughput even though these extra slots are never used later?

The answer stems from the DQDB protocol itself. The middle bridge will only issue the next slot request whenever it has successfully transmitted the current slot on to the DQDB bus. The middle bridge is forced by the V_BWB modulus to let free slots go by and thus will not be issuing requests as often. The end station will be able to obtain a higher proportion of the DQDB bandwidth because it will not be responding to as many requests from the middle bridge. The improved system performance does not stem from the slots taken from the middle bridge but rather the reduced number of requests generated by the middle bridge.

The five scenarios of case 1 were repeated but this time I included the V_BWB mechanism. The end access nodes were given a V_BWB modulus of 100000 (effectively turning off BWB) and the middle access node was given the V_BWB modulus of 2. The value 2 was chosen because it produced the best results of all V_BWB moduli tested (see figure 4.10). Figures 4.11 and 4.12 show that in all cases the system throughput and packet delay were equal to (scenarios 4,5) or better (scenarios 1,2,3) than the case without BWB. Again this is a very surprising result because we are actually taking DQDB slots away from the middle bridge

to improve fairness, yet the overall system throughput actually increases and average packet delays decrease.

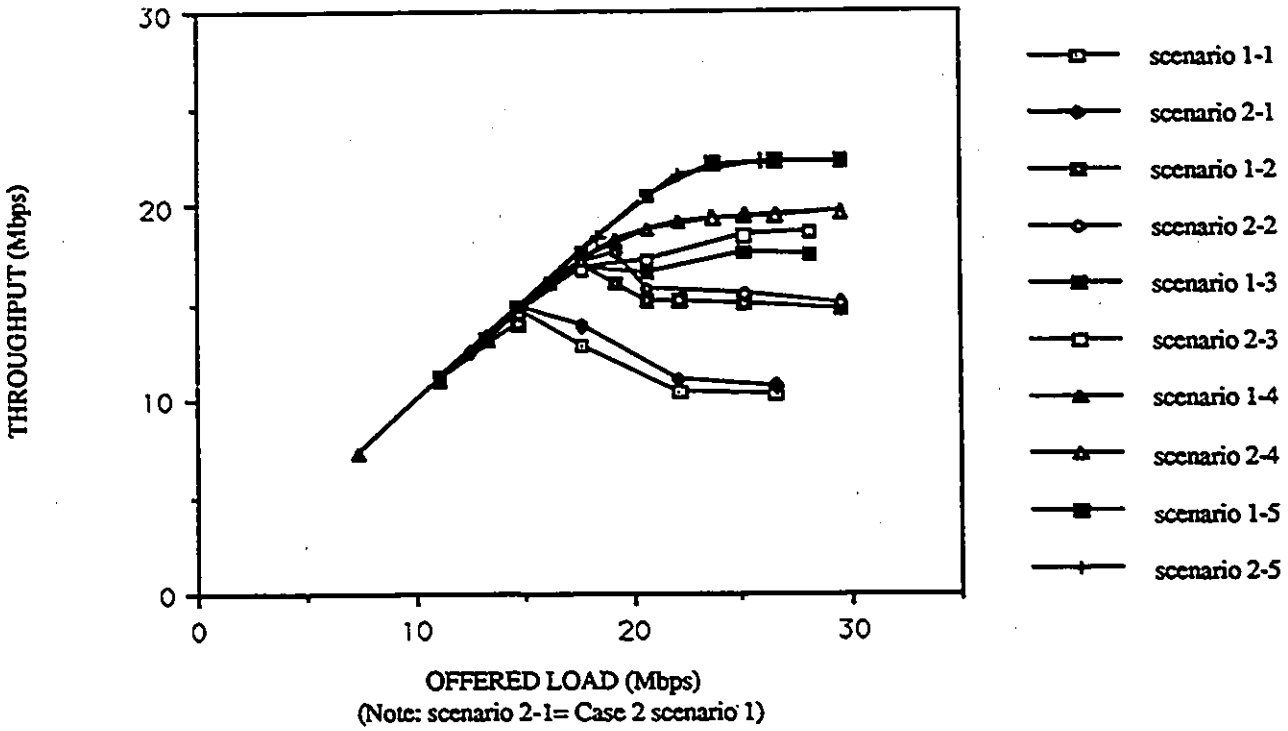


Figure 4.11: CASE 1 VS CASE2 Throughput Comparisons.

We can see that the station-to-station throughput no longer favours a single access node (see figure 4.13). The original goal of the V_BWB technique was to avoid unfairness within the DQDB MAN, this goal has been achieved. A second interesting and unexpected result which surfaced in the station-to-station comparisons is that by forcing slots away from the middle ring, the middle ring's throughput actually increases! We had seen earlier that the system throughput increased because of the V_BWB, but it was assumed that the system increase was a result of more packets being sent by the outer loops. The station-to-station comparisons shows that for all stations except 1 and 6 on ring 2 their throughput has actually increased (see figure 4.13). The increased throughput of stations on the middle ring is a direct result of the

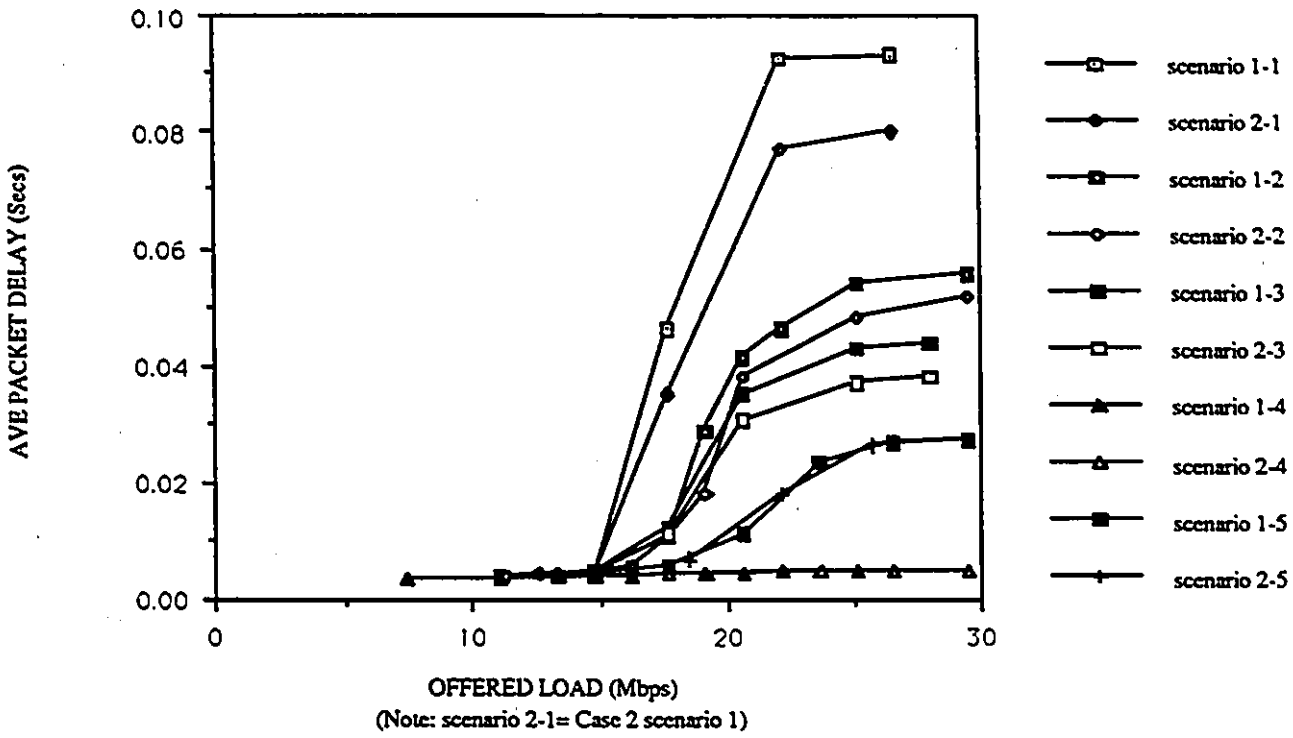


Figure 4.12: CASE 1 VS CASE 2 Delay Comparisons.

outer rings having more bandwidth and being able to return the positive acknowledgements faster to the middle ring. Once the middle ring received this acknowledgement it was then able to send its next packet.

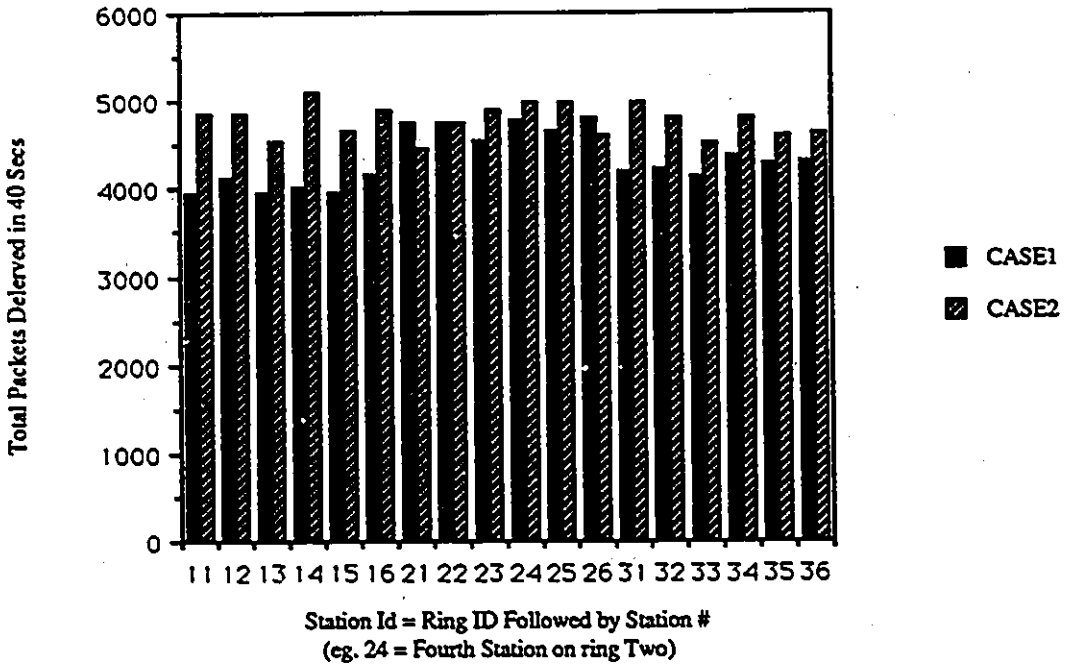


Figure 4.13: CASE 1 VS CASE 2 station-to-station Throughput Comparisons

This result is indeed a surprising one. This points out the fact that even though ring two was able to obtain a disproportionably large portion of DQDB's bandwidth its successful throughput was actually decreased. The reduction in successful throughput was caused by bridge two not allowing enough bandwidth to the other two bridges to properly return the required ACKs. In effect Bridge two was a victim of its own success and proves that more is not always better.

CASE 3: V_BWB Operating on a Very Heavily Loaded DQDB MAN

Case 2 showed us that the V_BWB technique could be used to reduce packet delays and to improve both system throughput and fairness with the first three scenarios of CASE 1. The fourth and fifth scenarios were unchanged by V_BWB. The lack of a change in the fourth scenario is understandable because it is the stop and wait protocol. The stop and wait protocol has a maximum throughput capacity based on the return distance between source and destination. The limiting factor with scenario 5 (unlimited buffer,LLC window = 8,V_BWB) was the bridge speed, which is not a function of either the LLC or the DQDB protocol. It would be interesting to see what effects, if any, V_BWB could have on scenario 5 if the gateway speed was no longer the limiting factor.

The third and final portion of this chapter's study was to re-examine scenario 5 of Case two (unlimited buffer, window 8,V_BWB) to see if the V_BWB technique could be used to improve its throughput, delay or fairness. It was quickly determined that improving throughput was not really possible because the bridge capacity (22.2 Mbps), now being the limiting factor, would be quickly replaced by the token rings as the limiting factor. Each token ring operates at 16 Mbps, thus the 3 ring system has a total capacity of 48 Mbps. 24Mbps are used to place traffic on to DQDB and 24 MBPS to take data off DQDB. The 24 Mbps placing data on to DQDB includes the packet overhead (4.88%) which leaves us with a new maximum throughput of 22.8 Mbps. This new maximum is so close to the old limit of 22.2 Mbps, that I decided to focus on possible fairness improvement.

The fairness issue would come to the forefront, if the DQDB bandwidth allocated for the QA traffic was reduced to a point where the three rings were constantly competing for slots. To accomplish this I reduced the number of QA slots from 6 to 4 slots per frame of 12. This reduction will ensure constant competition between bridges for access to the DQDB MAN QA slots.

Figure 4.14 displays the station-to-station throughput comparisons with no BWB for

scenario 5 (unlimited buffer,LLC window = 8) and with only 4 QA slots per frame. It is clear that the stations on ring 2 have the clear advantage, as do the stations who communicate with ring 2. The 3 full duplex communication links not involved with ring 2 obtain much less of the DQDB capacity than do the other 3 links. Station 5 on ring 2 transmitted 75% more traffic than did station 3 on ring 3. These stations were chosen for comparison purposes because they are the best and worst cases. It is clear from figure 4.14 that the DQDB MAN is displaying a high degree of unfairness towards the outside bridges.

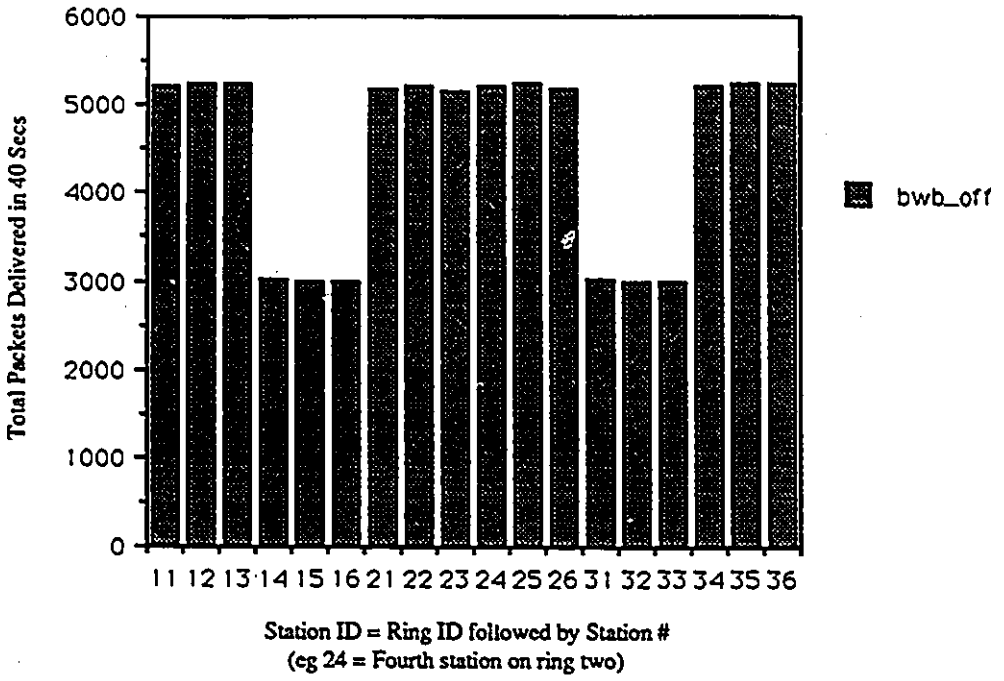


Figure 4.14: CASE 3 station-to-station Throughput Comparisons without V_BWB.

After the V_BWB techniques were introduced, it was observed that the V_BWB modulus of 1 is the only value which effectively divides the DQDB bandwidth. The difference between the most and least advantaged stations with V_BWB is now only 1.5% (see figure 4.15) which is down from 75% without V_BWB. The cost of using the V_BWB mechanism must also be investigated.

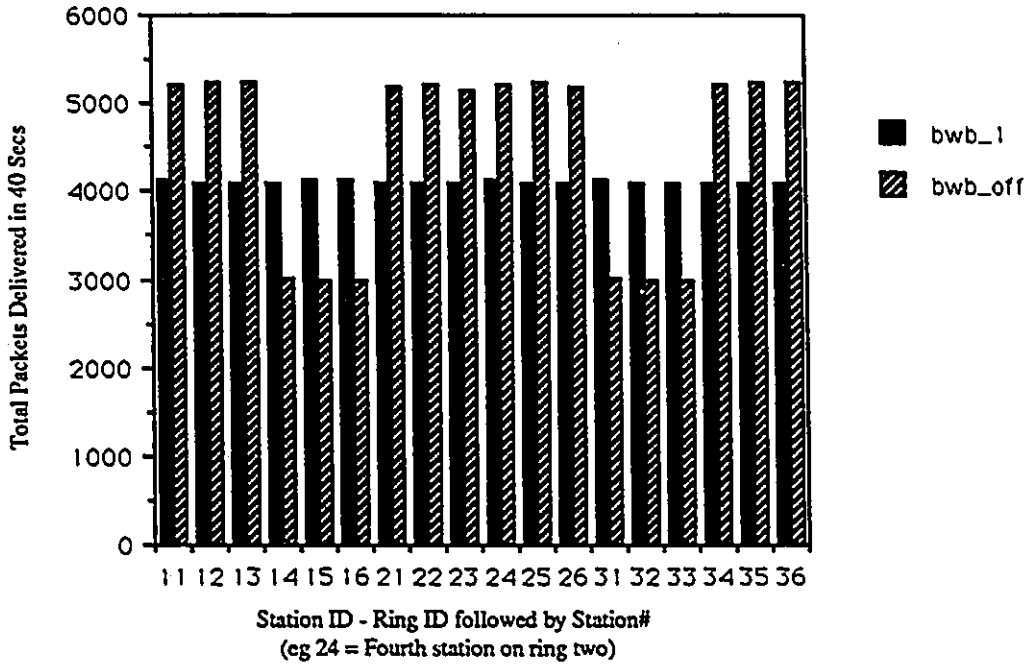


Figure 4.15: CASE 3 station-to-station Throughput Comparisons With and Without V_BWB.

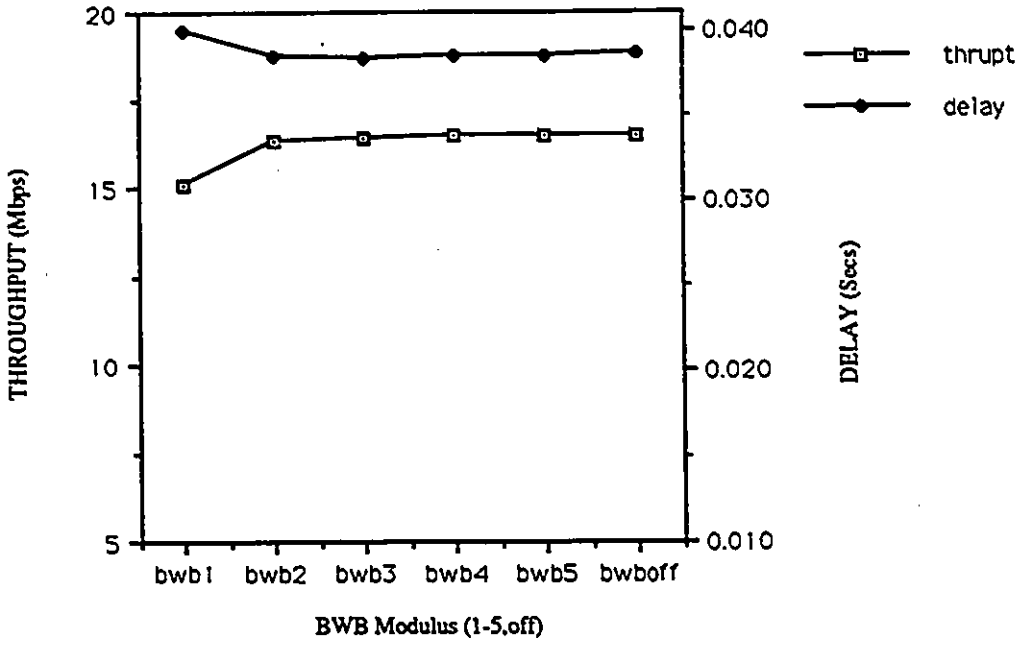


Figure 4.16: V_BWB Throughput and Delay Overhead Costs.

As was done in case 2, I varied the V_BWB modulus of the middle ring and effectively turned off BWB for the bridges on the outer edges of the private community of interest. The middle bridge had its V_BWB modulus varied from 1 to 5 and I also ran the simulation without the BWB mechanism activated for comparison purposes. Figure 4.16 shows that the overall system throughput decreases by 8.1% and the average packet delay will rise by only 2% when the V_BWB modulus is set to 1 when compared to the no BWB case. These figures indicate that a very unfair system which allows one station to send 75% more traffic than the others could be made quite fair using the V_BWB technique. This fairness can be achieved with minimal overhead costs (throughput loss = 8.1% and ave packet delay increase = 2%).

4.4 Conclusion

In this chapter, I have investigated the fairness and performance aspects of an interwork system using the IEEE 802.6 DQDB MAN standard as a backbone network interconnecting three IEEE 802.5 Token Rings. Such a system introduces the problems of interconnecting networks which do not use the same access protocol and maximum packet length. Bridges must be used to provide protocol translation between both types of networks and to perform the segmentation and reassembly of packets.

I have considered two modes of priority dealing with the token rings: first all stations on the LANS including the bridge have equal priority, and second higher ring-access priority is given to the DQDB-LAN bridge with exhaustive service.

The application studied was the transfer of packets between homogeneous stations situated on different token rings. It was shown that as long as there is sufficient buffer space the larger window sizes of LLC perform better. If congestion does appear it would be desirable to reduce the window size until congestion is clear. The use of priority mode for the bridges provides better performance, mainly because congestion problems are considerably reduced in the bridge.

The simulation results have shown that the proposed DQDB standard acting as a private MAN has problems to treat all LANs fairly within a community of interest. It also appears that congestion problems in the bridges can be the source of important delays and packet losses if buffer overflow occurs. The use of higher ring access priority and exhaustive service for the bridges has been proposed for reducing congestion. The most interesting observation discovered during the study was the unfairness of the DQDB protocol. Some stations were able to transmit up to 75% more data than other stations. The fairness problem was overcome by implementing a modified version of the proposed bandwidth balancing mechanism called V_BWB.

The observations made during this chapter raise some very serious questions about the fairness of the DQDB protocol. Of particular interest is the unfair treatment of stations on the DQDB MAN who only utilize one of the two DQDB buses. This chapter dealt with a single community of interest which is effectively a private MAN. The fairness problem was correctable using the V_BWB technique introduced to help with the unfair DQDB bandwidth allocations.

The obvious expansion to this study is to increase the number of communities of interest using the DQDB MAN. This would change the nature of the MAN from Private to Public. It would be interesting to measure the degree of unfairness which is observed within a Public DQDB MAN supporting multiple separate communities of interest. The DQDB public MANs offered by the carriers should treat all paying customers equally but the results of this chapter suggest that this will not be the case. It would also be interesting to determine if the lessons learned from this chapter dealing with V_BWB could easily be mapped into a larger and more general situation with many communities of interest. These questions will be addressed in the next chapter of this thesis.

Chapter 5

Public DQDB MAN Supporting Multiple Communities of Interest

The last chapter provided us with a detailed description of the performance and fairness issues involved when DQDB is used to support a private MAN consisting of three IEEE 802.5 Token Rings. The performance and fairness aspects of this interwork system were investigated. The interaction between the LLC protocol and the DQDB protocol was a focus of the previous chapter and it was observed that by changing certain parameters of either the LLC or DQDB protocols we could observe drastic changes in the performance characterises of the overall system.

The two protocols are so tightly interconnected that it was difficult at times to separate which performance and fairness changes were being caused by which protocol. For example in the last section of chapter 4 it was observed that the throughput of the middle token ring actually improved by reducing its allocation of DQDB bandwidth. This improvement occurred because the LLC acknowledgement packets were being returned to the middle ring faster than before. This allowed the middle ring stations to send their next packets quicker to improve its overall throughput.

The unfairness of the DQDB dual bus architecture was highlighted in the last chapter and this unfairness was overcome using the V_BWB technique. The V_BWB technique is a modification to the BWB mechanism which is included within the DQDB standard. A

limiting factor to the study of chapter 4 was the fact that we only had a single community of interest using the DQDB MAN.

This chapter will take a closer look at the DQDB MAN in supporting multiple communities of interest. This DQDB MAN which is used to support multiple communities of interest is referred to as a Public MAN. The DQDB Public MAN will be a service offered by the carriers. It will be expected that all stations using the Public service will receive equal treatment as they will be paying an equal fee for the service.

The unfairness of stations with unidirectional traffic will be investigated. This unfairness was pointed out in the previous chapter for a single community of interest. This chapter will expand on the investigation of the last chapter to include three separate communities of interest each with three LANs. Of the twelve access nodes required to support the three communities of interest six will be unidirectional. This will provide us with a clear picture of the unfairness of the DQDB MAN to support these small communities of interest type customers. Again V_BWB will be used to overcome the unfairness encountered.

5.1 Introduction

This chapter will again investigate the unfairness of the DQDB protocol in equally dividing its available asynchronous bandwidth between active stations. The focus of this chapter will differ from the last in that I will deal with 3 communities of interest (e.g. university campus, a municipal government and a major accounting firm) all commonly sharing a single DQDB Public MAN. I will not consider the LLC protocol in this chapter as I am more interested in the ability of DQDB to treat all active access nodes equally. The last chapter showed that separating the performance aspects of the LLC and DQDB protocols can be very difficult and even misleading at times.

Each community of interest consists of 3 token rings and all traffic generated within a community of interest will have a destination address within the originators own community.

This chapter will only deal with a single level of priority. The DQDB standard allows 3 levels of priority, but since we are investigating equal treatment between communities of interest we assume all are working from the same priority level.

The results of this chapter will help managers of DQDB based Public MANs, which are used to supporting many communities of interest, to more equally distribute the available bandwidth between the stations.

5.2 Application Overview

The DQDB standard has been studied a great deal for its efficiency and fairness. Most of these studies assume a uniform traffic distribution, that is all stations send their traffic to all other stations on the network with an equal probability (I took this approach for my performance study during chapter 3). One of the principal functions of the DQDB MAN will be to provide the communication resources required for many different customers within a metropolitan area in the form of a Public MAN.

A typical costumer will be a medium sized organization which has neither the requirement for, nor the resources to afford a private MAN. This type of customer will acquire the communication bandwidth requirements from the local carrier's MAN and pay for these services knowing they are sharing the MAN with several other customers on the network. A MAN of this nature will not have the uniform traffic conditions that have been extensively studied for DQDB's behaviour. The MAN users will not attempt to transmit to all other stations on the MAN, in fact they probably do not know who else is on the network. As all customers will pay an equal fee for service, they expect equal service for all users of the MAN. This chapter will investigate the performance and fairness of a DQDB based Public MAN to service the requirements of 3 independent communities of interest all sharing a common DQDB Public MAN.

The 3 communities of interest under study can be envisioned as in figure 5.1, where each

community of interest consists of 3 LANS which must be interconnected. The University campus, Municipal Government and Accounting firm are all independent organizations and will only send traffic within its own community of interest. The 3 customer system can be pictured as a 9 access node system, 3 for each customer. Figure 5.2 shows how an abstract access node architecture would look if the connecting MAN was a typical shared media such as an eithernet or token bus. It is easy to picture how this type of system would be fair for all access nodes attempting to access the MAN.

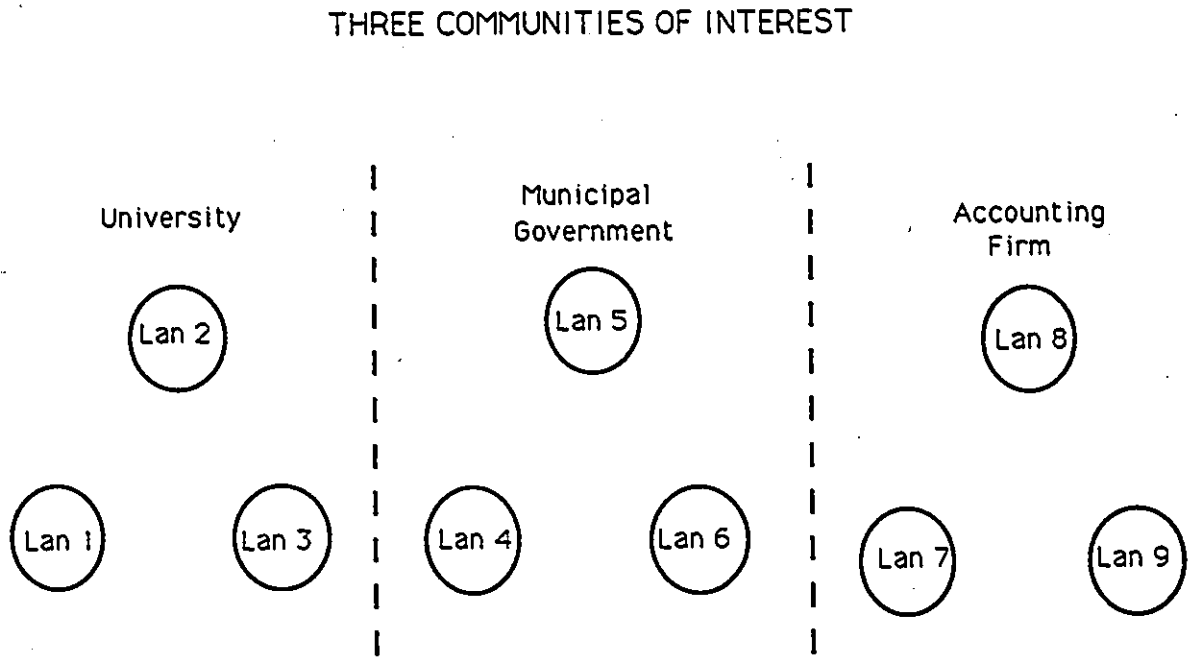


Figure 5.1: Three Communities of Interest Each with Three Token Ring LANs to be interconnected.

The DQDB based architecture given in figure 5.3 shows how our 3 communities of interest now require 12 access nodes because of the dual bus architecture. It is not quite as simple to picture the fairness characteristics of this system and thus they must be investigated. Figure 5.3 shows that with the dual bus architecture and with the 3 divided communities of interest we certainly do not have a uniform traffic distribution for our DQDB system. For example, the accounting firm's LAN on the far left of its community of interest (Lan 7 in figure 5.3) uses

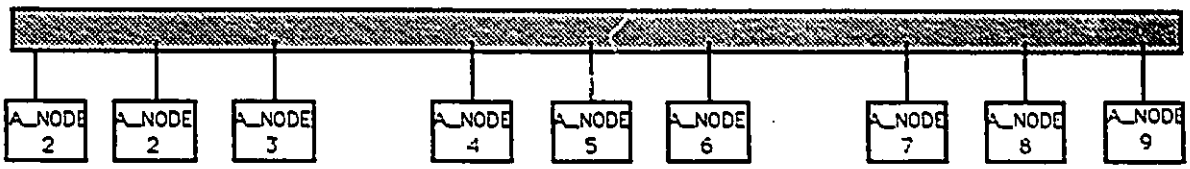


Figure 5.2: Abstract MAN Architecture used to interconnect the LANs of Three Separate Communities On a Single Shared Medium.

the second from last access node on BUS A, yet it sends 100% of its traffic further downstream on bus A. This type of traffic distribution is far from uniform but will be quite common for DQDB based Public MANs supporting many communities of interest.

As pointed out in earlier chapters 2 and 3 the DQDB MAN has displayed a great deal of unfairness when dividing its asynchronous bandwidth between active stations during high network utilization. The 802.6 standardization committee approved a major modification to the DQDB standard which introduced a bandwidth balancing mechanism which is solely intended to equally distribute the DQDB bandwidth[HAH90].

This mechanism includes a bandwidth balancing counter (BWB_CT) and a bandwidth balancing modulus (BWB_MOD). Each station has two BWB_CT's, one for each bus. These counters will keep track of the number of slots sent on each bus by incrementing its value for each slot sent. The BWB_MOD is a system parameter which states the maximum number of slots a station may send on each bus before the station is forced to let a slot pass by unused. While the BWB_CT is incremented with each slot transmission, its value is compared to the BWB_MOD when the two are equal the BWB_CT is reset to zero and the next slot which the station would normally take for itself is let to pass unused. The idea being that other stations will use the slots which are let to pass unused and in turn generate more requests to more evenly distribute the DQDB bandwidth.

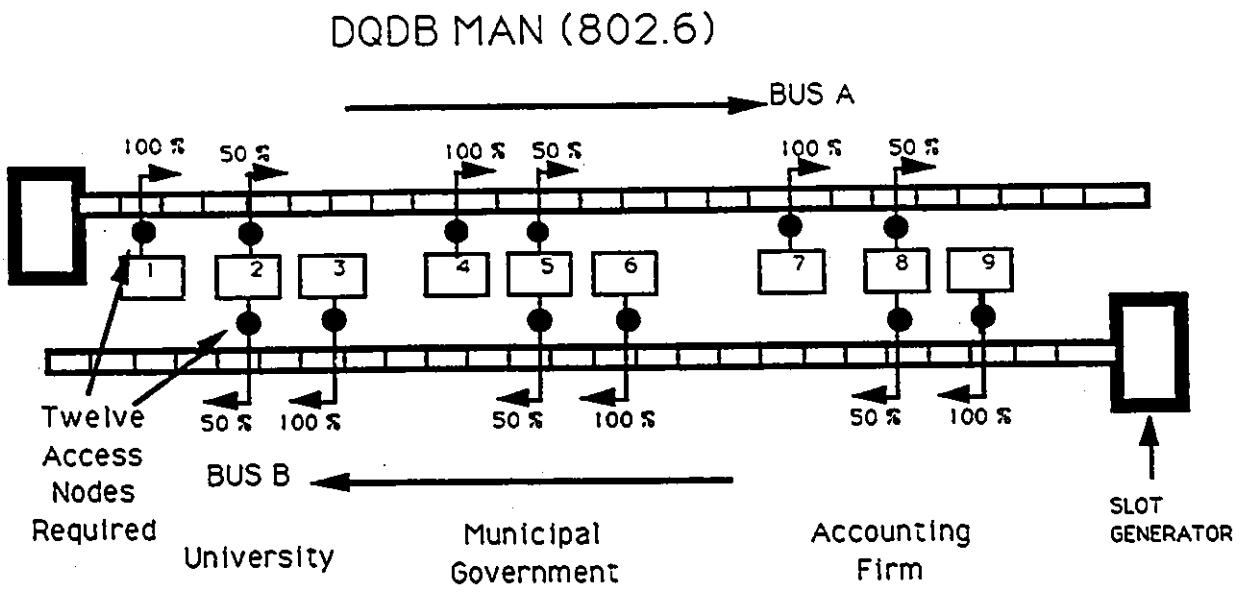


Figure 5.3: Three Communities of Interest interconnected Via a DQDB Public MAN.

The BWB_MOD has been made a system wide parameter within the DQDB standard [IEE90]. Its value may range from 0-64. If the value is 0 this indicates that the bandwidth balancing mechanism is disabled, otherwise the value is chosen between 1 and 64. This value once chosen will be the value used by all stations on both buses. The standard calls for the default value of the BWB_MOD = 8.

It will be shown in this chapter that by assigning all stations on both buses the same modulus without considering the station's traffic arrival rate or the station's traffic destination distribution may cause more damage to the fairness of the DQDB protocol instead of aiding fairness. It will also be shown that the V_BWB mechanism introduced in the last chapter which is a modified version of DQDB's bandwidth balancing can be used to help rectify the unfairness problems of DQDB. The BWB_MOD must not be a system wide parameter treating all stations on both buses equally. The stations bus position and traffic distribution should be considered when choosing the bandwidth balancing modulus for both access nodes of each station.

5.3 Performance Evaluation Overview

The performance study was conducted using the discrete-event simulator of the QNAP2 simulation package[QNAS4]. The DQDB portion of the study is based on the latest version of the IEEE DQDB MAN standard[IEE90] which was described in detail in chapter 3. The DQDB MAN simulation comprised of two slot generators located at the head end of each bus and produced slots according to the 125 μs timing structure of DQDB. After the slot was generated it would be successively passed down the bus from node to node. All six access nodes along each bus will examine each slot's BUSY and REQUEST bits. Based upon the values of these control bits and also the current values of the REQUEST, COUNTDOWN and BANDWIDTH BALANCING counters each access node will know when to access a free slot on the DQDB bus.

The end stations have an application layer which will generate slots destined to other stations within the same community of interest. The slot inter-arrival time is exponentially distributed. The mean arrival rate for each station will always be $\frac{1}{9}$th of the total load currently on the DQDB MAN. This will ensure that all stations are trying to place equal amounts of traffic on to the DQDB MAN under all conditions. For example, if the DQDB load is light, each station will be offering $\frac{1}{9}$th of this light load. All stations will vary their loads equally as the DQDB MAN is tested under various load conditions.

5.3.1 Assumptions

The values for the model parameters have been chosen by considering the recommendations stated in the DQDB draft standard version 14 [IEE90]. The DQDB bit rate is fixed by the number of slots contained in the DQDB 125 μs implicit frame. For this study I have chosen a frame length of 12 slots which requires a bit rate of 41 Mbps (compatible with DS3 circuits). A background load of PA (Isochronous) type traffic is assumed to require 1 of the 12 slots within the DQDB frame. The remaining 11 slots are available to the 12 access nodes of the

three communities of interest to send their QA (Asynchronous) type traffic slots.

I assume a dual bus length of 25 kms. The access nodes are equally separated over the 25 kms. The station latency delay is 600 ns(1 Byte) and the propagation delay of the link is 5.085 μs per km, as the use of optical fibre is assumed.

I will be interested in determining the fairness behaviour of the DQDB MAN while supporting these multiple communities of interest. The load on the DQDB MAN will be varied from as low as 50% up to 99% of the DQDB capacity. It is assumed that each station will offer $\frac{1}{9}$th of the DQDB offered load and thus should receive $\frac{1}{9}$th of the bandwidth. All stations should experience roughly the same delays gaining access on to the DQDB MAN.

5.4 Results and Analysis

My study was divided into three phases. During the first phase I investigated the fairness of the current version of the draft DQDB standard to support these three communities of interest. Phase two attempts to determine the reasons for the unfairness which were observed during the first phase. Based on these reasons phase two attempts to develope a strategy which might correct the problems. The final phase compares the performance of the DQDB MAN (operating with the strategy developed in phase 2) to the results using the current DQDB standard. All of these results were taken into consideration when designing an approach for optimal system performance which will be put forward in the final phase of this chapter.

5.4.1 Phase One: Current DQDB Standard Performance and Fairness

The goal of this first phase was to determine if the current version of the standard could equally divide its resources between the 3 communities of interest. The current version of the standard calls for the bandwidth balancing mechanism to be enabled with the system wide parameter BWB_MOD set to 8. Figure 5.4 displays the average response delay of all 9

stations to get a slot onto the DQDB MAN. The graph clearly shows that at low to medium loads the MAN gives a fairly even response delay to all stations. During 75% MAN utilization the average slot delay for all stations is 5.07 slots. The most advantaged station 1 has an average access delay of 4.07 slots and the most disadvantaged station 3 has an average delay of 6.47 slots. We can see that the unfairness observed at 75% load is not extreme, as all stations are within 1.4 slot times of the network average.

As the load increases beyond 75% we start to see a trend of DQDB favouring stations 2,5 and 8. As the load on DQDB becomes very heavy (99%) we see that this unfairness becomes extreme. It must be pointed out that figure 5.4 is using a logarithmic scale and thus the unfairness is in the order of several magnitudes. At 99% load the average access delay of the most advantaged station 5 is only 12.33 slot times compared to the least advantaged station 7 with an average delay of 26,540 slot times. If we assume that the stations are using a higher layer protocol such as LLC to support the end-to-end communications, the disadvantaged stations would effectively turn off. The timeout and NACK packets which would be generated due to the extremely unfair behaviour of DQDB would reduce the amount of successful data transfer to almost zero.

The stations which enjoy an unfair advantage given to them by DQDB might also not send much traffic as acknowledgement packets are required by the LLC protocol. If the disadvantaged stations are the ones required to send these acknowledgements, virtually no data packets would be transferred by this system. The advantaged stations would be continually retransmiting the same unacknowledged packets. This type of phenomenon was observed in the last chapter of this thesis.

5.4.2 Phase Two: Unfairness Investigation

Recently there have been a great number of studies on the behaviour of the DQDB protocol[BIS90, BUD86, CON89, CON91, DAV89, DES90, FDI90, FIL90, HAH90, KAU90, KIM90] [NEW86, NEW88A, ROD90, RUB90, VAL89, VAN90A, VAN90B, WAI89, WON89, WON90, ZUK87,

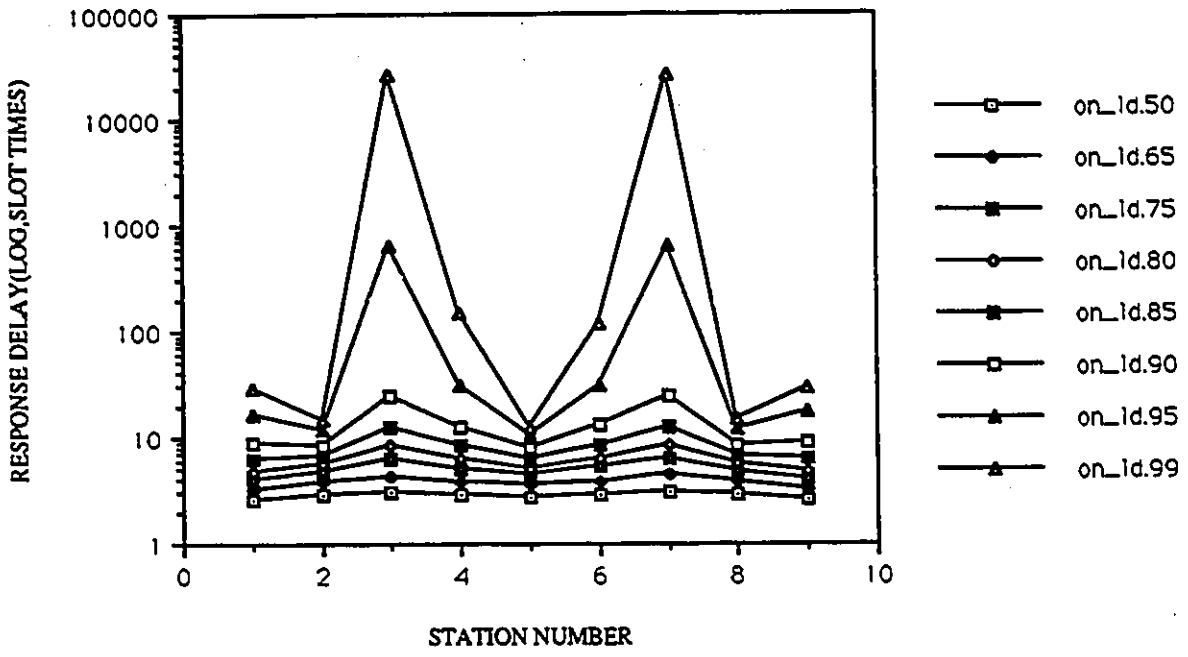


Figure 5.4: Station Mean Access Delays With The Default BWB mechanism enabled at all load levels.

ZUKSSA, ZUKSSB, ZUKSSC]. In the early days of the standard it was first claimed that this protocol would behave fairly and efficiently up to 100% load[NEWS86]. It soon became clear that this protocol could behave quite unfairly depending on offered load, station separation distances and initial traffic conditions just before overload occurs[WONS89]. The "standby state" has been removed from the standard and the BWB mechanism has been added to correct the unfairness in the standard. Several recent studies have shown that the new standard can indeed be made fair when using the BWB mechanism added to the standard[CON91, HAH90].

My results obtained in phase one (see figure 5.4) clearly conflict with these resent studies. I have incorporated the BWB mechanism into my study, yet I still observe drastic unfairness. Investigation has determined that there are two major types of unfairness within the DQDB MAN. The first is positional unfairness. This has been extensively discussed in recent papers which have concluded that stations at the head end of the DQDB MAN have the advantage when accessing slots. The second type of unfairness is traffic direction unfairness. This unfairness is a direct result of the DQDB MAN architecture using the two unidirectional buses. This second type of unfairness stems from the fact that many stations do not enjoy the privilege of using both DQDB buses to transmit their traffic. This second type of unfairness has not been given the attention it requires by the current literature because most studies assume a uniform traffic distribution and that the problem therefore is not extreme.

Figure 5.4 shows that under heavy load conditions (99%), both stations 3 and 7 are treated extremely unfairly when trying to send traffic on the DQDB based MAN. Station 7 suffers from both positional unfairness and traffic direction unfairness. The positional unfairness is clear from the fact that station 7 is the second from the end on bus A and thus is acting from a disadvantage compared to the stations positioned closer to the head end of the bus. The traffic direction unfairness arises from the fact that station 7 is on the outer edge of the accounting firms community of interest and is forced to send all its traffic on bus A back towards the other two stations within its community of interest.

A fundamental concept of the DQDB standard is that each active station may only have

two slots, at each priority level, in DQDB's distributed queue at one time. Each station is allowed one slot in each of the two distributed queues used to control access to the two DQDB buses. Station 7 is only using bus A to send its traffic and thus it will only have one slot in the distributed queue of DQDB. This effectively cuts the available bandwidth to these unidirectional stations in half when you compare them with stations which are able to utilize both buses of DQDB. The same results for station 3 are observed because it is in a similar situation to station 7 only on the opposite bus, B.

Station 8 is the furthest active access node from the head end on bus A, and from the early studies of DQDB we would expect it to have a higher access delay than does station 7. This is not the case in our study. Station 8 enjoys the use of both bus A and B for communicating with the other two stations within its own community of interest. This permits station 8 to have two slots in DQDB's distributed queue at one time which produces smaller access delay values than station 7 displays.

This problem of drastic unfairness towards certain stations within small to medium sized communities of interest must be addressed. One of the perceived principal uses of the DQDB MAN will be to act as a Public MAN to economically connect these types of users. This DQDB based Public MAN will be expected to treat all customers equally. As the standard now exists each customer will be paying an equal fee for what will be far from equal service.

To correct the unfairness, it was attempted to separate the two types of unfairness to determine the effects of each. The traffic direction unfairness was removed by increasing the offered load of all active stations on each bus to the point of trying to obtain all of the bus capacity. The BWB mechanism was disabled for this part of the study. The differing amounts of bus capacity obtained by each station would be a function of the station's position on the bus. This will give a network planner a feel for the unfairness caused by the station's position. This is not an exact science because it has been shown that several other factors will affect bandwidth distribution. As an example, the initial ratio of busy to free slots when the experiment is completed will affect the amount of bandwidth obtained by each

NODE #	Bus A % Slots attempted	Bus A % Slots Obtained	Bus A % Slots Adv/disadv	Ranking Most to Least Advantaged stations	V_BWB MODULUS ASSIGNED
1	22.2	25.0	2.8	3	7
2	11.1	22.8	11.7	1	3
3		Not	active	on bus A	
4	22.2	15.9	-6.3	5	8
5	11.1	14.7	3.6	2	4
6		Not	active	on bus A	
7	22.2	11.3	-10.9	6	64
8	11.1	10.2	-.9	4	7
9		Not	active	on bus A	

Table 5.1: Station performance Comparisons

station. I initialized all slots to be not busy and all stations were turned on at the same time[WON89, VAN90A].

Table 5.1 gives the portions of bandwidth obtained by each station on bus A, both buses are symmetric and thus only bus A will be described here. It is clear that the head-end stations do obtain a significantly larger proportion of the DQDB bandwidth than do the down stream stations. Station 1 obtained 25.0% of bus A's bandwidth compared to station 8's 10.2%. To determine which stations are working from an unfair advantage, we must compare the amount of bandwidth obtained by each station, to the amount of bandwidth each station should get on bus A. The capacity of DQDB should be equally divided among all 9 stations thus all should receive $\frac{1}{9}$th or 11.1% of the total DQDB bandwidth. Station 1 sends all its QA traffic on bus A, thus requiring 22.2% of bus A's capacity to obtain its 11.1% of the total DQDB bandwidth. Station 1 obtained 25.0% of bus A's capacity, therefore gaining 2.8% more of bus A's bandwidth than necessary. This gives station 1 an unfair advantage.

Similar types of results are given for all 6 active stations on bus A in table 5.1. It is clear from this table why station 7 is so poorly treated by DQDB. It requires 22.2% of bus A's bandwidth but has obtained only 11.3%. Station 7 is only able to obtain about half of its required bandwidth to service its needs and thus displays horrendous waiting times to access

the DQDB MAN. A visual indication of which stations are working from an advantage and which from a disadvantage is given in figure 5.5. Equipped with this information we may now rank the stations from most to least advantaged (see Table 5.1). The question now arises as to how we correct the unfairness between the stations without completely redesigning the DQDB MAN standard.

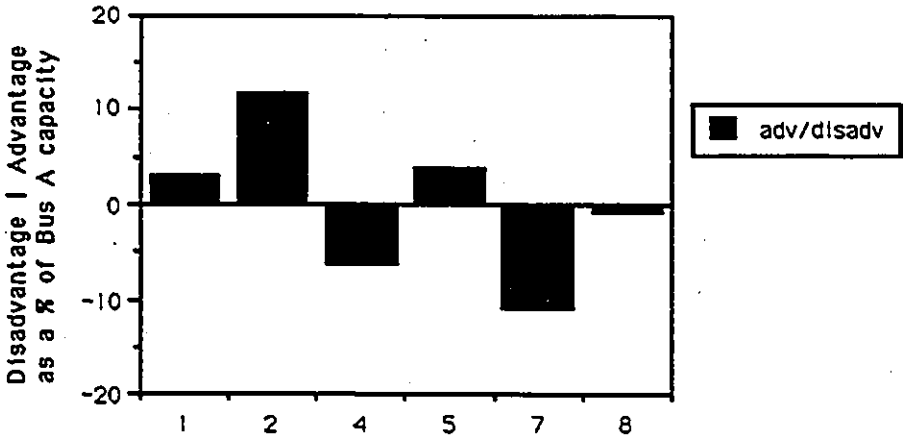


Figure 5.5: Visual Indication of Which Stations are Working From An Advantage and Which From a Disadvantage.

The solution is to modify the existing BWB mechanism in order to not treat all stations equally. This might sound strange, but why should station 7 be forced to give up one free slot for every 8 it uses, as the standard now calls for? This station is only able to obtain about 50% of its bandwidth requirements before bandwidth balancing is enabled. On the opposite side of the coin, station 2 will only be forced to let 1 slot pass for every 8 slots it uses. In my study station 2 was able to obtain 100% more bandwidth than it required on bus A and in reality should allow more than 1 in 9 slots pass by unused.

The modification to the standard involves selection of a different BWB_MOD for each active access node based on the ranking of each access node in the most to least advantaged list (See Table 5.1). I refer to this as variable bandwidth balancing (V_BWB) because the

balancing modulus will vary from station-to-station, and in fact from bus to bus for the same station. This method was introduced in the last chapter to overcome the unfairness of the Private DQDB MAN.

As an example of the V_BWB_MOD differing between the two access nodes of the same station we may look at station 2. Station 2's access node on bus A, as just discussed, is enjoying a great advantage and was assigned a V_BWB_MOD of 3. This will cause this access node to let a free slot pass by unused for every 3 it uses. Station 2's access node on the opposite bus B does not similarly enjoy the privilege of being so close to the head-end and therefore cannot be forced to give up a slot for every 3 it uses. Station 2's bus B access node is symmetrical to that of station 7 on bus A and both will receive the same V_BWB_MOD = 7. Figure 5.6 gives the complete DQDB MAN with the V_BWB_MOD chosen for each of the 12 access nodes required to service our 3 communities of interest.

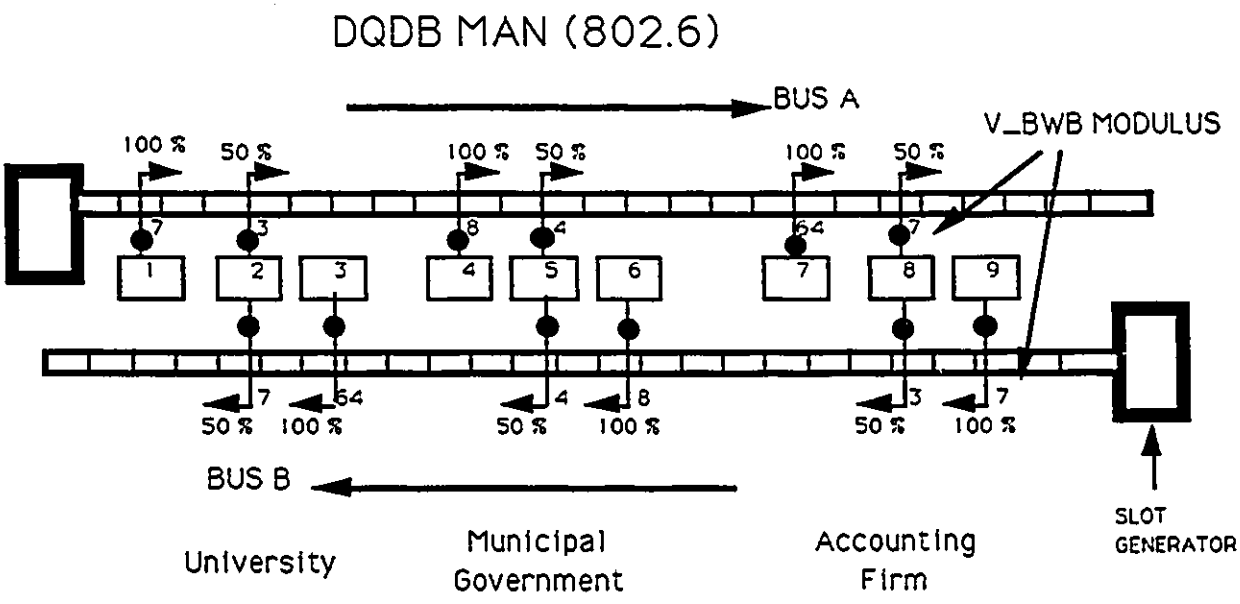


Figure 5.6: Three Communities of Interest interconnected with a DQDB Based MAN with Each Access Nodes's Assigned V_BWB_MOD.

5.4.3 Phase Three: V_BWB Test Results and Optimal Performance Approach

The final phase of the study was to investigate the performance of the V_BWB based DQDB MAN under similar load conditions to that of the first phase. Figure 5.7 displays the dramatic improvement in fairness for the slot access delays of the different stations. The variable bandwidth balancing mechanism was successful in redistributing the bandwidth from the advantaged stations to the disadvantaged stations. No longer do we see any dramatic spikes in the graphs indicating unfairness for certain stations when accessing the DQDB MAN.

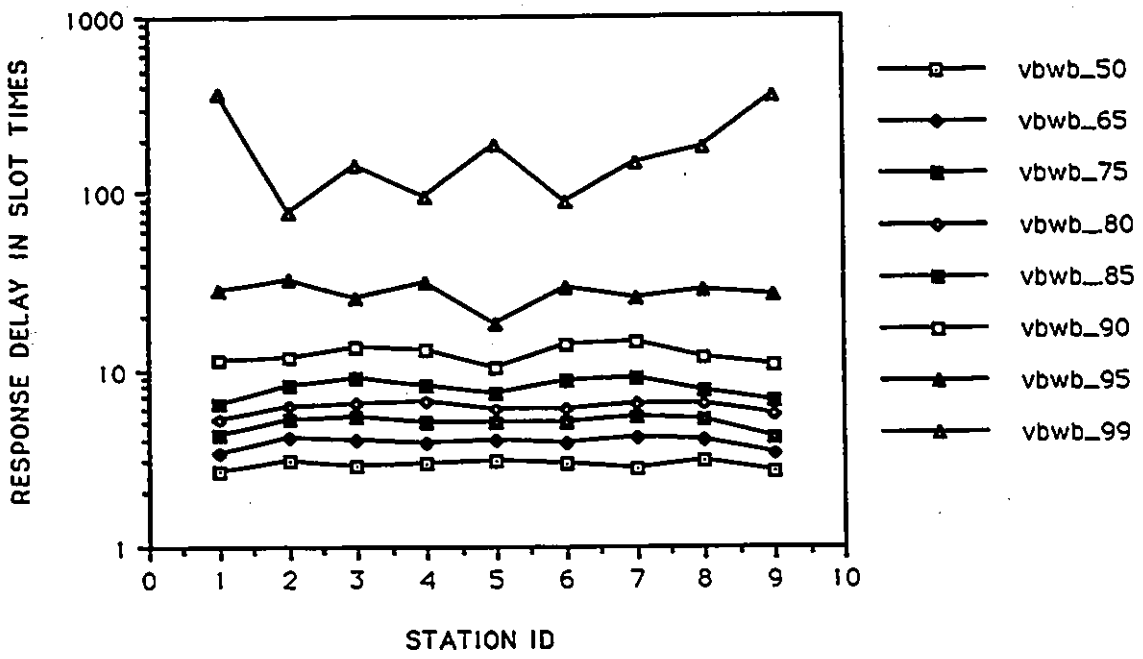


Figure 5.7: Station Mean Access Delays With The V_BWB mechanism enabled at all load levels.

It is clear from these results that this V_BWB technique may be used to significantly improve the fairness of a DQDB based Public MAN. When V_BWB is enabled the stations are forced to respect strict MAN access procedures. These procedures will cause some overhead

delays. The ideal situation would be to have a network free of overhead when controlling the bandwidth distribution between stations. It would be interesting to investigate the performance of this Public MAN operating without any form of BWB for comparison purposes.

Figure 5.8 displays the results of our study repeated with the BWB mechanism disabled. It is clear that this DQDB based Public MAN is unfair in allocating its bandwidth. It is interesting to note however that the unfairness is not as extreme as in the case with the DQDB default value for the BWB_MOD = 8. This shows that the default BWB mechanism produced a higher degree of unfairness than the case without any form of BWB (compare figures 5.8 and 5.4). This increased unfairness occurs when stations which are operating under both forms of disadvantage are forced to let a free slot to pass unused for every 8 taken even though they cannot afford to do so.

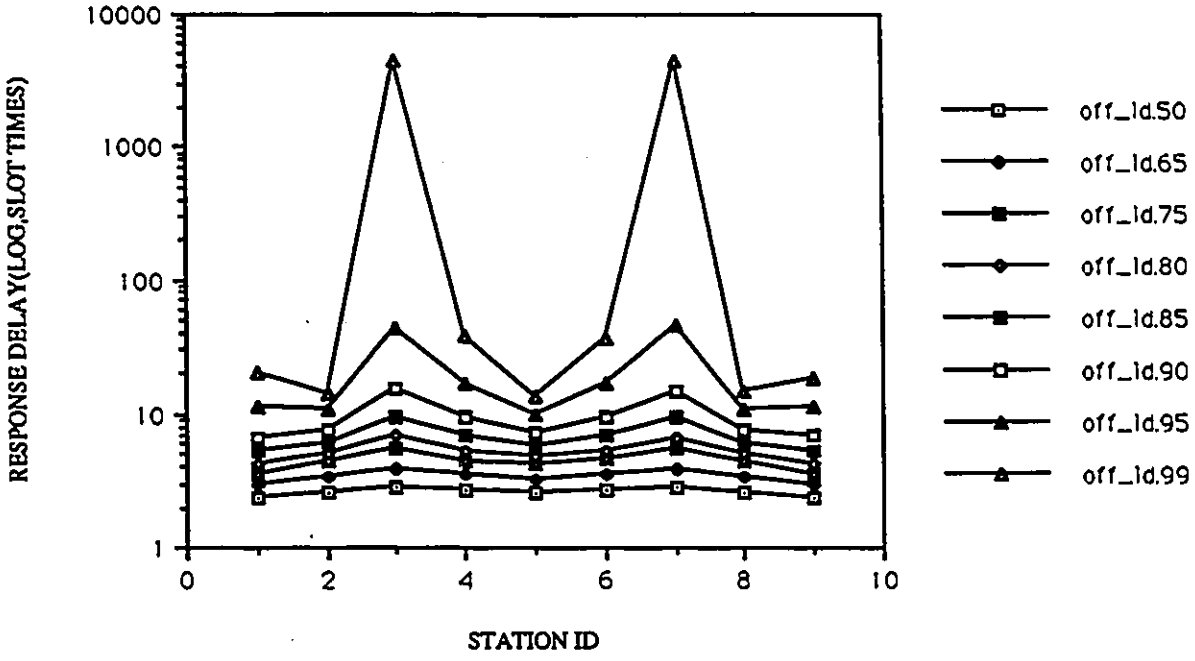


Figure 5.8: Station Mean Access Delays With No Form of BWB enabled.

The results of figure 5.8 also indicate that the DQDB protocol without any form of band-

width balancing is reasonably fair up to about 80% load. A network average access time of only 5.2 slot times was observed. The quickest station (1) accessing in 4.2 slot times and the slowest station (3) in 6.8 slot times without any form of bandwidth balancing. This result is quite significant in that we may allow the network to operate without any form of bandwidth balancing up to a load level of 80%. A forced bandwidth balancing mechanism at low to medium loads increases average access times and may, as we have just shown, increase unfairness. In our study at a load of 80% the average access time increased from 5.2 to 6.2 slot times and the unfairness increased dramatically by including the default BWB values when compared to using no BWB mechanism at all. (see figures 5.3 and 5.4).

My proposed solution to the unfairness observed by the DQDB Public MAN is to allow the DQDB MAN to operate without any form of BWB up to a load of 80%. When the MAN reaches the 80% load level on one of its buses the V_BWB mechanism will be enabled for that bus in order not to have any serious unfairness when dividing its bandwidth between active users.

The monitoring of the load can easily be done by the most downstream end station of the bus by counting the ratio of busy to empty slots as they are destroyed. When this end station determines that the load has reached 80% it will send a V_BWB enable signal to all stations on the busy bus. The enable V_BWB message can easily be sent by this station because this downstream end station on the busy bus is the head-end of the opposite bus and is in the perfect position to inform all active nodes on the busy bus. When the load drops below 80% the end station could send the disable V_BWB message to all others on the bus. These enable/disable messages should be included into the DQDB layer Management functions as a modification to the DQDB protocol. This 80% on/off value is not set in stone and should be determined by the network manager (ie.the carriers for these Public MANs). The enable/disable values may also be unequal. For example we may enable V_BWB at a 80% load level and disable it only when the bus load has reduced to 75%. The idea is similar to closing the LLC window size under periods of heavy load and then widening the window as the traffic conditions improve[BUX85].

Figure 5.9 displays the access delays for our DQDB Public MAN when the system is allowed to operate without BWB until the load becomes 80%. At this point the V_BWB mechanism is enabled. This figure shows clearly that the access times for all stations are quite fair for all load levels. The philosophy here is to allow the system run unrestricted and with as little overhead as possible, up to the point when things could become quite unfair. At this point strict operating parameters are enforced to ensure all stations have an equal chance at accessing the bus. This will provide the users with unrestricted access and minimal delays at low to medium loads. As the network becomes busy, strict operating procedures can be used to ensure fairness.

The addition of both the V_BWB mechanism and the enable/disable V_BWB messages to the DQDB standard will provide for optimal system performance. This will provide quick access times at low loads and orderly controlled accessed during heavy load conditions. These are not major modifications to the DQDB standard as the BWB mechanism already exists and thus only requires that each access node have its own V_BWB modulus. The enable/disable V_BWB messages could be added to the existing list of DQDB layer management messages which are already incorporated into the standard.

The costs of these proposed modifications to the standard must be investigated. The final figure 5.10 gives a comparison of the average access delays for all slots entering the MAN. I had shown earlier (see figure 5.9) that the station to station access delays had become relatively equal when V_BWB is enabled. It is important to show that the equality is not caused by simply making the fast stations slower. Figure 5.10 compares the average access delay for all slots from all stations when we use the DQDB default BWB parameters to the average delay when no BWB is used up to a 80% load level and then V_BWB is used above this value. In all cases the average access time is better than the default DQDB parameters. The performance is better at lower loads because there is no forced delays caused by the BWB mechanism. During heavy load conditions the performance is improved because the DQDB resources are redistributed according to need.

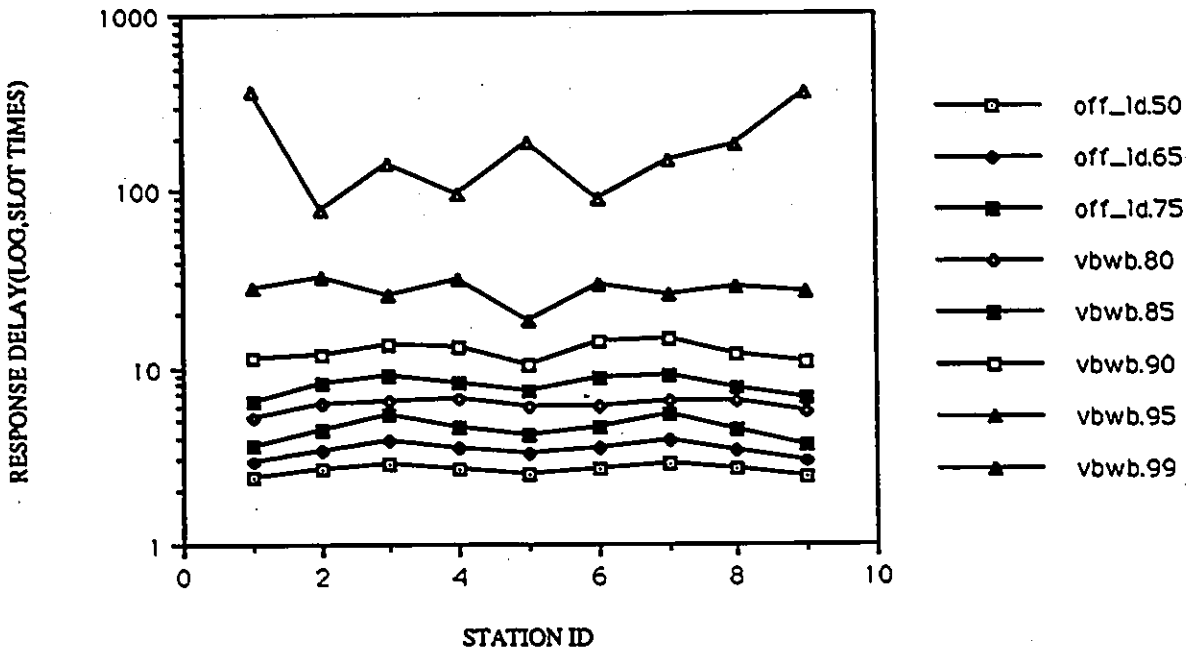


Figure 5.9: Station Mean Access Delays With The V_BWB mechanism enabled for bus loads above 80%.

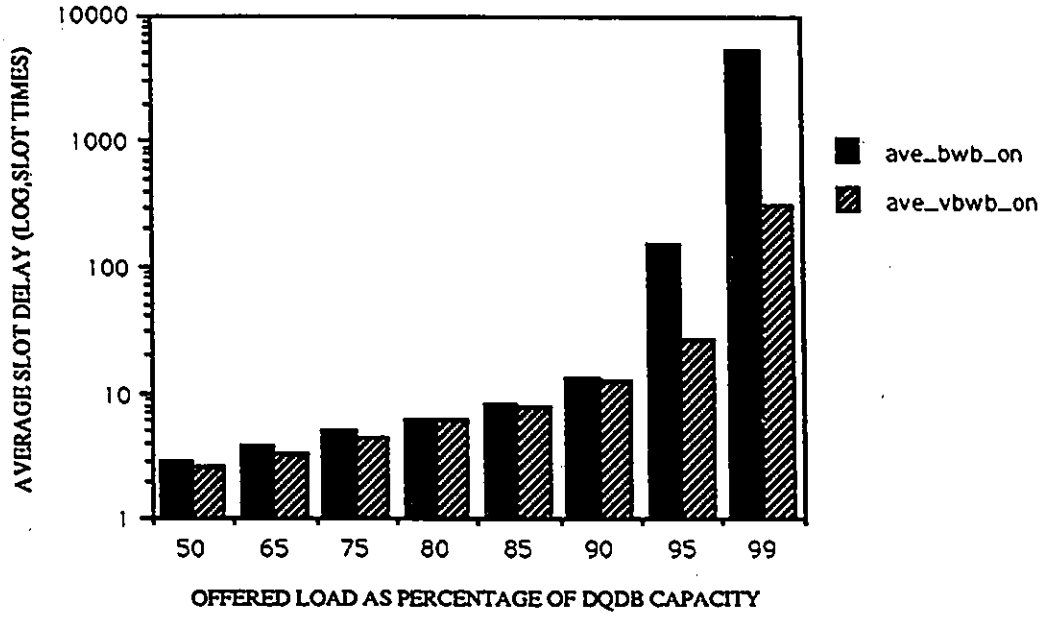


Figure 5.10: Global Access Delay Comparisons for the Current Standard vs a V_BWB based MAN where V_BWB will be enabled above 80% load.

5.5 Conclusion

In this chapter I have investigated the fairness and efficiency of a DQDB Public MAN to support multiple communities of interest. The results have shown that the standard as it now stands is unable to fairly divide the DQDB bandwidth among its stations. In fact, it has been shown that the BWB mechanism as it now stands may actually worsen the unfairness of DQDB to support this type traffic. A modification to the standard has been proposed to allow DQDB based MANS to operate without any form of bandwidth balancing up to a predetermined load (80% for our study). When this load value is reached the V_BWB mechanism will be enabled to ensure that all stations are given fair treatment by the standard under these heavy load conditions. The V_BWB mechanism will redistribute the DQDB bandwidth according to need rather than allowing a station's position along the bus and destination traffic patterns dictate the stations DQDB bandwidth allocation.

Chapter 6

Conclusions And Suggestions For Further Research

This thesis investigated the fairness and performance behaviour of the IEEE 802.6 DQDB MAN. The DQDB architecture has been accepted as the standard for the IEEE Metropolitan Area Network. Chapter 1 outlined how the DQDB MAN will take on the role of either that of a Private MAN supporting the needs of a large organization or that of a Public MAN which will be used by small to medium sized organizations as an interconnection backbone or collector network for the ATM based switches of the future.

Chapter 2 gave an overview of the OSI 7 layer model and outlined where the DQDB MAN will fit into this international standard. The need for a standardized MAN architecture was outlined, as were the goals and functional requirements for this new network. A review of the associated 802 standards which are related to the 802.6 MAN standard brought up the debate of minimal overhead with unpredictable performance versus complex protocols with very rigid performance characteristics. The positive and negative performance characteristics of these other 802 protocols gave an insight into the possible performance aspects of the IEEE DQDB MAN. This debate was carried through out the thesis and eventually led to the proposed optimal performance suggestions made in chapter 5. The suggestions would improve the DQDB MAN performance by balancing minimum overhead with rigid performance.

The DQDB architecture and protocol were described in detail in chapter 3. The history of

the DQDB protocol from the initial QPSX proposal to the standard in its present form was presented. The standard has undergone several major modifications during the complex and lengthy standardization process. Most of the modifications have been attempts to improve on the standards unfair allocation of asynchronous bandwidth during certain traffic conditions. The problem again arose as to how much control we can enforce on a network before the added overhead causes problems.

The performance study performed in chapter 3 showed that with the removal of the "standby" state from the protocol and with the addition of the BWB mechanism we observed a great deal more control information flowing through the network. We no longer observed the head-end stations enjoying the extremely unfair advantage of quick access times for their QA slots. My study also showed that during periods of heavy bus load the new BWB mechanism so severely restricts the head end stations that their performance radically changes from slightly advantaged in light loads to extremely disadvantaged during heavy load conditions. The problem is threefold for these end stations. The end stations can only use one of the two DQDB buses for sending slots. The BWB mechanism introduced to equally divide the DQDB bandwidth forces these unidirectional stations to give up slots even though they cannot afford to do so. And thirdly, the end stations will have to action many more requests with the removal of the "standby" state.

The fourth chapter investigated the fairness and performance aspects of a DQDB Private MAN to interconnect stations of three 802.5 Token Rings which forms a single community of interest. The LLC protocol is used as the end-to-end communication mechanism. Bridges were used to move the token ring I-frames over the DQDB MAN as QA slots. The study showed that it is important to give the bridges priority on the token rings to avoid buffer overflow. If buffer overflow does occur the system performance will radically degrade with packet retransmissions and time out messages. If congestion does occur in the bridges it is desirable to reduce the LLC window size to ease the congestion. It was shown that as long as you have sufficient buffer space, a large LLC window will provide for improved performance.

The most interesting observation of chapter 4 was that the DQDB MAN was unfair in its bandwidth distribution during periods of heavy load. The unfairness was a function of the dual bus architecture of DQDB. The middle token ring of the private community of interest enjoyed the use of both DQDB buses while the outside token rings were forced to use only one of DQDB's Two buses. The unfairness was overcome by implementing a modification to the DQDB BWB mechanism called variable bandwidth balancing (V_BWB).

The performance of LLC and DQDB working together showed how complicated systems become when several protocols must work together. It was difficult, at times, to separate the performance aspects of each protocol as they work so closely together. The major result of this chapter was that the current version of DQDB is unfair in its treatment of some stations, but this unfairness is correctable using V_BWB. This unfairness of DQDB to support a single community of interest on a Private MAN raised serious questions as to how DQDB would support multiple communities of interests acting as a Public MAN.

Chapter 5 investigated the DQDB architecture serving as a Public MAN to support three separate communities of interest. The focus again was on DQDB's performance and fairness. The results of chapter 5 have shown that the DQDB standard as it now stands is unable to fairly divide the DQDB bandwidth between stations when it supports multiple communities of interest. In fact, it has been shown that the BWB mechanism introduced to rectify the fairness problems my actually worsen the unfairness of DQDB to support these types of customers.

Chapter 5 compared the performance of the DQDB Public MAN to support these multiple communities of interest with and without the BWB mechanism. The results of the same Public MAN using the V_BWB mechanism to avoid the unfairness problems is also presented. When all the results were analyzed a suggestion for optimal performance was put forward. The suggestion returned us to the earlier debate of minimal network overhead with unpredictable performance versus complex protocols with very rigid performance characteristics.

Optimal system performance occurred in my study when I balanced these two approaches for network operation. During periods of light to medium load I suggest that the DQDB

network be permitted to operate with as little overhead as possible. This would require us to completely disable the BWB mechanism of the standard. It was shown in chapter 5 that the performance does slightly favour the headend stations but not to the point of justifying a very rigid control mechanism like BWB.

The DQDB bus utilization can be easily monitored by the most down stream end station as the slots are destroyed. When this station determines that the bus utilization is approaching a point where unfairness will be a problem, it could send a V_BWB enable message to all stations on the busy bus. After the congestion clears this same station could be used to disable the V_BWB control procedures.

This balance between rigid controls during heavy utilization and easy access during light loads is important for efficient and fair network performance. I personally believe that it is important that these, or a modified version of these changes, be incorporated into the DQDB standard. They give the network planners the required flexibility to properly manage their metropolitan area networks.

A rich area for future work stemming from this thesis would be to develop a method which dynamically determines the V_BWB moduli based on: (1) the current traffic conditions on each of the DQDB buses, (2) the number and locations of active stations, and (3) the dynamic traffic conditions of each station. I have determined the V_BWB moduli of my performance evaluations based on the relatively static values of station location and average traffic conditions.

An extension to a completely dynamic system will be crucial when DQDB based MANs will be used to support very fast workstations with very bursty traffic conditions. It is envisioned that the DQDB MAN will be one of the principal traffic collectors for the ATM switches of the future and this "Dynamic Variable Bandwidth Balancing" (D_V_BWB) mechanism will be required to ensure adequate service to these high speed work stations. This expansion of the concepts discussed in this thesis will provide for much more flexibility in the networks of the future. In fact this type of dynamic bandwidth control will be essential for virtually all

future ATM based networks which will have to continuously deal with bursty types of traffic sources attempting to gain access to an ATM switch.

Bibliography

[AND84] F. T. Andrews, "ISDN '83", *IEEE Communications Magazine*, vol. 22, no. 1, Jan. 1984, pp. 6-10.

[ANS85A] ANSI/IEEE Standard 802.2, "Logical Link Control", IEEE Project 802, IEEE, New York, 1985.

[ANS85B] ANSI/IEEE Standard 802.3, "CSMA/CD Access Method and Physical Layer Specification", IEEE Project 802, Local Area Network Standards, IEEE, New York, 1985.

[ANS85C] ANSI/IEEE Standard 802.4, "Token Bus Access Method and Physical Layer Specification", IEEE Project 802, Local Area Network Standards, IEEE, New York, 1985.

[ANS85D] ANSI/IEEE Standard 802.5, "Token Ring Access Method and Physical Layer Specification", IEEE Project 802, Local Area Network Standards, IEEE, New York, 1985.

[BER87] D. Bertsekas and R. Gallager, Data Networks, *Prentice Hall Inc.*, New Jersey, 1987.

[BIS90] C. Bisdikian "Waiting Time Analysis in A Single Buffer DQDB(802.6) Network", *Proc. IEEE INFOCOM'90*, San Francisco, June 90, pp. 610-616,.

[BUD84] Z. L. Budrikis and A. N. Netravali, "A Packet/Circuit Switch", *Bell Systems Technical Journal*, vol. 63, no. 8, Oct. 1984, pp. 1499-1520.

[BUD86] Z. L. Budrikis, J. L. Hullett, R. M. Newman, D. Economou, F. M. Fozdar and R. D. Jeffery, "QPSX: A Queued Packet and Synchronous Circuit Exchange", *Proc. 8th Int. Conf. Comp. Comm.*, Munich, 1986, pp.288-293.

[BUX81] W. Bux, "Local-Area Subnetworks: A Performance Comparison", *IEEE Trans. on Comm.*, vol. COM-29, no. 10, Oct. 1981, pp. 1465-1473.

[BUX85] W. Bux and D. Grillo, "Flow Control in Local Area Networks of Interconnected Token Rings", *IEEE Trans. on Comm.*, vol. COM-33, no. 10, pp. 1058-1066, October 1985, reprinted in *Advances in Local Area Networks*, K. Kummerle, J. Limb and F. Tobagi Eds., New York: IEEE Press,1987,pp. 496-516.

[BUZ73] J. P. Buzen, "Computational Algorithms for Closed Queueing Networks with Exponential Servers", *Communications of the ACM*, vol. 16, no. 9, Sept 1973, pp. 527-531.

[CCIS8] CCITT Recommendation I.122, "Framework for Providing Additional Packet-Mode Bearer Services", July 1988.

[CON89] M. Conti, E. Gregori and L. Lenzini, "DQDB Media Access Control Protocol: Performance Evaluation and Unfairness Analysis", *Third IEEE Workshop on Metropolitan Area Networks*, San Diego, California, USA, March 1989.

[CON91] M Conti, E Gregori, L Luciano "A Methodogical Approach to an Extensive Analysis of DQDB Performance and Fairness", *IEEE JSAC*, vol 9 No. 1, January 1991, pp 76-87.

[CRA90] C. J. Cranfill, "ISSUES IN IMPLEMENTING A PUBLIC METROPOLITAN AREA NETWORK", *ICC'90*, Atlanta, Georgia, Apr. 1990, pp. 1572-1575.

[DAV89] P. Davids, "Performance Analysis of DQDB based on Simulation", *Third IEEE Workshop on Metropolitan Area Networks*, San Diego, California, USA, March 1989.

[DES90] A. DeSimone and V. R. Saksena, "PERFORMANCE OF ALTERNATE MAN ARCHITECTURES FOR HIGH-SPEED DATA APPLICATIONS", *ICC'90*, Atlanta, Georgia, Apr. 1990, pp. 1584-1590.

[EMM83] Willard F. Emmons and A. S. Chandler, "OSI Session Layer: Services and Protocols", *Proc. IEEE*,vol. 71, no. 12,Dec. 1983, 1397-1400.

[FDI90] S. FDIDA and H. SANTOSO, " Performance Issues of DQDB", *"Fourth International Conference on Data Communications Systems and Their Performance"*., Barcelona, June, 1990.

[FIE86] J. A. Field, "Logical Link Control", *Proc. INFOCOM'86*, Miami, Florida, USA, April 1986, pp. 331-336.

[FIL90] J. Filipiak, "ACCESS PROTECTION FOR FAIRNESS IN A DISTRIBUTED QUEUE DUAL BUS METROPOLITAN AREA NETWORK", *ICC'89*, Boston, Mass, June. 1989, Paper 20.4.1, pp. 635-639.

[GAR90] M. W. Garrett and S. Li, "A Study of Slot Reuse in Dual Bus Multiple Access Networks", *Proc. IEEE INFOCOM'90*, San Francisco, June 90, pp 617-629.

[GEO89] N. D. Georganas, M. Goldberg, L. Orozco-Barbosa and J. Masteronardi, "A Multimedia Communications Systems foe Medical Applications", *Proc ICC'89*, Boston, June 1989.

[GER80] M. Gerla and L. Kleinrock, "Flow Control: A Comparative Survey", *IEEE Trans. on Comm.*, vol. COM-28, April 1980, pp. 553-574.

[GER88] M. Gerla and L. Kleinrock, "Congestion Control in Interconnected LANs", *IEEE Network*, vol. COM-2, no. 1, Jan. 1988, pp. 72-76.

[GLA89] B. Glass, "Under the Hood: The Token Ring", *Byte*, vol. 14, no. 1, Jan.1989, pp. 363-376.

[HAH90] E. Hahne, A. Choudhury, N. Maxemchuck, "IMPROVING THE FAIRNESS OF DISTRIBUTED-QUEUE-DUAL-BUS NETWORKS", *Proc. IEEE INFOCOM'90*, San Francisco, June 90, pp175-184.

[HOL83] Lloyd L. Hollis, "OSI Presentation Layer Activities", *Proc. IEEE*,vol. 71, no. 12,Dec. 1983, 1401-1403.

[IBM88] IBM Canada Ltd, "New LAN Technology Quadruples Speed of Token Ring Data Transmissions", *IBM Solutions*, no. 5, 1988, pp. 14-15.

[IEE85] Draft Standard IEEE P802.6 Metropolitan Area Network, "Media Access Control Specifications", IEEE 802.6 Document 802.6/85-D, Revision D, August 1985.

[IEE87A] Draft Standard IEEE P802.6 Metropolitan Area Network (MAN) Dual Buses, "Media Access Control (MAC) and Physical Layer Protocol Documents", Draft B.0, July 1987.

[IEE88A] Minutes from the March, July and Sept. 1988 meetings of IEEE 802.6 Metropolitan Area Network Working Group.

[IEE88B] Draft Standard IEEE P802.6, Distributed Queue Dual Buses Metropolitan Area Network (DQDB MAN), "Media Access Control (MAC) and Physical Layer Protocol Documents", Draft D.6, Nov. 1988.

[IEE89] Draft Standard IEEE P802.6, Distributed Queue Dual Buses Metropolitan Area Network (DQDB MAN), "Media Access Control (MAC) and Physical Layer Protocol Documents", Draft D.8, June. 1989.

[IEE89B] Draft Standard IEEE P802.6, Distributed Queue Dual Buses Metropolitan Area Network (DQDB MAN), "Media Access Control (MAC) and Physical Layer Protocol Documents", Draft D.10, Oct. 1989.

[IEE90] Draft Standard IEEE P802.6, Distributed Queue Dual Bus (DQDB) Subnetwork of a Metropolitan Area Network (MAN), Draft D.14, July 1990

[IEE89] Draft Standard IEEE P802.9, Integrated Voice/Data LAN Interface, "Media Access Control (MAC) and Physical Layer Specifications", Draft D3, Mar. 1989.

[ISO86] ISO/TC97/SC6/WG1, "Draft Proposed Addendum to ISO DIS 8802/2 Logical Link Control-Ackowledgement Connectionless Service", 14th Draft, June 1986.

[JUL86] U. de Julio, "Layer 1 ISDN Recommendations", *IEEE JSAC*, vol. SAC-4, no. 3, May 1986, pp. 349-354.

[KANS6] S. Kano, "Layers 2 and 3 ISDN Recommendations", *IEEE JSAC*, vol SAC-4, no. 3, May 1986, pp. 355-359.

[KAU90] Harbans Kaur, Grahan Campbell "DQDB -AN ACCESS DELAY ANALYSIS",*Proc. IEEE INFOCOM'90*, San Francisco, June 90, pp 630-635.

[KIM90] B. G. Kim, "Packet Delays in the IEEE 802.6 DQDB Protocol", *ICC'90*, Atlanta, Georgia, Apr. 1990, pp. 1692-1696.

[KLE75] L. Kleinrock, Queueing Theory - Volume I: Theory, *Wiley-Interscience*, New York, 1975.

[KNI83] K. G. Knightson, "The Transport Layer Standardization", *Proc. IEEE*, vol. 71, no. 12, Dec. 1983, pp. 1394-1396.

[LAI89] W. S. Lai, "Frame Relaying Service: An Overview", *Proc. INFOCOM'89*, Ottawa, Canada, April 1989, pp. 668-673.

[LIM82] J. O. Limb and C. Flores, "Description of Fasnet, A Unidirectional Local Area Communications Networks", *Bell System Technical Journal*, Sept. 1982, , pp.1413-1440.

[LIN83] P.F. Linington "Fundamentals of the Layer Service Definition and Protocol Specifications", *Proc IEEE*, vol 71 No. 12, Dec 1983, pp 1341-1345.

[LOM89] A. Lombardo and S. Palazzo" AN ARCHITECTURE FOR A PURE ATM METROPOLITAN AREA NETWORK", *GLOBECOM'89*, Dallas, Texas, Nov. 1987, pp. 1828-1834.

[MAT89] B. Materna, B. Vaughan and C. Britney "EVOLUTION FROM LAN AND MAN ACCESS NETWORKS TOWARDS THE INTEGRATED ATM NEWTWORK", *GLOBECOM'89*, Dallas, Texas, Nov. 1987, pp. 1455-1461.

[MOL88A] J. F. Mollenauer, "Standards for Metropolitan Area Networks", *IEEE Communications Magazine*, vol. 26, no. 4, Apr. 1988, pp. 15-19.

[MOL88B] J. F. Mollenauer, "Status of the IEEE 802.6 MAN Standards", *13th Conf. on Local Computer Networks*, Minneapolis, USA, 1988, pp. 166-169.

[MOU88] J. Moughton, "PHY and MAC Layer Modeling Considerations", IEEE Documents 802.9-88, July 1988.

[NEW86] R. M. Newman and J. L. Hullett, "Distributed Queueing: A Fast and Efficient Packet Access Protocol for QPSX", *Proc. 8th Int. Conf. Comp. Comm.*, Munich, 1986, pp. 294-299.

[NEW88A] R. M. Newman, "Distributed Queueing Performance Characterisation", IEEE Documents 802.6-88/16, March 1988.

[NEW88B] R. M. Newman, Z. L. Budrikis and J. L. Hullett, "The QPSX MAN", *IEEE Communications Magazine*, vol. 26, no. 4, April 1988, pp. 20-28.

[PAN88] J. Pang and F. A. Tobagi, "Throughput Analysis of a Timer-Controlled Token-Passing Protocol under Heavy Load", *Proc. INFOCOM'88*, New Orleans LA, USA, March 1988.

[PAN89] J. Pang and F. A. Tobagi, "Generalized Access Control Strategies for Token Passing Systems", *Proc. INFOCOM'89*, Ottawa, Canada, April 1989, pp. 332-341.

[QNA84] *QNAP2, Reference Manuel*, Version V03, Bull and INRIA, May 1984.

[REI80] M. Reiser and S. S. Lavenberg, "Mean-value Analysis of Closed Multichain Queueing Networks", *Journal ACM*, vol. 27, no. 2, April 1980, pp. 313-322.

[ROD90] M. A. Rodrigues, "Erasure Node: Performance Improvements for the IEEE 802.6 MAN",*Proc. IEEE INFOCOM'90*, San Francisco, June 90, pp 636-643.

[RUB90] I. Rubin and T. Cheng, " PERFORMANCE OF TRAFFIC MANAGEMENT STATEGIES FOR INTERCONNECTED HIGH-SPEED LOCAL AND METROPOLITAN AREA NETWORKS", *ICC'90*, Atlanta, Georgia, Apr. 1990, pp. 986-990.

[RYB80] A. Rybczynski, "X.25 Interface and End-to-End Virtual Circuit Service Characteristics", *IEEE Trans. on Comm.*, vol COM-28, no. 4, April 1980, , pp. 500-509.

[RYD89] M. Ryder, "Protocols for ATM Access Methods", *IEEE Network Magazine*, vol. 3, no. 1, Jan. 1989, pp. 17-22.

[SCH87] M. Schwartz, Telecommunication Networks, *Addisson-Wesley Publishing Company*, 1987.

[SHO79] J. F. Shoch, "Packet Fragmentation in Inter-Network Protocols", *Computer Networks*, vol. 3, 1979, pp. 3-8.

[STU83] P.vonStudnitz, "Transport Protocols: There Performance and Status in International Standardization", *Computer Networks*, vol. 7, 1983, pp. 27-35.

[TAN88] A. S. Tanenbaum, Computer Networks, *Prentice Hall Inc.*, New Jersey, 1988.

[VAL89] Richard Vallée, Luis Orozco-Barbosa and N. D. Georganas, "Interconnection of Token Rings by IEEE 802.6 Metropolitan Area Networks", *INFOCOM'89*, Ottawa, Canada, April 1989, pp. 934-942.

[VAN90A] H.R. van As, J. Wong, P. Zafiropulo,"FAIRNESS,PRIORTY AND PREDICTABILITY OF THE DQDB MAC PROTOCOL UNDER HEAVY LOAD",*1990 Int'l Seminar on Digital Communication,"Electronic Circuits and Systems for Communication"*, March 5-8, 1990, Zurich, Switzerland.

[VAN90B] H.R. van As. " Performance Evaluation of Bandwidth Balancing in the DQDB Mac Protocol",*EFOC/LAN 90*, Amsterdam, June 1990, Paper 5.3.2, pp 231-239.

[WARS3] Christine Ware, "The OSI Network Layer: Standards to cope with the Real World", *Proc. IEEE*, vol. 71, no. 12, Dec. 1983, pp. 1384-1387.

[WAI89] N. Wainwright and A. Myles, "A Comparison of The Delay Characteristics of the FDDI and IEEE 802.6 MAC Layer Protocols", *Third IEEE Workshop on Metropolitan Area Networks*, San Diego, California, USA, March 1989.

[WONS9] J. Wong, "Throughput of DQDB Networks under heavy Load",*EFOC/LAN 89*, Amsterdam, June 1989, Paper 1.4.3, pp 146-151.

[WON90] L. N. Wong and M. Schwartz, " ACCESS CONTROL IN METROPOLITAN AREA NETWORKS", *ICC'90*, Atlanta, Georgia, Apr. 1990, pp. 1591-1595.

[ZIM80] H. Zimmermann, "OSI Reference Model - The OSI Model of Architecture for Open Systems Interconnection", *IEEE Trans. on Comm.*, vol. COM-28, no. 4, April 1980, pp. 425-432.

[ZIM83] H. Zimmermann, "The OSI Reference Model", *Proc. IEEE*, vol. 71, no. 12, Dec. 1983, pp. 1334-1340.

[ZUK87] Moshe Zukerman, "On Packet Switching Capacity in QPSX", *GLOBECOM'87*, Tokyo, Japan, Nov. 1987, pp. 1777-1781.

[ZUK88A] Moshe Zukerman, "Overload Control of the Isochronous Traffic in QPSX", *GLOBECOM'88*, Hollywood, Florida, USA, Nov. 1988, pp. 1241-1245.

[ZUK88B] M. Zukerman, "Approximations for Performance Evaluation of the Packet Access Queue in QPSX:The IEEE 802.6 Evolving Man Standard",*Australian Communications Research* vol 22, No. 2 1988.

[ZUK88C] M. Zukerman, "Queueing Performance of QPSX",*Proc of the 12th International Teletraffic Congress ITC 12*, Torino, Italy, June, 1988.

[ZUK88C] M. Zukerman, "QPSX- THE EFFECT OF CIRCUIT ALLOCATION ON PACKET CAPACITY UNDER BURSTY TRAFFIC", *ICC'88*, Philadelphia, Penn., June. 1988, Paper 20.1, pp. 599-603.

Appendix A

QNAP2 Overview

The simulations performed for this thesis used the QNAP2 (Queue Network Analysis Package) software package[QNA84]. This package is used to evaluate the performance of queueing network models of specified systems under investigation.

My study used QNAP2 to evaluate the performance of DQDB in supporting various MAN based networks. These MAN networks were modeled as queueing networks. QNAP2 allows the user to define the network as a set of stations through which customers will pass according to a set of rules defined by the QNAP2 package user. This makes QNAP2 ideal for simulating a DQDB based MAN. The access nodes of DQDB may be simulated as stations. These stations will circulate customers or in our case DQDB slots according to the DQDB protocol which is detailed by the set of rules written to provide the DQDB distributed queueing protocol. Similar models were developed for the Token rings and Bridges.

QNAP2 provides for a specification language (based on Pascal) which is used for describing the models under study and also to write the set of rules for customer station interaction. QNAP2 incorporates known analytical solutions of queueing networks such as the MVA (mean value analysis) [REI80] and convolution algorithms [BUZ73]. It also incorporates a discrete-event simulator which was used for my performance evaluations during this research.

In QNAP2, a queueing network consists of a set of stations (each typical station comprising of one server and one queue) through which customers circulate according to given rules. The

processing done by each station may be as simple as a time delay or it could be a complex algorithm which may include synchronization operations. This algorithm will be written by the package user.

For each station, some parameters must be defined: the "type" of the station (source, server, resource); the "scheduling" scheme performed by the station (FIFO, priority, processor sharing); the "service" performed by the station (exponential, constant, or user defined); and finally the "next station" where the customer in service is going to be moved (using the TRANSIT command). During my study the access nodes would pass the customer (slot) to the next access node on each bus.

The results produced by QNAP2 give the mean service time, the utilization, the mean number of customers queued and in service for each station. QNAP2 also provides the mean response time and the number of customers served from the beginning of the simulation to the end. These results provided me with the mean waiting time for my DQDB slots and other important information to evaluate the performance of the DQDB based MAN.

QNAP2 provides for confidence intervals which may be produced with a confidence level of 95% for simulation results. The confidence intervals produced are themselves estimates because the computation of exact confidence intervals would require a prior knowledge of the estimated characteristics.

QNAP2 includes three methods for confidence interval estimation:

- The **replication method** which consists of generating several simple paths of the model studied. These sample paths are statistically independent and identical. This is achieved by reseting the original initial state of the model at the start of each replication, and by using a different random number stream for each replication.
- The **regeneration method** splits the simulation run into several successive intervals. The model behavior for these intervals will be statistically equivalent and independent.

- The **spectral method** applies to correlated samples provided an identical distribution property is satisfied. My performance studies utilized this **spectral method** to achieve confidence intervals of 95% for all simulations.

Appendix B

Program listings

B.1 Simulation Program Used for Both Chapters Three and Five

```
&************************
This program was used for both chapters three and five of my thesis.
&************************

/CONTROL/ OPTION =NSOURCE;
        CLASS = ALL QUEUE;
        ACCURACY = ALL QUEUE;
/DECLARE/
INTEGER   au = 9,                  & number of access units
          pl = au+1,               & number of slot places
          q = au-1,                & number of inter-stations
          b = 2,                   & number of buses
          nb_fr_2 = 40,            & twice frame size for counter
          nb_fr = 20,              & number of frame used
&************************
          stn_buff = 20000,
          nb_na = 1,               & nb of NA slots/frame
          nb_qa = 11,              & nb of QA slots/frame
          bwb_flag = 0,
          bwb_ct(b,au),
          bwb_mod(b,au),           & bandwidth balanceing modulus
          erasure(b,au),           & erasure node locations
          qa_tot,
          qa_use,
          req_tot,
          req_use,
          d_util,
&*************************
          nb_sl = nb_na+nb_qa,     & number of slots/frame
          i,j,w,z,                 & counter variables
          ind = 4,                 & number of numbers within 1 %
          cd(b,au),                & initial of counters (REQ,CD)
          cd_ct(b,au),             & countdown counter
          req_ct(b,au),            & REQ counter
          rqm(b,au),               & REQ bit queue on contra bus
          sl_len=424;              & slot length (bits)
REAL      mdelay(ind),             & mean access delay
&*****************************
          offer=0.9,               & network load
&*****************************
          t_cus,                   & total of packets sent
          dummy,                   & dummy variable
          tdelay,                  & total delay
&*****************************
          bus_len=25,              & bus length in km
&*****************************
          fr_time=125E-6,          & frame period time in second
          latency=180E-9,          & station latency in seconds
          p_delay=5E-6,            & propagation delay per km
          fr_len=nb_sl*sl_len,     & frame length (bits)
          sl_gen=fr_time/nb_sl,    & slot generation period
          st_to_st=bus_len/q,      & distance between two stations
          n_to_n=p_delay*st_to_st, & node to node propagation delay
          speed=fr_len/fr_time,    & ring bit rate (bits/s)
          sl_time=sl_len/speed,    & slot length in seconds
          load=offer*b*nb_qa/au/fr_time, & inter-arrival time
          arrival=1/load;          & arrival rate from each stations
FLAG      send(b,au);              & to control sending of information
CLASS INTEGER  icl;                & class number
```

```
            sh,                 & dummy class
            a_que,d_que;        & packets sent by stations
QUEUE INTEGER   ib,             & bus identifibation
            icb,                & contra bus
            in,                 & station number
            inext,              & next station on the bus
            x,y;                & counter variable
QUEUE       head_a,head_b,      & head and end point
            a_switch(au),d_switch(au), & ascending and descending switches
            link,               & link between stations
            a_node(au),d_node(au),  & node which queues arrival packets
            station(au),        & stations generating packets
            shut;               & dummy station
CUSTOMER INTEGER bus,           & destination bus
            busy,               & busy bit
            next,               & next station on the bus
            req,                & req bit
            type;               & type of slot
REF CUSTOMER    SL(nb_fr_2,nb_sl);      & QA slot in a frameSL


&*****************************
& Head of bus station
&*****************************
& This station generates a sequence of slots comprised in a frame period of 125
& micro second.There are respectively "nb_na" NA slots and "nb_qa" QA slots
& which are genrated in each frame period.
&*****************************
/STATION/ NAME = head_a;
      SERVICE = BEGIN
            WHILE b=2 DO
             BEGIN
              FOR x:=1 STEP 1 UNTIL nb_fr DO & Frame generation
               BEGIN
                FOR y:=1 STEP 1 UNTIL nb_na DO & NA slot generation
                 BEGIN
                  SL(x,y):=NEW(CUSTOMER);
                  SL(x,y).type:=1;    & TYPE bit set to 1
                  SL(x,y).busy:=0;    & Idem for BUSY bit
                  SL(x,y).req:=0;
                  SL(x,y).bus:=1;
                  SL(x,y).next:=1;
                  CST(sl_gen);
                  req_tot := req_tot+1;
                  TRANSIT(SL(x,y),a_switch(1));
                 END;
                FOR y:=nb_na+1 STEP 1 UNTIL nb_sl DO
                 BEGIN              & QA slot generation
                  SL(x,y):=NEW(CUSTOMER);
                  SL(x,y).type:=0;    & TYPE bit reset to 0
                  SL(x,y).busy:=0;    & BUSY bit reset to 0
                  SL(x,y).req:=0;
                  SL(x,y).bus:=1;
                  SL(x,y).next:=1;
                  CST(sl_gen);
                  req_tot := req_tot+1;
                  qa_tot := qa_tot+1;
                  TRANSIT(SL(x,y),a_switch(1));
                 END;
               END;
```

```
            END;
          END;
      TRANSIT = head_a;
      INIT(slot) = 1;


/STATION/ NAME = head_b;
      SERVICE = BEGIN
            WHILE b=2 DO
             BEGIN
              FOR w:=nb_fr + 1 STEP 1 UNTIL nb_fr + nb_fr DO & Frame generation
               BEGIN
                FOR z:=1 STEP 1 UNTIL nb_na DO & NA slot generation
                 BEGIN
                  SL(w,z):=NEW(CUSTOMER);
                  SL(w,z).type:=1;    & TYPE bit set to 1
                  SL(w,z).busy:=0;    & Idem for BUSY bit
                  SL(w,z).req:=0;
                  SL(w,z).bus:=2;
                  SL(w,z).next:=au;
                  CST(sl_gen);
                  req_tot := req_tot+1;
                  TRANSIT(SL(w,z),d_switch(au));
                 END;
                FOR z:=nb_na+1 STEP 1 UNTIL nb_sl DO
                 BEGIN                & QA slot generation
                  SL(w,z):=NEW(CUSTOMER);
                  SL(w,z).type:=0;    & TYPE bit reset to 0
                  SL(w,z).busy:=0;    & BUSY bit reset to 0
                  SL(w,z).req:=0;
                  SL(w,z).bus:=2;
                  SL(w,z).next:=au;
                  CST(sl_gen);
                  req_tot := req_tot+1;
                  qa_tot := qa_tot+1;
                  TRANSIT(SL(w,z),d_switch(au));
                 END;
               END;
             END;
           END;
      TRANSIT = head_b;
      INIT(slot) = 1;


&*****************************************
& QA slot access control in each station situated along the buses
&*****************************************
& This station simulating the distributed queueing protocol controlling the
& the access to the QA slots. Slots(customers), coming from the previous
& station situated on the bus, are examined. In particular, their REQ, TYPE
& and BUSY bits are checked. From the contents of these bits, REQ and
& countdown counters may be modified.
&*****************************************
$MACRO switch (return)
SERVICE = BEGIN
      IF req=1
      THEN    & if REQ bit equal 1 then REQ counter on the access
       BEGIN  & queue sending segments on the other bus is incremented
        req_ct(icb,in):=req_ct(icb,in)+1;
```

```
        END
       ELSE                & REQ bit is equal to zero
        BEGIN
         IF rqm(icb,in)>=1    & Does the AQ have a REQ bit to send**
         THEN               & on the reverse bus ?
          BEGIN
           req:=1;          & Set the REQ bit
           rqm(icb,in):=rqm(icb,in)-1;
          END;
        END;
       IF type=0            & Is the slot is a QA slot ?
       THEN
        BEGIN
         IF busy=0          & Is the QA slot empty ?
         THEN
          BEGIN
           IF cd(ib,in)=1   & Is the AQ in countdown  state ?
           THEN
            BEGIN
             IF cd_ct(ib,in)=0  & Is the CD counter equal to zero ?
             THEN
              BEGIN
               IF bwb_ct(ib,in) = bwb_mod(ib,in)
               THEN
                 BEGIN
                 bwb_ct(ib,in) := 0;
                 END
               ELSE
                 BEGIN
                 IF bwb_flag = 1 THEN   & bwb on/off test
                 bwb_ct(ib,in) := bwb_ct(ib,in) + 1;
                 busy:=1;        & The QA slot is set BUSY
                 cd(ib,in):=0;   & The AQ is moved to the CD state
& PRINT("insie switch setting send ");
                 SET(send(ib,in));
                 RESET(send(ib,in));
                 END;
               END
             ELSE cd_ct(ib,in):=cd_ct(ib,in)-1; & CD counter decremented
            END
           ELSE
            BEGIN
             IF req_ct(ib,in)> 0 & Is the REQ counter equal to zero ?
             THEN req_ct(ib,in):= req_ct(ib,in)-1; & REQ counter is
            END;                                   & decremented
          END;
        END;
       IF inext > 0
       THEN
        BEGIN
         IF inext = pl
         THEN
          BEGIN
          IF busy = 1
          THEN qa_use := qa_use +1;
          IF req = 1
          THEN req_use := req_use +1;
          TRANSIT(OUT);  & Slot sent to the end of the bus
          END
         ELSE
```

```
            BEGIN
             CST(latency);
             next=inext;          & Slot sent to the station on the bus
             IF erasure(ib,in) = 1 THEN
               BEGIN
                   CST(sl_gen);
                   busy := 0;
               END;
             TRANSIT(link);
             END;
          END
        ELSE
            BEGIN
            IF busy = 1
            THEN qa_use := qa_use +1;
            IF req = 1
            THEN req_use := req_use +1;
            TRANSIT(OUT);   & Slot sent to the end of the bus
            END;
          END;
SEND

&*****************************************
& Stations sending on bus A.
&*****************************************
/STATION/ NAME = a_switch(1 STEP 1 UNTIL au);
        $switch (a_switch)
&*****************************************
& Stations sending on bus B.
&*****************************************
/STATION/ NAME = d_switch(1 STEP 1 UNTIL au);
        $switch (d_switch)
& DQDB buses.
&*****************************************
& This station simulates the propagation delay for transferring the slots
& between the stations.
&*****************************************
/STATION/ NAME = link;
TYPE=INFINITE;
SERVICE =  BEGIN
          CST(n_to_n);
          IF bus=1
          THEN TRANSIT(a_switch(next))
          ELSE TRANSIT(d_switch(next));
         END;


&*****************************************
& End of the bus.
&*****************************************
& This station received, reinitializes the slots. Then, this returns the slots
& on the reverse bus.
&*****************************************
&*****************************************
& QA slot access queue
&*****************************************
& This station, in common work with the "SWITCH" station, simulates the
& the distributed protocol controlling the access to the QA slots.
&*****************************************
$MACRO node (class)
```

```
SERVICE(class)= BEGIN
            rqm(ib,in):=rqm(ib,in)+1; & must set an empty REQ bit
            cd(ib,in):=1;          & AQ moved in countdown state
            cd_ct(ib,in):=req_ct(ib,in);    & CD counter = REQ counter
            req_ct(ib,in):=0;       & REQ counter is cleared
            WAIT(send(ib,in));      & Wait the next QA slot
            CST(sl_time);           & Segment transmission
            TRANSIT(OUT);
           END;
SEND

&*****************************************
& Access queue for sending on bus a.
&*****************************************
/STATION/ NAME = a_node;
       Snode (a_que)

&*****************************************
& Access queue for sending on bus B.
&*****************************************
/STATION/ NAME = d_node;
       Snode (d_que)


&*****************************************
& Source station
&*****************************************
& This station generates asynchronous segments with a time exponentially
& distributed between each segment generation.
&*****************************************
/STATION/ NAME = station(1);
       TYPE = SOURCE;
       SERVICE = BEGIN
              EXP(arrival);
              IF (a_node(in).NB < stn_buff) THEN
              TRANSIT (a_node(in),a_que)
              ELSE TRANSIT(OUT);
              END;

/STATION/ NAME = station(2);
       TYPE = SOURCE;
       SERVICE = BEGIN
              EXP(arrival);
              IF DRAW (0.5) THEN
                BEGIN
                IF (a_node(in).NB < stn_buff/2) THEN
                TRANSIT (a_node(in),a_que)
                ELSE TRANSIT(OUT);
                END
              ELSE
                BEGIN
                IF (d_node(in).NB < stn_buff/2) THEN
                TRANSIT (d_node(in),d_que)
                ELSE TRANSIT(OUT);
                END;
               END;

/STATION/ NAME = station(3);
       TYPE = SOURCE;
       SERVICE = BEGIN
```

```
            EXP(arrival);
            IF (d_node(in).NB < stn_buff) THEN
            TRANSIT (d_node(in),d_que)
            ELSE TRANSIT(OUT);
            END;

/STATION/ NAME = station(4);
      TYPE = SOURCE;
      SERVICE = BEGIN
            EXP(arrival);
            IF (a_node(in).NB < stn_buff) THEN
            TRANSIT (a_node(in),a_que)
            ELSE TRANSIT(OUT);
            END;

/STATION/ NAME = station(5);
      TYPE = SOURCE;
      SERVICE = BEGIN
            EXP(arrival);
            IF DRAW (0.5) THEN
              BEGIN
              IF (a_node(in).NB < stn_buff/2) THEN
              TRANSIT (a_node(in),a_que)
              ELSE TRANSIT(OUT);
              END
            ELSE
              BEGIN
              IF (d_node(in).NB < stn_buff/2) THEN
              TRANSIT (d_node(in),d_que)
              ELSE TRANSIT(OUT);
              END;
             END;


/STATION/ NAME = station(6);
      TYPE = SOURCE;
      SERVICE = BEGIN
            EXP(arrival);
            IF (d_node(in).NB < stn_buff) THEN
            TRANSIT (d_node(in),d_que)
            ELSE TRANSIT(OUT);
            END;

/STATION/ NAME = station(7);
      TYPE = SOURCE;
      SERVICE = BEGIN
            EXP(arrival);
            IF (a_node(in).NB < stn_buff) THEN
            TRANSIT (a_node(in),a_que)
            ELSE TRANSIT(OUT);
            END;

/STATION/ NAME = station(8);
      TYPE = SOURCE;
      SERVICE = BEGIN
            EXP(arrival);
            IF DRAW (0.5) THEN
              BEGIN
              IF (a_node(in).NB < stn_buff/2) THEN
              TRANSIT (a_node(in),a_que)
```

```
            ELSE TRANSIT(OUT);
            END
          ELSE
            BEGIN
            IF (d_node(in).NB < stn_buff/2) THEN
            TRANSIT (d_node(in),d_que)
            ELSE TRANSIT(OUT);
            END;
           END;


/STATION/ NAME = station(9);
        TYPE = SOURCE;
        SERVICE = BEGIN
                EXP(arrival);
                IF (d_node(in).NB < stn_buff) THEN
                TRANSIT (d_node(in),d_que)
                ELSE TRANSIT(OUT);
                END;

$MACRO print_out
 BEGIN
 PRINT("TIME = ",TIME);
 PRINT(" ");
 PRINT("Access unit service ave both buses A+B (in slot unit)");
 PRINT(" ");
 FOR i:=1 STEP 1 UNTIL au DO
   BEGIN
    dummy:=(a_node(i).NBOUT*MSERVICE(a_node(i))+
          d_node(i).NBOUT*MSERVICE(d_node(i)))/
         (a_node(i).NBOUT+d_node(i).NBOUT)/sl_time;
    PRINT("Station ",i," : ",dummy);
   END;
 PRINT(" ");
 PRINT("Access unit service ave  bus A only (in slot unit)");
 PRINT(" ");
 FOR i:=1 STEP 1 UNTIL au DO
   BEGIN
    dummy:=MSERVICE(a_node(i))/sl_time;
    PRINT("Station ",i," : ",dummy);
   END;
 PRINT(" ");
 PRINT("Access unit service ave  bus B only (in slot unit)");
 FOR i:=1 STEP 1 UNTIL au DO
   BEGIN
    dummy:=MSERVICE(d_node(i))/sl_time;
    PRINT("Station ",i," : ",dummy);
   END;
 PRINT(" ");

 PRINT("Access unit response delay ave bus A only (in slot unit)");
 PRINT(" ");
&*****************************************
& Measurement of the mean response time for each station on the buses (the
& result must be soustracted by one slot time for getting the mean access delay
&*****************************************
 FOR i:=1 STEP 1 UNTIL au DO
   BEGIN
    dummy:= MRESPONSE(a_node(i))/sl_time;
    PRINT("Station ",i," : ",dummy);
```

```
 END;

PRINT("Access unit response delay bus B only (in slot unit)");
PRINT(" ");
FOR i:=1 STEP 1 UNTIL au DO
 BEGIN
  dummy:= MRESPONSE(d_node(i))/sl_time;
  PRINT("Station ",i," : ",dummy);
 END;

PRINT("Access unit response delay ave ,both buses A+B (in slot unit)");
PRINT(" ");
FOR i:=1 STEP 1 UNTIL au DO
 BEGIN
  dummy:=(a_node(i).NBOUT*MRESPONSE(a_node(i))+
        d_node(i).NBOUT*MRESPONSE(d_node(i)))/
       (a_node(i).NBOUT+d_node(i).NBOUT)/sl_time;
  PRINT("Station ",i," : ",dummy);
 END;



FOR i:=2 STEP 1 UNTIL ind DO
mdelay(i-1):=mdelay(i);
tdelay:=0;
t_cus:=0;
&*****************************************
& Measurement of the mean response time for all the station on the buses (the
& result must be soustracted by one slot time in order to get the mean
& access delay).
&*****************************************
FOR i:=1 STEP 1 UNTIL au DO
  BEGIN
   tdelay:=tdelay+a_node(i).NBOUT*MRESPONSE(a_node(i))+
              d_node(i).NBOUT*MRESPONSE(d_node(i));
   t_cus:=t_cus+a_node(i).NBOUT+d_node(i).NBOUT;
  END;
d_util := REALINT(qa_use)/REALINT(qa_tot);
mdelay(ind):=tdelay/t_cus/sl_time;  & mean access delay
PRINT(" ");
PRINT("TIME = ",TIME);
PRINT("Offered load              : ",offer);
dummy:=p_delay*bus_len*speed/sl_len;
PRINT("Parameter a               : ",dummy);
PRINT("Global delay (in slot unit)  : ",mdelay(ind));
PRINT("Bandwidth BAl 0=off,1=on =",bwb_flag);
PRINT("Bandwidth Balance Modulus: varies");
PRINT("DQDB ASYNC utilisation :  ",d_util);
PRINT("DQDB ASYNC slots generated :  ",qa_tot);
PRINT("DQDB ASYNC slots used :  ",qa_use);
PRINT("DQDB  slots generated :  ",req_tot);
PRINT("DQDB request issued  :  ",req_use);
PRINT(" ");
END;
SEND

/CONTROL/ TMAX = 5.0;
        PERIOD = 5.0;    & Sample period
        TEST = BEGIN     & Stop the simulation if the results converge
               OUTPUT;
```

```
        $print_out
        END;

    ENTRY = BEGIN
        PRINT("INPUT PARAMETER ----> DQDB");
        PRINTF("(1H,'Number of frames        : ',F3.0)",nb_fr);
        PRINTF("(1H,'Number of slots/frame   : ',F3.0)",nb_sl);
        PRINTF("(1H,'Number of NA slots/frame : ',F3.0)",nb_na);
        PRINTF("(1H,'Number of QA slots/frame : ',F3.0)",nb_qa);
        PRINTF("(1H,'Number of stations      : ',F3.0)",au);
        PRINTF("(1H,'Ring length (km)        : ',F5.1)",bus_len);
        PRINT("Frame period (sec)      :",fr_time);
        PRINT("Bit rate (bits/sec)     :",speed);
        PRINT("Slot time (sec)         :",sl_time);
        PRINT("Node to node delay      :",n_to_n);
        PRINT("Network capacity(mes/s) :",b*nb_qa/fr_time);
        PRINT("Bandwidth BAl 0=off,1=on :",bwb_flag);
        PRINT("Bandwidth Balance Modulus: varies");
       PRINT("Total arrival rate(mes/s):",au*load);
        PRINT("Parameter a             :",offer);
       END;
    EXIT = $print_out

/EXEC/ BEGIN
    FOR i := 1 STEP 1 UNTIL ind DO
    mdelay(i):=1E-18;
    FOR i := 1 STEP 1 UNTIL au DO
     BEGIN
      bwb_mod(1,i) := 8;
      bwb_mod(2,i) := 8;
      erasure(1,i) := 0;
      erasure(2,i) := 0;
      a_switch(i).in:=i;
      d_switch(i).in:=i;
      a_switch(i).ib:=1;
      d_switch(i).ib:=2;
      a_switch(i).icb:=2;
      d_switch(i).icb:=1;
      a_switch(i).inext:=i+1;
      d_switch(i).inext:=i-1;
      a_node(i).in:=i;
      d_node(i).in:=i;
      a_node(i).ib:=1;
      d_node(i).ib:=2;
      station(i).in:=i;
     END;
      bwb_mod(1,1) := 8;
      bwb_mod(1,2) := 8;
      bwb_mod(1,4) := 8;
      bwb_mod(1,5) := 8;
      bwb_mod(1,7) := 8;
      bwb_mod(1,8) := 8;
      bwb_mod(2,2) := 8;
      bwb_mod(2,3) := 8;
      bwb_mod(2,5) := 8;
      bwb_mod(2,6) := 8;
      bwb_mod(2,8) := 8;
      bwb_mod(2,9) := 8;
    FOR i := 1 STEP 1 UNTIL au DO
     BEGIN
```

```
PRINT("Bandwidth Balance Modulus: bwb_mod(1,",i,") =",bwb_mod(1,i));
PRINT("Bandwidth Balance Modulus: bwb_mod(2,",i,") =",bwb_mod(2,i));
        END;
  PRINT(" ");
  PRINT(" ");
       FOR i := 1 STEP 1 UNTIL au DO
        BEGIN
PRINT("ERASURE NODE FLAG VALUE: erasure(1,",i,") =",erasure(1,i));
PRINT("ERASURE NODE FLAG VALUE: erasure(2,",i,") =",erasure(2,i));
        END;
        SIMUL;
      END;
/END/
```

B.2 Simulation Program Used for Chapter Four

```
/CONTROL/ OPTION = NSOURCE;
&*****************************************
& DQDB INTERCONNECTING three 16 Mbps LANS
&*****************************************
&*****************************************
& In this program we simulates communications between nodes situated on three
& differents token rings (IEEE 802.5) interconnected by DQDB (IEEE 802.6)
& The type 2 of the IEEE 802.2 Logical Link Control (LLC) standard is used as
& end-to-end protocol between the end stations.
&*****************************************
/CONTROL/ NMAX = 300;
        CLASS = ALL QUEUE;
&       ACCURACY = ALL QUEUE;
&*****************************************
& Declaration of variables
&*****************************************
/DECLARE/
INTEGER     r = 3,                  &3 number of token ring
            st = 6,                 & number of stations/ring
            nod = st+1,             & number of nodes/ring
            b = 2,                  & number of buses
            au = 3,                 &3 number of access units on DQDB
            pl = au+1,              & number of slot places
            nb_qa = 6,              & number of non iso slots per frame
            nb_na = 6,              & number of iso slots/frame
            nb_sl = nb_na+nb_qa,    & number of slots/frame
            nb_fr = 20,             & ## frames
            nb_fr_2 = 40,           & twice frame size for counter
            mod = 128,              & modulus size
            index = mod*2,          & index range
            bwb_ct(b,au),           & bandwidth balancing counter
            bwb_flag = 0,           & bwb on/off flag
            bwb_mod = 1,            & bandwidth balanceing modulus
            window = 1,             & window size used by by LLC
            byte = 8,               & 1 byte --> 8 bits
            count=3,                & number of values within 1 %
            range = 15,             & for I-frame sequence
            top = 4,                & for backpressure control
            i,j,k,                  & looped variables
            llc_ctr = 4,            & LLC PDU control bits
            info = 1024,            & length for MAX DQDB MAC PDU info
            i_llc = info+llc_ctr,   & length for MAX DQDB MAC PDU info
            dq_hdr = 24,            & DQDB MAC PDU header
            dq_tr = 4,              & DQDB MAC PDU trailer
            tk_hdr = 15,            & TR frame header
            tk_tr = 6,              & TR frame trailer
            s_frame = tk_hdr+tk_tr+llc_ctr, & s-frame length in TR
            i_frame = tk_hdr+i_llc+tk_tr,   & length for MAX TR frame
            payload= 44,            & *** payload segment length
            sl_len=53,              & ***slot length (bytes)
            nb_seg = 25,            & nb of cells for DQDB MAC PDU
            ack_fr(r,st),           & next expected frame by dest
            all(r,st,index),        & indicates all segment received
            exp_sg(r,st,index),     & expected segment
            ask(r,nod),             & has a packet queued
            counter(r),             & For counting the received DM PDUs
            recov(r,st),            & timer recovery condition status
            dum(r,st),              & dummy variable
            full(r,st,index),       & busy space in buffer
```

```
            first(r,st,mod),      & indicated first I=-frame trans.
            l_rn(r,st),           & last received I-frame..N(R)
            l_sn(r,st),           & higher last sent I-frame.. S(R)
            pos(r,st,mod),        & position in the window
            resent(r,st),         & indicated a resent situation
            seq(r,st,mod),        & I-frame position in rec. window
            s_rej(r,st),          & reject mode status
            vr(r,st),             & receive state variable V(R)
            vs(r,st),             & send state variable V(S)
            cd(b,au),             & initial of counters (REQ,CD)
            cd_ct(b,au),          & countdown counter
            req_ct(b,au),         & REQ counter
            rqm(b,au);            & REQ bit queue on contra bus

REAL        buffer = 8*i_frame,   & buffer size
&*****************************************
            ar_rate = 75,         & arrival rate for each workstation
&*****************************************
            arrival=1/ar_rate,    & inter-arrival time
            dusty = 5E-6,         & token passing delay
&*****************************************
            tok_spd = 16E6,       & token ring bit rate
&*****************************************
            thru(st),             & Througput value
            del(st),              & Delay value
            t_del = 75E-6,        & rec out of seq and nothing
            t_first = 250E-6,     & transmission time (first time)
            t_f_bit = 125E-6,     & rec F-bit and retrans I-frame
            t_hand = 125E-6,      & handle timer and send P-bit
            t_llc = 25E-6,        & dummy processing time
            t_p_bit = 125E-6,     & rec P-bit and retrans F-bit
            t_rej = 125E-6,       & out of seq and send REJ-frame
            t_rr = 300E-6,        & rec I-frame and send RR-frame
            t_slide = 100E-6,     & rec RR-frame and del I-frame
            t_timer = 250E-3,     & timeout period
            t_conv = 150E-6,      & TR frame in DQDB MAC PDU
            t_rec = 20E-6,        & proc. for decpsulating QA slots
            t_sgm = 31E-6,        & time to format QA slots
            offer=ar_rate*r*st*info*byte,  & offer load
            bus_len=25,           & bus length in km
            fr_time=125E-6,       & frame period time in second
            latency=180E-9,       & station latency in seconds
            p_delay=5.085E-6,     & propagation delay per km
            st_to_st=bus_len/au,  & distance between two stations
            n_to_n=p_delay*st_to_st,  & node to node propagation delay
            fr_len=nb_sl*sl_len*byte,  & frame length (bits)
            dq_spd=fr_len/fr_time,     & DQDB speed in bits/sec
            sl_time=sl_len*byte/dq_spd,  & slot length in second
            t_prog=sl_time/2+sl_time+n_to_n+latency, &time in link
            sl_gen=fr_time/nb_sl;      & slot generation period
FLAG        fl(r,nod),
            pbit(r,st),       & Indicates if the P-bit timer is running
            poll(r),          &
            fire(r,st),dump(r,st),flag(b,pl),
            req_fl(b,au),send(b,au);  & to indicate REQ bit trans
CLASS INTEGER  iring,icl,icr,ida,        &3 indicate classes
            iclop,            & indicate remote class
            length,           & packet length in bits
            size;             & packet length in bytes
```

```
CLASS        token(r),              & token
             dm1(st),dm2(st),dm3(st),  &3 DM PDUs
             i_f1(st),i_f2(st),i_f3(st),&3 I-frames
             s_f1(st),s_f2(st),s_f3(st),&3 S-frames
             slot;                  & QA and NA slots
QUEUE INTEGER  active,              & token ring status
             ir,is,iso,             & ring,station number
             ind,                   & index
             init,                  & initialization
             mac,                   & position of token
             num,                   & send sequence number
             ib,                    & bus identifibation
             icb,                   & contra bus
             in,                    & station number
             inext,                 & next station on the bus
             x,y,w,z;               & counter variable
QUEUE REAL   load,tdelay;           & total delay
QUEUE        stat1(st),stat2(st),stat3(st),&3 stations
             hlay1(st),hlay2(st),hlay3(st),&3 high layers
             wind1(st),wind2(st),wind3(st),&3 windows
             buff1(st),buff2(st),buff3(st),&3 buffers
             llc1(st),llc2(st),llc3(st),  &3 LLC
             sink1(st),sink2(st),sink3(st),&3 sinks
             tim1(st),tim2(st),tim3(st),  &3 timer
             tok_dq(r),dq_tok(r),    & gateway
             mac1(nod),mac2(nod),mac3(nod),&3 MAC
             ring(r),               & token rings
             head_a,head_b,         & head and end point
             a_switch(au),d_switch(au), & ascending and descending switches
             link,                  & link transferring DQDB slots
             a_node(au),d_node(au);   & node which queues arrival packets
CUSTOMER INTEGER p,f,rn,sn,         & receive/send sequence number
             sup,sg,                & sup=0-->RR,sup=2-->REJ
             next,                  & next station on DQDB
             req,                   & req bit
             unic,                  & variable for identifying customers
             bus,                   & destination bus
             busy,                  & busy bit
             type;                  & type of slot
CUSTOMER REAL  start;               & start measure for a given packet
REF CUSTOMER   I(r,st,index),       & I-frame
             S(r,st,index),         & S-frame
             T(r,st),               & Used to restart the timer
             FR(r,st,mod),          & Initial data unit
             RR(r,st,index),        & RR-frame sent by the P bit timer
             DM(r,st,index,nb_seg),   & DM PDUs generated in the gateways
             TOK(r,nod),            & Token
             SL(nb_fr_2,nb_sl);     & QA slot in a frame


&*****************************************
& Application layer generating data units
&*****************************************
& This station generates fixed-length data units at time interval exponentially
& distributed. Then the generated data units are sent to the next layer
&*****************************************
SMACRO stat (hlay,i_f)
  TYPE = SOURCE;
```

```
 SERVICE = BEGIN
         EXP(arrival);              & Data unit generation
         FR(ir,is,num):=NEW(CUSTOMER);
         FR(ir,is,num).sn:=num;
         IF hlay(is).NB > top-1     & Is the "hlay" station buffer full
         THEN
           BEGIN
            RESET(fire(ir,is));
            WAIT(fire(ir,is)); & Wait until one "hlay" buffer partition is
           END;               & released
         TRANSIT(FR(ir,is,num),hlay(is),i_f(is));
         num:=num+1;
         IF num > mod
         THEN num:=1;
         TRANSIT(OUT);
        END;
SEND

&*****************************************
& Application layer generating data unit eventually sent on token ring # 1
&*****************************************
/STATION/ NAME = stat1(1 STEP 1 UNTIL st);
        Sstat (hlay1,i_f1)

&*****************************************
& Application layer generating data unit eventually sent on token ring # 2
&*****************************************
/STATION/ NAME = stat2(1 STEP 1 UNTIL st);
        Sstat (hlay2,i_f2)
&3*****************************************
&3 Application layer generating data unit eventually sent on token ring # 3
&3*****************************************
/STATION/ NAME = stat3(1 STEP 1 UNTIL st);
        Sstat (hlay3,i_f3)


&*****************************************
& High layer
&*****************************************
& This station is throttling the generation of data unit generated by the
& corresponding application layer entitylinggenerating data units.
&*****************************************
SMACRO highlayer (wind,llc,i_f)
  SERVICE = BEGIN
         P(wind(is));      & The data unit is inserted in the LLC window
         IF init=1
         THEN
           BEGIN           & Intialization of the station
            SET(pbit(ir,is));
            init:=0;
           END;
         WAIT(pbit(ir,is)); & Wait if the LLC is in P-timer recovery cond.
         start:=TIME;      & start to measure the end-to-end delay
         SET(fire(ir,is)); & one buffer partition is released
         TRANSIT(llc(is),i_f(icl)); & Data unit is transmitted to the LLC
        END;
SEND

&*****************************************
```

```
& High layer serving data units eventually sent on token ring # 1
&*****************************************
/STATION/ NAME = hlay1(1 STEP 1 UNTIL st);
        Shighlayer (wind1,llc1,i_f1)

&*****************************************
& High layer serving data units eventually sent on token ring # 2
&*****************************************
/STATION/ NAME = hlay2(1 STEP 1 UNTIL st);
        Shighlayer (wind2,llc2,i_f2)

&3****************************************
&3 High layer serving data units eventually sent on token ring # 3
&3****************************************
/STATION/ NAME = hlay3(1 STEP 1 UNTIL st);
        Shighlayer (wind3,llc3,i_f3)


&*****************************************
& Buffer corresponding to the LLC window
&*****************************************
& This station contained the outstanding I-frames. The flag "dump" is used to
& that command the sliding of the LLC window as well as the retransmission of
& of I-frames.
&*****************************************
SMACRO buffer (wind,tank,llc,i_f)
  SERVICE = BEGIN
&       PRINT("IN side buffer----> ");
          WAIT(dump(ir,is));
&        PRINT("IN side buffer----> just past wait dunp ",resent(ir,is));
          IF resent(ir,is)=0   & Has a N(R) been received ?
           THEN                & When N(R) simply acknowledges I-frame
            BEGIN
&       PRINT("IN side buffer----> resent(ir,is) = 0 ",resent(ir,is));
             IF pos(ir,is,sn)=pos(ir,is,ack_fr(ir,is))-1 & Last frames to
             THEN RESET(dump(ir,is));                    & be removed ?
             V(wind(is));     & One I-frame is removed from the window
             TRANSIT(OUT);
             END
           ELSE                & I-frames can have to be retransmitted
            BEGIN              & (REJ-frame or RR-frame with P=1)
&        PRINT("IN side buffer----> resent(ir,is) = 2",resent(ir,is));
             IF pos(ir,is,sn) < pos(ir,is,ack_fr(ir,is))
             THEN              & Remove I-frames from the window
              BEGIN
 &       PRINT("IN side buffer----> resent(ir,is) = 3 ",resent(ir,is));
               IF sn=resent(ir,is)   & Last I-frame to be removed ?
               THEN RESET(dump(ir,is));
               V(wind(is));  & I-frames removed
               TRANSIT(OUT);
              END
             ELSE              & Retransmission of I-frames
              BEGIN
&        PRINT("IN side buffer----> resent(ir,is) = 4",resent(ir,is));
               IF sn=resent(ir,is)
               THEN RESET(dump(ir,is));
               TRANSIT(llc(is)); & I-frames are sent to the LLC process
              END;
            END;
```

```
        END;
SEND

&*****************************************
& Buffer serving I-frames eventually sent on token ring #1
&*****************************************
/STATION/ NAME = buff1(1 STEP 1 UNTIL st);
        Sbuffer (wind1,tank1,llc1,i_f1)

&*****************************************
& Buffer serving I-frames eventually sent on token ring #2
&*****************************************
/STATION/ NAME = buff2(1 STEP 1 UNTIL st);
        Sbuffer (wind2,tank2,llc2,i_f2)

&*****************************************
& Buffer serving I-frames eventually sent on token ring #3
&*****************************************
/STATION/ NAME = buff3(1 STEP 1 UNTIL st);
        Sbuffer (wind3,tank3,llc3,i_f3)

&*****************************************
& LLC window serving I-frames eventually sent on token ring #1
&*****************************************
/STATION/ NAME = wind1(1 STEP 1 UNTIL st);
        TYPE = RESOURCE,MULTIPLE(window);

&*****************************************
& LLC window serving I-frames eventually sent on token ring #2
&*****************************************
/STATION/ NAME = wind2(1 STEP 1 UNTIL st);
        TYPE = RESOURCE,MULTIPLE(window);

&3*****************************************
&3 LLC window serving I-frames eventually sent on token ring #3
&3*****************************************
/STATION/ NAME = wind3(1 STEP 1 UNTIL st);
        TYPE = RESOURCE,MULTIPLE(window);



&*****************************************
& LLC protocol
&*****************************************
& This station simulates the LLC protocol. Three classes of customers are
& served.The I-frames transmitted by the station, and the I- and S-frames
& received from the remote LLC entity.
&*****************************************
SMACRO llc (i_loc,i_rem_1,i_rem_2,s_loc,s_rem_1,s_rem_2,llcc,macc,tim,buff,sink)
&*****************************************
& Service of the I-frames sent by the station
&*****************************************
SERVICE(i_loc)= BEGIN
&       PRINT("IN side llc sevice i_loc---> ");
            IF sn<>vs(ir,is)   & Is the I-frame the one to be transmitted ?
            THEN
              BEGIN
               CST(t_llc);
               TRANSIT(llcc(is));
```

```
 END;
IF pos(ir,is,sn)>pos(ir,is,l_sn(ir,is))
THEN l_sn(ir,is):=sn;
ind:=ind+1;
IF ind>index
THEN ind:=1;
I(ir,is,ind):=NEW(CUSTOMER); & Generate a copy of the I-frame
I(ir,is,ind).sn:=sn;
I(ir,is,ind).rn:=vr(ir,is);
I(ir,is,ind).unic:=ind;
I(ir,is,ind).start:=start;
FOR x:=1 STEP 1 UNTIL 2*range DO & Refresh of the I-frame
 BEGIN                          & positions in the window
  IF (l_rn(ir,is)+x-range)>0
  THEN
   BEGIN
    IF (l_rn(ir,is)+x-range)<=mod
    THEN pos(ir,is,l_rn(ir,is)+x-range):=x-range+1
    ELSE pos(ir,is,l_rn(ir,is)+x-range-mod):=x-range+1;
   END
  ELSE pos(ir,is,l_rn(ir,is)+x-range+mod):=x-range+1;
 END;
IF (sn-2*range) > 0              & Reset variable "first"
THEN first(ir,is,sn-2*range):=0  & for a subsequent I-frame
ELSE first(ir,is,sn-2*range+mod):=0;
IF first(ir,is,sn)=0   & Is it the first trans. of the I-frame
THEN
 BEGIN
  CST(t_first);
  first(ir,is,sn):=1;
 END
ELSE CST(t_llc);
TRANSIT(I(ir,is,ind),macc(is),i_loc(is));
IF tim(is).NB = 0      & Is the Ack. timer not running ?
THEN                   & If yes, Ack. timer is started
 BEGIN
  T(ir,is):=NEW(CUSTOMER);
  TRANSIT(T(ir,is),tim(is));
 END;
vs(ir,is):=vs(ir,is)+1; & Incrementation of the V(S)
IF vs(ir,is)>mod
THEN vs(ir,is):=1;
TRANSIT(buff(is));
END;

&****************************************
& Service of the I-frames sent by the remote LLC entities
&****************************************
SERVICE(i_rem_1,i_rem_2)=BEGIN
 &     PRINT("IN side llc sevice i_rem---> ");
       IF pos(ir,is,rn)>pos(ir,is,l_rn(ir,is))  & Check if  R(N) is
       THEN                                     & higher than the "last
        BEGIN                                   & received R(N)"
         l_rn(ir,is):=rn;
         IF recov(ir,is)=0  & Is LLC not in timer recovery condtion ?
         THEN
          BEGIN
           MOVE(tim(is),OUT);   & Clear the Ack. timer
           IF pos(ir,is,l_sn(ir,is))>pos(ir,is,rn)
```

```
            THEN              & Restart the timer if I-frames
              BEGIN           & are still outstanding
               T(ir,is):=NEW(CUSTOMER);
               TRANSIT(T(ir,is),tim(is));
              END;
             ack_fr(ir,is):=rn;   & Next I-frame to be transmitted
             resent(ir,is):=0;    & No I-frame to be retransmitted
             SET(dump(ir,is));    & The "buffer" station can slide the
             CST(t_slide);        & window
            END;
           FOR x:=1 STEP 1 UNTIL 2*range DO  & Refresh of the I-frame
            BEGIN                          & positions in the window
             IF (rn+x-range)>0
             THEN
              BEGIN
               IF (rn+x-range)<=mod
               THEN pos(ir,is,rn+x-range):=x-range+1
               ELSE pos(ir,is,rn+x-range-mod):=x-range+1;
              END
             ELSE pos(ir,is,rn+x-range+mod):=x-range+1;
            END;
          END;
         IF ((sn=vr(ir,is)) OR ((s_rej(ir,is)=0) AND
           (seq(ir,is,sn)>seq(ir,is,vr(ir,is)))))
              & Process when an expected I-frame is received (or if
              & the the LLC is not in Reject mode) and when an
              & I-frame with a S(N) greater than V(R)
         THEN
          BEGIN
           ind:=ind+1;
           IF ind>index
           THEN ind:=1;
           S(ir,is,ind):=NEW(CUSTOMER);  & Generate a S-frame
           S(ir,is,ind).p:=0;
           S(ir,is,ind).f:=0;
           S(ir,is,ind).sn:=mod+1;
           IF sn=vr(ir,is)   & Has received I-frame been expected ?
           THEN
            BEGIN
             FOR x:=1 STEP 1 UNTIL 2*range DO & Refresh of the I-frame
              BEGIN                          & positions in the window
               IF (sn+x-range)>0
               THEN
                BEGIN
                 IF (sn+x-range)<=mod
                 THEN seq(ir,is,sn+x-range):=x-range+1
                 ELSE seq(ir,is,sn+x-range-mod):=x-range+1;
                END
               ELSE seq(ir,is,sn+x-range+mod):=x-range+1;
              END;
             s_rej(ir,is):=0;   & Reject mode cleared
             vr(ir,is):=vr(ir,is)+1;  & V(R) incremented
             IF vr(ir,is)>mod
             THEN vr(ir,is):=1;
             S(ir,is,ind).sup:=0;
             S(ir,is,ind).rn:=vr(ir,is); & R(N) in S-frame set to V(R)
             CST(t_rr);
&      PRINT("s_loc A IR =",ir,"IS = ",is,"ICR=",icl);
TRANSIT(S(ir,is,ind),macc(is),s_loc(is)); & S-frame-->MAC
```

```
                TRANSIT(sink(is)); & I-frame info moved to higher layers
                END
              ELSE  & Has an unexpected been received for the first time ?
                BEGIN
                 CST(t_rej);
                 s_rej(ir,is):=1;  &LLC moves to the reject mode
                 S(ir,is,ind).sup:=2;        & REJ control
                 S(ir,is,ind).rn:=vr(ir,is); & R(N) in S-frame set to V(R)
&       PRINT("s_loc B IR =",ir,"IS = ",is,"ICR=",icr);
                 TRANSIT(S(ir,is,ind).macc(is).s_loc(is)); & S-frame-->MAC
                END;
              END
            ELSE CST(t_del);
            TRANSIT(OUT);
           END;
&*****************************************
& Service of the S-frames sent by the remote LLC entities
&*****************************************
SERVICE(s_rem_1,s_rem_2)= BEGIN
&     PRINT("IN side llc sevice s_rem----> ");
            IF f=1    & Does the S-frame has its F-bit set to 1 ?
            THEN
             BEGIN
              MOVE(tim(is),OUT);  & Cleared the timer
              recov(ir,is):=0;    & Cleared the P-bit timer recovery cond.
              SET(pbit(ir,is));   & IDEM
              IF pos(ir,is,rn)>pos(ir,is,l_rn(ir,is)) & Is R(N) > last
              THEN l_rn(ir,is):=rn;                   & higher R(N) rece.
              vs(ir,is):=l_rn(ir,is);   & V(S) set to last higher R(N) rec
              ack_fr(ir,is):=l_rn(ir,is); & Next I-frame to be retrans.
              resent(ir,is):=l_sn(ir,is); & Last I-frame to be retrans.
              FOR x:=1 STEP 1 UNTIL 2*range DO   & Refresh of the I-frame
               BEGIN                             & positions in the window
                IF (l_rn(ir,is)+x-range)>0
                THEN
                 BEGIN
                  IF (l_rn(ir,is)+x-range)<=mod
                  THEN pos(ir,is,l_rn(ir,is)+x-range):=x-range+1
                  ELSE pos(ir,is,l_rn(ir,is)+x-range-mod):=x-range+1;
                 END
                ELSE pos(ir,is,l_rn(ir,is)+x-range+mod):=x-range+1;
               END;
              SET(dump(ir,is));
              CST(t_f_bit);
             END
            ELSE
             BEGIN
              IF (pos(ir,is,rn)>=pos(ir,is,l_rn(ir,is))) AND (sup=2)
              THEN  & The S-frame is a REJ-frame
               BEGIN
                l_rn(ir,is):=rn;
                IF recov(ir,is)=0
                THEN
                 BEGIN
                  MOVE(tim(is),OUT);  & Timer is cleared
                  vs(ir,is):=rn;      & V(S) set to R(N)
                  ack_fr(ir,is):=rn;  & Next I-frame to be retransmitted
                  resent(ir,is):=l_sn(ir,is);  & Last I-frame to be retran.
                  SET(dump(ir,is));   & "Buffer" station can slide the
```

```
                CST(t_f_bit);       & window
               END;
              END
             ELSE & The S-frame is a RR-frame
              BEGIN
              IF pos(ir,is,rn)>pos(ir,is,l_rn(ir,is))
              THEN   & R(N) > the last higher R(N) received
               BEGIN
               l_rn(ir,is):=rn;
               IF recov(ir,is)=0  & Is the LLC being in P-timer rec.
               THEN
                BEGIN
                MOVE(tim(is),OUT); & The timer is cleared
                IF pos(ir,is,l_sn(ir,is))>pos(ir,is,rn)
                THEN & I-frames are still outstanding
                 BEGIN
                 T(ir,is):=NEW(CUSTOMER);   & Restart the timer
                 TRANSIT(T(ir,is),tim(is));
                 END;
                ack_fr(ir,is):=rn; & Next I-frame to be transmitted
                resent(ir,is):=0;  & No I-frame to be retransmitted
                SET(dump(ir,is)); & "Buffer" station can slide the
                CST(t_slide);     & window
               END;
              END;
             IF p=1
             THEN   & The RR-frame has its P-bit set to 1
              BEGIN
              ind:=ind+1;
              IF ind>index
              THEN ind:=1;
              S(ir,is,ind):=NEW(CUSTOMER);
              S(ir,is,ind).sup:=0;
              S(ir,is,ind).p:=0;
              S(ir,is,ind).f:=1;  & Set the F-bit to 1
              S(ir,is,ind).sn:=mod+1;
              S(ir,is,ind).rn:=vr(ir,is);
              CST(t_p_bit);
&     PRINT("s_loc C IR =",ir,"IS = ",is,"ICR=",icr);
               TRANSIT(S(ir,is,ind),macc(is),s_loc(is));
              END;
             END;
            FOR x:=1 STEP 1 UNTIL 2*range DO  & Refresh of the I-frame
             BEGIN                            & positions in the window
              IF (l_rn(ir,is)+x-range)>0
              THEN
               BEGIN
                IF (l_rn(ir,is)+x-range)<=mod
                THEN pos(ir,is,l_rn(ir,is)+x-range):=x-range+1
                ELSE pos(ir,is,l_rn(ir,is)+x-range-mod):=x-range+1;
               END
              ELSE pos(ir,is,l_rn(ir,is)+x-range+mod):=x-range+1;
             END;
            END;
           TRANSIT(OUT);
          END;
SEND

&*****************************************
```

```
& LLC serving I-frames eventually sent on token ring #1
&*****************************************
/STATION/ NAME = llc1(1 STEP 1 UNTIL st);
        Sllc (i_f1,i_f2,i_f3,s_f1,s_f2,s_f3,llc1,mac1,tim1,buff1,sink1)

&*****************************************
& LLC serving I-frames eventually sent on token ring #2
&*****************************************
/STATION/ NAME = llc2(1 STEP 1 UNTIL st);
        Sllc (i_f2,i_f1,i_f3,s_f2,s_f1,s_f3,llc2,mac2,tim2,buff2,sink2)

&3****************************************
&3 LLC serving I-frames eventually sent on token ring #3
&3****************************************
/STATION/ NAME = llc3(1 STEP 1 UNTIL st);
        Sllc (i_f3,i_f1,i_f2,s_f3,s_f1,s_f2,llc3,mac3,tim3,buff3,sink3)


&*****************************************
& I-frames sink
&*****************************************
& This station has been used to measure the mean end-to-end for
& transferring I-frame, as well as the throughput of each token station
&*****************************************
SMACRO sink
  SERVICE = BEGIN
            tdelay:=tdelay+TIME-start;
            TRANSIT(OUT);
           END;
SEND

&*****************************************
& I-frame sink in stations situated on token ring #1
&*****************************************
/STATION/ NAME = sink1(1 STEP 1 UNTIL st);
        Ssink

&*****************************************
& I-frame sink in stations situated on token ring #2
&*****************************************
/STATION/ NAME = sink2(1 STEP 1 UNTIL st);
        Ssink

&3****************************************
&3 I-frame sink in stations situated on token ring #3
&3****************************************
/STATION/ NAME = sink3(1 STEP 1 UNTIL st);
        Ssink
&*****************************************
& LLC TIMERS
&*****************************************
& This station simulates two timers: the acknowledgement timer, and the P-bit
& timer
&*****************************************
SMACRO timer (macc,tim,s_f)
  SERVICE = BEGIN
            CST(t_timer);    & Timeout period
            recov(ir,is):=1; & LLC moves in P-bit timer recovery conditions
            CST(t_hand);     & (if the timer has expired)
```

```
          RESET(pbit(ir,is)); & Indicates that LLC moves in ...
          ind:=ind+1;
          IF ind>index
          THEN ind:=1;
          RR(ir,is,ind):=NEW(CUSTOMER);  & Generation of a RR-frame
          RR(ir,is,ind).sup:=0;
          RR(ir,is,ind).p:=1;            & With P-bit set to 1
          RR(ir,is,ind).f:=0;
          RR(ir,is,ind).sn:=mod+1;
          RR(ir,is,ind).rn:=vr(ir,is);   & Next expected I-frame
          TRANSIT(RR(ir,is,ind),macc(is),s_f(is)); & RR-frame --> MAC
          TRANSIT(tim(is));
         END;
SEND

&*****************************************
& LLC timers used by stations situated on token ring #1
&*****************************************
/STATION/ NAME = tim1(1 STEP 1 UNTIL st);
       Stimer (mac1,tim1,s_f1)

&*****************************************
& LLC timers used by stations situated on token ring #2
&*****************************************
/STATION/ NAME = tim2(1 STEP 1 UNTIL st);
       Stimer (mac2,tim2,s_f2)

&3*****************************************
&3 LLC timers used by stations situated on token ring #3
&3*****************************************
/STATION/ NAME = tim3(1 STEP 1 UNTIL st);
       Stimer (mac3,tim3,s_f3)

&*****************************************
& MAC layer serving end stations on token rings
&*****************************************
& This server works in commun with the "ring" station. Receiving a frame for
& transmission, this server waits for the token (flag).
&*****************************************
$MACRO maclayer(i_f,s_f,server)
      ask(ir,is):=1;  & Indicate to the "ring" server that its has a frame
      IF ring(ir).active=0                    & to transmit
      THEN    & Reinitialize the token ring
        BEGIN
         TOK(ir,is):=NEW(CUSTOMER);
         TRANSIT(TOK(ir,is),ring(ir));
        END;
      WAIT(fl(ir,is));     & Wait for the token
      RESET(fl(ir,is));
      CST(length/tok_spd); & Transmit the frame
      ask(ir,is):=0;
      SET(poll(ir));       & Return the token to the "ring" server
      load:=sl_len*CUSTNB(server(ir));  & measure the number of bytes
      FOR x:=1 STEP 1 UNTIL st DO       & which are used in the buffer of
      load:=load+s_frame*CUSTNB(tok_dq(ir),s_f(x))+ & situated in the
           i_frame*CUSTNB(tok_dq(ir),i_f(x));     & entrance gateway
      IF load <= buffer-size  & Is the buffer can receive the present frame ?
      THEN BEGIN
&PRINT( "mac",ir,"() icl=",icl,"is=",is);
```

```
TRANSIT(tok_dq(ir)); & If yes, ---> frame transmission
END
       ELSE BEGIN
&  PRINT(" inside Mac ring",ir, "stat",is," deleteing a frame");
TRANSIT(OUT);       & If not, the frame is discarded
            END;
SEND

&*****************************************
& MAC layer serving end stations on token ring #1
&*****************************************
/STATION/ NAME = mac1(1 STEP 1 UNTIL st);
SERVICE=BEGIN
        Smaclayer(i_f1,s_f1,a_node)
      END;

&*****************************************
& MAC layer serving end stations on token ring #2
&*****************************************
/STATION/ NAME = mac2(1 STEP 1 UNTIL st);
SERVICE=BEGIN
        IF (is < 4)
        THEN BEGIN
            Smaclayer(i_f2,s_f2,d_node)
         END
        ELSE BEGIN
            Smaclayer(i_f2,s_f2,a_node)
            END;
      END;

&3*****************************************
&3 MAC layer serving end stations on token ring #3
&3*****************************************
/STATION/ NAME = mac3(1 STEP 1 UNTIL st);
SERVICE=BEGIN
        Smaclayer(i_f3,s_f3,d_node)
      END;

*
& MAC layer serving gateways on token rings
&*****************************************
& This server works like the previous defined severs, but it does not have to
& verify if the destination station has enough room to receive the frame
&*****************************************
SMACRO gatemac(server)
  SERVICE = BEGIN
          ask(ir,nod):=1;
          IF ring(ir).active=0
          THEN
            BEGIN
             TOK(ir,is):=NEW(CUSTOMER);
             TRANSIT(TOK(ir,is),ring(ir));
            END;
          WAIT(fl(ir,nod));
          RESET(fl(ir,nod));
          CST(length/tok_spd);
          ask(ir,nod):=0;
          SET(poll(ir));
          TRANSIT(server(ida));
```

```
        END;
SEND

&*****************************************
& MAC layer serving the gateway situated on token ring #1
&*****************************************
/STATION/ NAME = mac1(nod);
            $gatemac(llc1)

&*****************************************
& MAC layer serving the gateway situated on token ring #2
&*****************************************
/STATION/ NAME = mac2(nod);
            $gatemac(llc2)

&3****************************************
&3 MAC layer serving the gateway situated on token ring #3
&3****************************************
/STATION/ NAME = mac3(nod);
            $gatemac(llc3)


&*****************************************
& Token ring monitor
&*****************************************
& This server passes the token to the stations which have frames queued. The
& token ring monitor verifies each station according to predetermined order
& (in fact, by following the order in which the end stations are defined).
&*****************************************
$MACRO tok_ring(macc)
 SERVICE = BEGIN
          active:=1;     & Token ring active
          y:=RINT(1,nod);
          WHILE mac >= 0 DO
           BEGIN
            x:=x+1;
            mac:=mac+1-INTREAL(mac/nod)*nod;
            IF ask(ir,mac)=1  & Verifies if station "m" has a frame queued
            THEN
             BEGIN
              x:=0;
              SET(fl(ir,mac));  & Pass the token to the station
              WAIT(poll(ir));   & Wait for the token
              RESET(poll(ir));
              CST(dusty);
             END;
&*****************************************
& The next few lines must be used when higher ring access priority and
& exhaustive service is allocated to the gateway MACs.
&*****************************************
            WHILE ask(ir,nod)=1 DO   & The token is returned to the gateway
             BEGIN
              x:=0;
              SET(fl(ir,nod));  & Pass the token to the gateway
              WAIT(poll(ir));
              RESET(poll(ir));
              CST(dusty);
             END;
&*****************************************
```

```
            IF x > nod+y
            THEN
              BEGIN
               x:=0;        & Use to kill the token (for saving
               active:=0;   & computation time)
               TRANSIT(OUT);
              END;
            END;
          END;
        TRANSIT = ring(ir);
        INIT(token) = 1;
SEND

&*****************************************
& Token ring monitor on ring #1
&*****************************************

/STATION/ NAME = ring(1);
        Stok_ring(mac1)

&*****************************************
& Token ring monitor on ring #2
&*****************************************
/STATION/ NAME = ring(2);
        Stok_ring(mac2)

&3*****************************************
&3Token ring monitor on ring #3
&3*****************************************
/STATION/ NAME = ring(3);
        Stok_ring(mac3)


&*****************************************
& Processor performing the convertion and segmentation of token ring frames
& in the gateways
&*****************************************
/STATION/ NAME = tok_dq(1 STEP 1 UNTIL r);
SERVICE(i_f1,i_f2,i_f3) = BEGIN
                  &************
                  & I-frame processing
                  &************
                  CST(t_conv);
                  all(ir,icl,unic):=0;
                  exp_sg(ir,icl,unic):=1;
                  FOR x:=1 STEP 1 UNTIL nb_seg-1 DO   & Nb of segm minus 1
                    BEGIN                             & to format
                     DM(ir,icl,unic,x):=NEW(CUSTOMER); & Segment generated
                     DM(ir,icl,unic,x).sn:=sn;
                     DM(ir,icl,unic,x).unic:=unic;
                     DM(ir,icl,unic,x).sg:=x;
                     CST(t_sgm);
                     IF (ir=1)
                     THEN TRANSIT(DM(ir,icl,unic,x),a_node(1),dm1(icl));
                     IF ((ir = 2) AND (icl < 4)) THEN
                     TRANSIT(DM(ir,icl,unic,x),d_node(2),dm2(icl));
                     IF ((ir = 2) AND (icl > 3)) THEN
                     TRANSIT(DM(ir,icl,unic,x),a_node(2),dm2(icl));
                     IF (ir=3)
```

```
                THEN TRANSIT(DM(ir,icl,unic,x),d_node(3),dm3(icl));
                END;
              CST(t_sgm);
              IF (ir=1)
              THEN TRANSIT(a_node(1),i_f1(icl));
              IF ((ir = 2) AND (icl < 4)) THEN
              TRANSIT(d_node(2),i_f2(icl));
              IF ((ir = 2) AND (icl > 3)) THEN
              TRANSIT(a_node(2),i_f2(icl));
              IF (ir=3)
              THEN TRANSIT(d_node(3),i_f3(icl));
             END;
SERVICE(s_f1,s_f2,s_f3) = BEGIN
              &************
              & S-frame processing
              &************
              CST(t_conv+t_sgm);
              IF (ir=1)
              THEN BEGIN
              &PRINT( "IN TOK_dQ",ir,"(1) icl=",icl);
              TRANSIT(a_node(1),s_f1(icl));
               END;
              IF ((ir = 2 )AND (icl < 4)) THEN
              TRANSIT(d_node(2),s_f2(icl));
              IF ((ir = 2 )AND (icl >3)) THEN
              TRANSIT(a_node(2),s_f2(icl));
              IF (ir=3)
              THEN TRANSIT(d_node(3),s_f3(icl));
              IF (ir=4)
              THEN
             END;

&*************************************
& Processor performing the reassembly of token ring frames in the gateways
&*************************************
/STATION/ NAME = dq_tok(1 STEP 1 UNTIL r);
SERVICE(dm1,dm2,dm3) = BEGIN          & BOM, and COM DM PDUs reception
               CST(t_rec);
               IF sg=exp_sg(iring,icl,unic) & Is received segme expected?
               THEN
                BEGIN
                 counter(ir):=exp_sg(iring,icl,unic); & For further use
                 exp_sg(iring,icl,unic):=exp_sg(iring,icl,unic)+1;
                 IF exp_sg(iring,icl,unic)=nb_seg-1 & all BOM AND COM
                 THEN all(iring,icl,unic):=1;      & have been received
                END
               ELSE counter(ir):=0;      & Rejection of DM PDU
               TRANSIT(OUT);
               END;
SERVICE(i_f1,i_f2,i_f3)=BEGIN          & EOM DM PDU reception
             CST(t_rec);
             IF all(iring,icl,unic)=1 & Are all previous segments received
             THEN
              BEGIN
               CST(t_conv);    & The token ring is reassemblied
               counter(ir):=0;
               IF iclop=1
               THEN TRANSIT(mac1(nod));
               IF iclop =2
```

```
                THEN TRANSIT(mac2(nod));
                IF iclop = 3
                THEN TRANSIT(mac3(nod));
               END
              ELSE TRANSIT(OUT);
             END;
SERVICE(s_f1,s_f2,s_f3)=BEGIN          & Reception of DM PDU containing a S-frame
             CST(t_sgm+t_conv);
               IF iclop=1
               THEN TRANSIT(mac1(nod));
               IF iclop =2
               THEN TRANSIT(mac2(nod));
               IF iclop = 3
               THEN TRANSIT(mac3(nod));
              END;
&*****************************************
& Head of bus station
&*****************************************
& This station generates a sequence of slots comprised in a frame period of 125
& micro second.There are respectively "nb_na" NA slots and "nb_qa" QA slots
& which are genrated in each frame period.
&*****************************************
/STATION/ NAME = head_a;
      SERVICE = BEGIN
             WHILE b=2 DO
              BEGIN
               FOR x:=1 STEP 1 UNTIL nb_fr DO & Frame generation
                BEGIN
                 FOR y:=1 STEP 1 UNTIL nb_na DO & NA slot generation
                  BEGIN
                   SL(x,y):=NEW(CUSTOMER);
                   SL(x,y).type:=1;    & TYPE bit set to 1
                   SL(x,y).busy:=0;    & Idem for BUSY bit
                   SL(x,y).req:=0;
                   SL(x,y).bus:=1;
                   SL(x,y).next:=1;
                   CST(sl_gen);
                   TRANSIT(SL(x,y),a_switch(1));
                  END;
                 FOR y:=nb_na+1 STEP 1 UNTIL nb_sl DO
                  BEGIN              & QA slot generation
                   SL(x,y):=NEW(CUSTOMER);
                   SL(x,y).type:=0;    & TYPE bit reset to 0
                   SL(x,y).busy:=0;    & BUSY bit reset to 0
                   SL(x,y).req:=0;
                   SL(x,y).bus:=1;
                   SL(x,y).next:=1;
                   CST(sl_gen);
                   TRANSIT(SL(x,y),a_switch(1));
                  END;
                END;
               END;
             END;
      TRANSIT = head_a;
      INIT(slot) = 1;
&*****************************************
&*****************************************
/STATION/ NAME = head_b;
      SERVICE = BEGIN
```

```
            WHILE b=2 DO
             BEGIN
              FOR w:=nb_fr + 1 STEP 1 UNTIL nb_fr + nb_fr DO & Frame generation
               BEGIN
                FOR z:=1 STEP 1 UNTIL nb_na DO & NA slot generation
                 BEGIN
                  SL(w,z):=NEW(CUSTOMER);
                  SL(w,z).type:=1;    & TYPE bit set to 1
                  SL(w,z).busy:=0;    & Idem for BUSY bit
                  SL(w,z).req:=0;
                  SL(w,z).bus:=2;
                  SL(w,z).next:=au;
                  CST(sl_gen);
                  TRANSIT(SL(w,z),d_switch(au));
                 END;
                FOR z:=nb_na+1 STEP 1 UNTIL nb_sl DO
                 BEGIN                & QA slot generation
                  SL(w,z):=NEW(CUSTOMER);
                  SL(w,z).type:=0;    & TYPE bit reset to 0
                  SL(w,z).busy:=0;    & BUSY bit reset to 0
                  SL(w,z).req:=0;
                  SL(w,z).bus:=2;
                  SL(w,z).next:=au;
                  CST(sl_gen);
                  TRANSIT(SL(w,z),d_switch(au));
                 END;
               END;
             END;
          END;
      TRANSIT = head_b;
      INIT(slot) = 1;


&*****************************************
& QA slot access control in each station situated along the buses
&*****************************************
& This station simulating the distributed queueing protocol controlling the
& the access to the QA slots. Slots(customers), coming from the previous
& station situated on the bus, are examined. In particular, their REQ, TYPE
& and BUSY bits are checked. From the contents of these bits, REQ and
& countdown counters may be modified.
&*****************************************
$MACRO switch (return)
SERVICE = BEGIN
      IF req=1
      THEN     & if REQ bit equal 1 then REQ counter on the access
       BEGIN   & queue sending segments on the other bus is incremented
        req_ct(icb,in):=req_ct(icb,in)+1;
       END
      ELSE                & REQ bit is equal to zero
       BEGIN
        IF rqm(icb,in)>=1    & Does the AQ have a REQ bit to send**
        THEN               & on the reverse bus ?
         BEGIN
          req:=1;          & Set the REQ bit
          rqm(icb,in):=rqm(icb,in)-1;
         END;
       END;
      IF type=0            & Is the slot is a QA slot ?
```

```
        THEN
         BEGIN
          IF busy=0          & Is the QA slot empty ?
          THEN
           BEGIN
            IF cd(ib,in)=1   & Is the AQ in countdown  state ?
            THEN
             BEGIN
              IF cd_ct(ib,in)=0  & Is the CD counter equal to zero ?
              THEN
               BEGIN
                IF bwb_ct(ib,in) = bwb_mod
                THEN
                  BEGIN
                  bwb_ct(ib,in) := 0;
                  END
                ELSE
                   BEGIN
                   IF bwb_flag = 1 THEN   & bwb on/off test
                   bwb_ct(ib,in) := bwb_ct(ib,in) + 1;
                   busy:=1;        & The QA slot is set BUSY
                   cd(ib,in):=0;   & The AQ is moved to the CD state
& PRINT("insie switch setting send ");
                   SET(send(ib,in));
                   RESET(send(ib,in));
                   END;
                END
             ELSE cd_ct(ib,in):=cd_ct(ib,in)-1; & CD counter decremented
            END
           ELSE
            BEGIN
             IF req_ct(ib,in)> 0 & Is the REQ counter equal to zero ?
             THEN req_ct(ib,in):= req_ct(ib,in)-1; & REQ counter is
            END;                                   & decremented
          END;
         END;
        IF inext > 0
        THEN
         BEGIN
          IF inext = pl
          THEN TRANSIT(OUT)   & Slot sent to the end of the bus
          ELSE
           BEGIN
            CST(latency);
            next:=inext;        & Slot sent to the station on the bus
            TRANSIT(link);
           END;
         END
        ELSE TRANSIT(OUT);
       END;
$END

&*****************************************
& Stations sending on bus A.
&*****************************************
/STATION/ NAME = a_switch(1 STEP 1 UNTIL au);
       $switch (a_switch)
&*****************************************
& Stations sending on bus B.
```

```
&*****************************************
/STATION/ NAME = d_switch(1 STEP 1 UNTIL au);
        Sswitch (d_switch)



&*****************************************
& This macro serves to measure the memory partitions of the buffer used by the
& exit gateway. From the result the segment which is served can be rejected.
&*****************************************
SMACRO trans (macc,i_f,s_f)
 full(iclop,icl,sn+sg):=payload*counter(iclop)+sl_len*CUSTNB(dq_tok(iclop));
 FOR x:=1 STEP 1 UNTIL st DO
 full(iclop,icl,sn+sg):=full(iclop,icl,sn+sg)+
                    s_frame*CUSTNB(macc(nod),s_f(x))+
                    i_frame*CUSTNB(macc(nod),i_f(x));
&  PRINT(" iclop =",iclop,"icl=",icl,"sn",sn,"sg",sg);
 IF full(iclop,icl,sn+sg)<=buffer-sl_len
 THEN TRANSIT(dq_tok(iclop))
 ELSE TRANSIT(OUT);
SEND


/STATION/NAME = link;  & ??????????????????????????????????????
TYPE=INFINITE;
SERVICE(slot) =  BEGIN
         CST(n_to_n);
          IF bus=1
        THEN TRANSIT(a_switch(next))
         ELSE TRANSIT(d_switch(next));
        END;

SERVICE(i_f1,s_f1,dm1)=BEGIN
&  PRINT("inside link i_f1,s_f1,dm1");
                    CST(n_to_n);
                    IF (icl < 4)
                    THEN BEGIN
                        Strans(mac2,i_f1,s_f1)
                        END
                    ELSE BEGIN
                        CST(n_to_n);
                        Strans(mac3,i_f1,s_f1)
                        END;
                   END;
SERVICE(i_f2,s_f2,dm2)=BEGIN
&   PRINT("inside link  if2,s_f2,dm2");
                    CST(n_to_n);
                    IF (icl < 4)
                    THEN BEGIN
                        Strans(mac1,i_f2,s_f2)
                        END
                    ELSE BEGIN
                        Strans(mac3,i_f2,s_f2)
                        END;
                   END;
SERVICE(i_f3,s_f3,dm3)=BEGIN
&  PRINT("inside link i_f3,s_f3,dm3");
                    CST(n_to_n);
                    IF (icl > 3)
```

```
                THEN BEGIN
                    Strans(mac2,i_f3,s_f3)
                    END
                ELSE BEGIN
                    CST(n_to_n);
                    Strans(mac1,i_f3,s_f3)
                    END;
                END;


&*****************************************
& QA slot access queue
&*****************************************
& This station, is similarly defined than the one in appendix D.1, except that
& the segemnt are sent on the "link".
&*****************************************
$MACRO node
SERVICE = BEGIN
&   PRINT(" inside node if1,if2,s_f1,s_f2,dm1,dm2");
            rqm(ib,in):=rqm(ib,in)+1; & must set an empty REQ bit
            cd(ib,in):=1;           & AQ moved in countdown state
            cd_ct(ib,in):=req_ct(ib,in);    & CD counter = REQ counter
            req_ct(ib,in):=0;       & REQ counter is cleared
            WAIT(send(ib,in));      & Wait the next QA slot
&   PRINT( "leaving  wait for send in node");
            CST(sl_time);           & Segment transmission
&   PRINT( " after the CST sl_time");
            TRANSIT(link);
&   PRINT( "leaving  node after tx to link if1,if2,s_f1,s_f2,dm1,dm2");
          END;
$END

&*****************************************
& Access queue for sending on bus a.
&*****************************************
/STATION/ NAME = a_node(1 STEP 1 UNTIL au);
        Snode
&*****************************************
& Access queue for sending on bus a.
&*****************************************
/STATION/ NAME = d_node(1 STEP 1 UNTIL au);
        Snode


$MACRO print_out
BEGIN
 PRINT(" ");
 PRINT("TIME =",TIME);
 PRINTF("(1H,'Window size (packets)      :  ',F3.0)",window);
 PRINTF("(1H,'Packets/sec                :  ',F5.1)",ar_rate);
 PRINT(" ");
 PRINT("RESULTS");
 PRINT("Offer load (bits/sec)     : ",offer);
 PRINT("Total throughput (bits/sec) :  ",thru(count));
 PRINT("Mean end-to-end delay (sec) :  ",del(count));
 PRINT("Bandwidth BAl 0=off,1=on :",bwb_flag);
 PRINT("Bandwidth Balance Modulus:",bwb_mod);
 PRINT("priorirty mode to gatwway");
```

```
 PRINT("8 k buffers each way");
 PRINT(" ");
 OUTPUT;

END;
SEND


/CONTROL/ TMAX = 20.0;
        PERIOD=5.0;
        TEST=BEGIN    & Use to stop the simulation if the results converged
            IF TIME > 20
            THEN
             BEGIN
              FOR i:=1 STEP 1 UNTIL count DO
               BEGIN
                thru(i-1):=thru(i);
                del(i-1):=del(i);
               END;
              thru(count):=0;
              del(count):=0;

&*****************************************
& Measurement of the global system throughput and the I-frame mean
& end-to-end delay.
&*****************************************
              FOR i:=1 STEP 1 UNTIL st DO
               BEGIN
  thru(count):=thru(count)+sink1(i).NBOUT+sink2(i).NBOUT+sink3(i).NBOUT;
                del(count):=del(count)+sink1(i).tdelay/sink1(i).NBOUT+
                          sink2(i).tdelay/sink2(i).NBOUT +
                          sink3(i).tdelay/sink3(i).NBOUT;
               END;
              thru(count):=byte*info*thru(count)/TIME;
              del(count):=del(count)/3/st;
              Sprint_out   & Verify if results converge
              IF ((ABS((thru(count)-thru(count-1))/thru(count)) < 1E-2) AND
                 (ABS((thru(count)-thru(count-2))/thru(count)) < 1E-2) AND
                 (ABS((del(count)-del(count-1))/del(count)) < 1E-2) AND
                 (ABS((del(count)-del(count-2))/del(count)) < 1E-2))
              THEN STOP;
&             END;
           END;
        ENTRY= BEGIN
              PRINT("PACKET TRANSFER BETWEEN HOMOGENEOUS STATIONS");
              PRINT("-----------------------------------");
              PRINT(" ");
              PRINT(" ");
              PRINT(" ");
              PRINT("INPUT PARAMETER—> TOKEN RING");
              PRINT("Speed (bits/sec)          :",tok_spd);
              PRINT(" ");
              PRINTF("(1H,'Number of rings             : ',F3.0)",r);
              PRINTF("(1H,'Number of stations/ring     : ',F3.0)",st);
              PRINTF("(1H,'Number of nodes/ring        : ',F3.0)",nod);
              PRINTF("(1H,'I-frame length (bytes)      : ',F5.0)",i_frame);
              PRINTF("(1H,'S-frame length (bytes)      : ',F3.0)",s_frame);
              PRINT("Packet transmission (sec) :",byte*i_frame/tok_spd);
              PRINT("Ack transmission (sec)    :",byte*s_frame/tok_spd);
```

```
            PRINT("Timeout (sec)            :",t_timer);
            PRINTF("(1H,'Packets/sec              : ',F5.1)",ar_rate);
            PRINTF("(1H,'Window size (packets)    : ',F3.0)",window);
            PRINT("Timeout (sec)            :",t_timer);
            PRINTF("(1H,'buffer size (Kbytes)     : ',F5.3)",
                                    buffer/1024);
            PRINT(" ");
            PRINT("INPUT PARAMETER ---> DQDB MAN");
            PRINT("Bandwidth BAI 0=off,1=on :",bwb_flag);
            PRINT("Bandwidth Balance Modulus:",bwb_mod);
            PRINTF("(1H,'Number of slots/frame    : ',F3.0)",nb_sl);
            PRINTF("(1H,'Number of QA slots/frame : ',F3.0)",nb_qa);
            PRINTF("(1H,'Loop length (km)         : ',F4.1)",bus_len);
            PRINT("Bit rate (bits/s)        :",dq_spd);
            PRINT("Asynch bit rate (bits/s) :",
                                nb_qa*sl_len*byte/fr_time);
            PRINT("Slot time (sec)          :",sl_time);
            PRINT("Time between stations    :",n_to_n);
            PRINT("DQDB capacity(slots/sec) :",2*nb_sl/fr_time);
          END;
      EXIT = BEGIN
           $print_out
          END;
/EXEC/ BEGIN
      FOR i := 1 STEP 1 UNTIL count DO
       BEGIN
        del(i):=1E-12;
        thru(i):=1E-12;
       END;
      FOR i := 1 STEP 1 UNTIL r DO
       BEGIN
        ring(i).ir:=i;
        ring(i).mac:=1;
        tok_dq(i).ir:=i;
        dq_tok(i).ir:=i;
        token(i).icr:=i;
        FOR j:=1 STEP 1 UNTIL st DO
         BEGIN
          vs(i,j):=1;
          vr(i,j):=1;
          l_rn(i,j):=1;
          l_sn(i,j):=1;
          FOR k:=1 STEP 1 UNTIL range DO
           BEGIN
            pos(i,j,k):=k;
            seq(i,j,k):=k;
           END;
         END;
       END;
      FOR i := 1 STEP 1 UNTIL st DO
       BEGIN
        stat1(i).ir:=1;
        stat2(i).ir:=2;
        stat3(i).ir:=3;
        stat1(i).is:=i;
        stat2(i).is:=i;
        stat3(i).is:=i;
        stat1(i).num:=1;
        stat2(i).num:=1;
```

```
stat3(i).num:=1;
hlay1(i).is:=i;
hlay2(i).is:=i;
hlay3(i).is:=i;
hlay1(i).ir:=1;
hlay2(i).ir:=2;
hlay3(i).ir:=3;
hlay1(i).init:=1;
hlay2(i).init:=1;
hlay3(i).init:=1;
buff1(i).is:=i;
buff2(i).is:=i;
buff3(i).is:=i;
buff1(i).ir:=1;
buff2(i).ir:=2;
buff3(i).ir:=3;
llc1(i).is:=i;
llc2(i).is:=i;
llc3(i).is:=i;
llc1(i).ir:=1;
llc2(i).ir:=2;
llc3(i).ir:=3;
sink1(i).is:=i;
sink2(i).is:=i;
sink3(i).is:=i;
sink1(i).ir:=1;
sink2(i).ir:=2;
sink3(i).ir:=3;
tim1(i).is:=i;
tim2(i).is:=i;
tim3(i).is:=i;
tim1(i).ir:=1;
tim2(i).ir:=2;
tim3(i).ir:=3;
i_f1(i).iring:=1;
i_f2(i).iring:=2;
i_f3(i).iring:=3;
dm1(i).iring:=1;
dm2(i).iring:=2;
dm3(i).iring:=3; &?????
i_f1(i).icl:=i;
i_f2(i).icl:=i;
i_f3(i).icl:=i; &???????
i_f1(i).size:=i_frame;
i_f2(i).size:=i_frame;
i_f3(i).size:=i_frame;
i_f1(i).length:=i_frame*byte;
i_f2(i).length:=i_frame*byte;
i_f3(i).length:=i_frame*byte;
s_f1(i).icl:=i;
s_f2(i).icl:=i;
s_f3(i).icl:=i; &????????
s_f1(i).size:=s_frame;
s_f2(i).size:=s_frame;
s_f3(i).size:=s_frame;
s_f1(i).length:=s_frame*byte;
s_f2(i).length:=s_frame*byte;
s_f3(i).length:=s_frame*byte;
dm1(i).icl:=i;
```

```
   dm2(i).icl:=i;
   dm3(i).icl:=i; &?????
  END;
 FOR i:=1 STEP 1 UNTIL 3 DO
  BEGIN
   i_f1(i).iclop:=2;
   i_f2(i).iclop:=1;
   i_f3(i).iclop:=1; &??????
   s_f1(i).iclop:=2;
   s_f2(i).iclop:=1;
   s_f3(i).iclop:=1; &????????
   dm1(i).iclop:=2;
   dm2(i).iclop:=1;
   dm3(i).iclop:=1; &???????
   i_f1(i).ida:=i;
   i_f2(i).ida:=i;
   i_f3(i).ida:=i+3; &??????
   s_f1(i).ida:=i;
   s_f2(i).ida:=i;
   s_f3(i).ida:=i+3; &????????
   dm1(i).ida:=i;
   dm2(i).ida:=i;
   dm3(i).ida:=i+3; &???????
  END;
 FOR i:=4 STEP 1 UNTIL 6 DO
  BEGIN
   i_f1(i).iclop:=3;
   i_f2(i).iclop:=3;
   i_f3(i).iclop:=2; &??????
   s_f1(i).iclop:=3;
   s_f2(i).iclop:=3;
   s_f3(i).iclop:=2; &????????
   dm1(i).iclop:=3;
   dm2(i).iclop:=3;
   dm3(i).iclop:=2; &???????
   i_f1(i).ida:=i-3;
   i_f2(i).ida:=i;
   i_f3(i).ida:=i; &??????
   s_f1(i).ida:=i-3;
   s_f2(i).ida:=i;
   s_f3(i).ida:=i; &????????
   dm1(i).ida:=i-3;
   dm2(i).ida:=i;
   dm3(i).ida:=i; &???????
  END;
 FOR i:=1 STEP 1 UNTIL nod DO
  BEGIN
   mac1(i).is:=i;
   mac2(i).is:=i;
   mac3(i).is:=i;
   mac1(i).ir:=1;
   mac2(i).ir:=2;
   mac3(i).ir:=3; &??????
  END;
 FOR i := 1 STEP 1 UNTIL au DO
  BEGIN
   a_switch(i).in:=i;
   d_switch(i).in:=i;
   a_switch(i).ib:=1;
```

```
          d_switch(i).ib:=2;
          a_switch(i).icb:=2;
          d_switch(i).icb:=1;
          a_switch(i).inext:=i+1;
          d_switch(i).inext:=i-1;
          a_node(i).in:=i;
          d_node(i).in:=i;
          a_node(i).ib:=1;
          d_node(i).ib:=2;
         END;
       SIMUL;
     END;
/END/
```