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 Evaluation of Multimedia Communications Systems Supported by IEEE 802.6 DQDB MAN

by
Richard Vallée

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 0-315-60542-1

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

à ma soeur Anne

Contents

List of Figures

List of Tables

Abstract

The work of this thesis is concerned about the performance evaluation of multimedia communications systems supported by IEEE 802.6 DQDB MAN (Distributed Queue Dual Buses Metropolitan Area Network). The proposed IEEE 802.6 DQDB MAN standard, more commonly called DQDB, defines a high speed network allowing integration of voice and data over circuit and packet switching networks. Such networks may be particularly useful for multimedia applications within medical environments involving the transfer of voice and high resolution images. Another application of DQDB can be a backbone network interconnecting low-speed LANs (local area networks). Simulation results, based on the development of queueing models by using the QNAP2 (Queue Network Analysis Package) simulation software, show that DQDB performs well in the above mentioned applications. However, since DQDB transfers information over short fixed-length slots, congestion problems related to the slot formatting process are observed under certain system configurations for multimedia applications. In order to reduce these congestion problems, two system improvements are proposed. The first is to consider the use of images compressed according to the technique called the reduced-difference pyramid while the second one is to allocate higher access priority to the gateways in the LANs interconnected by DQDB.

Acknowledgements

Firstly, I wish to express my sincere appreciation to my thesis supervisor, Dr. Nicolas D. Georganas, for his patience, his encouragement, his guidance and his financial support.

I also would like to thank Dr. Misha Niktash and Dr. Oliver Yang for the valuable comments concerning the contents of this thesis.

The invaluable help of the University of Ottawa staff is also acknowledged. The assistance of Ms. Lucette Lepage and Ms. Michèle Roy, working at the Department of Electrical Engineering, has been particularly appreciated.

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

Je souhaite aussi remercier l'Université d'Ottawa et l'organisme Québeçois FCAR (Fonds pour la formation de chercheurs et l'aide à la recherche) pour leur support financier.

Je remercie également mes parents, ainsi que les membres de ma famille, pour leur encouragements et leur support.

Finalement, je tiens à remercier sincèrement ma compagne, Manon Gauvin, pour son incroyable support, sa patience, et ses encouragements tout au long de cette maîtrise.

Acronyms

ACS	Accredited Standards Committee
ANSI	American National Standards Institute
AQ	Access Queue
AQ_I	AQ at Priority Level I
ATC	Asynchronous Transfer Control
ATM	Asynchronous Transfer Mode
ATSU	Asynchronous Transfer Service User
AU	Access Unit
BOM	Beginning of MAC PDU
B-ISDN	Broadband Integrated Services Digital Network
CCITT	Comité Consultatif International Télégraphique et Téléphonique
CCSG	Configuration Control and Slot Generator
CD_I	Countdown Counter at Priority Level I
COM	Continuation of MAC PDU
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CRC	Cycle Redundancy Checks
DA	Destination Address
DM PDU	Derived MAC PDU
DQDB	Distributed Queue Dual Buses
DS	Digital Signal
DSG	Default Slot Generator
ED	End Delimiter
EOM	End of MAC PDU
FC	Frame Control
FCS	Frame Check Sequence
FDDI	Fiber Distributed Data Interface
HDLC	High Level Data Link Control
ICP	Isochronous Convergence Protocol
ICS	Information Check Sequence
IEEE	Institute of Electrical and Electronics Engineers
ISDN	Integrated Services Digital Network
ISO	International Organisation for Standardization
ITC	Isochronous Transfer Control
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
MAP	Manufacturing Automation Protocol
MCP	MAC Convergence Protocol
MHC	MAC Header Checksum
MID	MAC PDU Identifier
MSAP	MAC Service Access Point
NA	Non Arbitrated
NA_AC	NA Access Control
NT	Network Terminator
OSI	Open System Interconnection
PBX	Private Branch Exchange
PDU	Protocol Data Unit
PLS	Physical Signalling
QA	Queue Arbitrated
QA_AC	QA Access Control
QNAP	Queue Network Analysis Package
QPSX	Queue Packet and Synchronous Circuit Exchange
REQ	Request bit
REJ-frame	Reject supervisory frame
RQ_I	REQ Counter at Priority Level I
RR-frame	Ready to Receive supervisory frame
SA	Source Address
SAP	Service Access Point
SCSI	Small Computer System Interface
SD	Start Delimiter
SFD	Start Frame Delimiter
SONET	Synchronous Optical Network
SSM	Single Segment MAC PDU
STM	Synchronous Transfer Mode
S-frame	Supervisory frame
TE	Terminal Equipment
THT	Token Holding Timer
TOP	Technical and Office Protocol
TRT	Token Rotation Timer
TTRT	Target Token Rotation Time
UNI	User-Network Interface
VCI	Virtual Channel Identifier
WAN	Wide Area Network

Chapter 1

Introduction

1.1 Metropolitan Area Networks

Local area networks (LANs) have been implemented for allowing users to share storage, computer and peripheral resources. For instance, through a LAN, workstation users can access the service provided by remote devices such as printers, file and/or database servers, and computation servers. LANs running with speeds of up to several Mbps are currently available on the market. Ethernet, token bus and token ring are examples of the most widely used LANs. However, with the growth of demands for computer and communications resources and with the development of new computer applications requiring higher bandwidth, such as high resolution image transfers or compressed video communications, it appears that single LANs are not always sufficient. In some situations, it would be preferable to distribute workstations and servers over several LANs in order to avoid LANs congestion problems. These LANs can then be interconnected by a backbone network running at a higher speed. Such a backbone network may even be used to interconnect remote LANs distributed over a larger geographical area, within a city. In fact, the type of backbone network that has just been defined is commonly called a MAN (Metropolitan Area Network). A MAN, running at speeds up to 100 Mbps or more, can interconnect LANs over small as well as larger areas in a metropolitan area (over 50 km in diameter). LANs are connected to a MAN by gateways which might have to provide various protocol services to ensure proper network interworking.

In the past decades, there has been growing interests for integrating voice, video and data communications. An example of applications, which can benefit of such integration, is a real-time multimedia conference system where two or more users, situated at different locations, can discuss about images or any kinds of document which are simultaneously displayed on their screen [AHUJA,FORS,KOSIT,SARIN]. Thus, in addition to interconnect low-speed LANs, MANs have also been proposed to provide integration of voice, video and data communications through the same network. Since voice, video and data communications have different characteristics, the MAN might be designed in consequence. Voice and video communications, being isochronous (uniform in time and recurring at regular equal intervals), require continuous bandwidth, low delay, and low delay variations between each sending. In addition, video communications require isochronous bandwidth up to several Mbps. On the other hand, data communications are bursty which means that their bandwidth demand varies and may contain peak periods which can cause temporary congestion within the network. However, in contrast to voice and video communications, data communications have less severe delay requirements. Hence, packet switching techniques [BERT,TANEN] can be used to provide communications requirements between users having data packets to transmit. Thus a typical MAN will be a high speed network providing fixed bandwidth for isochronous communications through circuit switching and supporting packet switching for transferring asynchronous packets.

1.2 The IEEE 802.6 Working Group

In 1982, the IEEE 802.6 committee started elaborating a standard for defining a MAN. The first proposal was a slotted ring approach [IEEE85]. Isochronous and non-isochronous slots (a slot referring to a fixed-length data unit periodically generated) were respectively defined for supporting voice (and video) and data communications. Each isochronous slot was assumed to be allocated by a master node upon demand. At the same time, each empty non-isochronous slot was assumed to be grabbed by the first node seeing the slot and having a packet queued

for transmission. However, the working group adopted later another proposal under the name DQDB (Distributed Queue Dual Buses) [IEEE87B,IEEE88B]. This proposal is based on a protocol called QPSX (Queue Packet and Synchronous Circuit Exchange) which was defined by an Australian research group [BUD86,NEW86,NEW88B]. DQDB is still defining isochronous and non isochronous slots, but the access to the non isochronous slots is based on a reservation scheme. The DQDB architecture comprises two unidirectional contra buses allowing full duplex communications between each pair of nodes (Fig. 1.1). The choice of DQDB (or QPSX) was made since this network can support high speed integrated voice/data communications, can be used over a large area without significant performance degradation, can be implemented as a ring or bus network, and is intended to be compatible to the ATM (Asynchronous Transfer Mode) protocol which is the transport mode mechanism presently proposed by the CCITT (Comité Consultatif International Télégraphique et Téléphonique) Study Group XVIII [CCITT89] responsible for standardizing B-ISDN (Broadband Integrated Services Digital Network) .

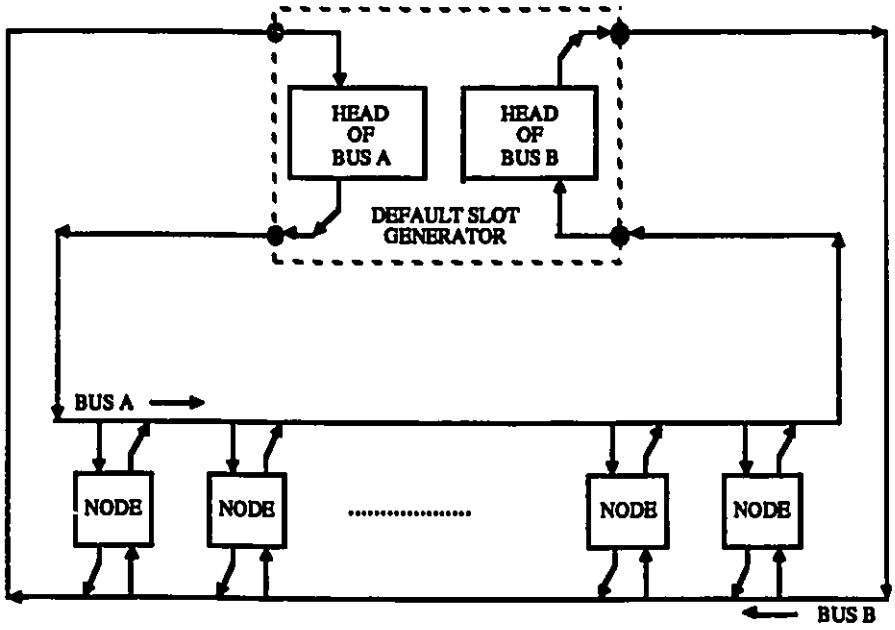


Figure 1.1: IEEE 802.6 DQDB MAN looped bus topology.

1.3 Multimedia Communications Systems Supported by DQDB

By offering high speed data transmissions and integrating voice and data communications, DQDB appears as a potential candidate for supporting multimedia communications systems in which various kinds of data can be exchanged between users. Thus, this section reviews such multimedia communications systems which must provide low response times to the end users as well as integration of different kinds of services. In addition, the work done by the MUSIC project group on multimedia medical communications systems supporting radiological image transfers [GEOR] is introduced.

1.3.1 General Overview of Multimedia Communications Systems

Computers have now become a common tool in all sectors of society (military, industrial, academic, and domestic). By digitizing various kinds of information such as text, pictures, and even voice recording, "multimedia" communications systems integrating all these types of information can be designed. As previously mentioned, a typical example of multimedia applications is a real-time conferencing system supported by voice (and possibly video) and data communications between conferees. Another example is the information retrieval operation performed by a remote user through a multimedia database server containing any kinds of digitized information such as text, graphics, pictures, drawing, and voice recording [GEOR,KARM].

The future "multimedia desktop device" may be a workstation composed of a high resolution screen, a pointing device which can be a mouse or an optical pen, a microphone and a speaker (or simply a phone), and possibly a camera and a video monitor. Within a real-time conference, the voice and the pointing device are the major tools for animating the conference. Such "multimedia desktop device" may be especially implemented in business offices and in hospitals. In business offices, the real-time conference components may play

the role of electronic blackboard used by the conferees for making presentations and elaborating projects. In hospitals, since large amounts of various kinds of information concerning patients' examinations are produced each day, underlying research studies aim to implement multimedia database servers containing all patient's details (personal data, medical tests, radiographies, physician's and radiologist's reports) [GEOR,KARM]. Indeed, by digitizing any data collected from medical tests, all information can be stored in high capacity disks (instead of never-ending numbers of sheds dispersed in the hospitals) and be retrieved in an efficient way in order to improve patient care.

1.3.2 Radiological Image Transfers within the MUSIC Project

At the University of Ottawa, the MUSIC (Multimedia System for Integrated Communications) project group is conducting studies to design and evaluate the performance of various multimedia medical communications systems [GEOR,KOSIT,VAL89A,VAL89B,VAL89C]. The purpose of such systems is to allow radiologists and physicians, using independent workstations, to access remote multimedia database servers for retrieving any kinds of information such as radiological images, varieties of medical test results, and voice and text reports [KARM]. Hence, one of the MUSIC group research objectives is to design communications systems which provide, in reasonable time, the access to the information contained in database servers. In particular, one of the most important factors of the system performance is the system response time for image display. Indeed, images produced by the radiology department are one of the main sources of information produced in a hospital. Currently, a high percentage is X-ray images, but a growing fraction comprises newer imaging modalities such as ultrasound, nuclear medicine, X-ray and emission tomography, and magnetic resonance imaging [HUANG,TURN]. Moreover, high resolution images are required to allow accurate physicians' diagnosis (1000 x 1200 pixels x 8 bits/pixel for 14" by 17" film [GEOR]). Each image is then represented by a sizable amount of data (over 1 Mbytes) which can occasionally cause delay problems in the system transferring the data between remote devices. Indeed, the reading of an image from the database server disk, its segmentation into packets and its

transmission through the network are costly in terms of time. Hence, the use of techniques for compressing images are also considered by the MUSIC group for reducing delays due to the image manipulation [WANG]. Besides, a new interest for image browsing activities, which implies sending a set of images in a short period of time, emphasizes the advantages of using compressed images.

Currently, the MUSIC group has developed a prototype multimedia communications system, which is undergoing clinical trials at the Ottawa Civic Hospital.

1.4 Outline of the Thesis

This thesis presents an evaluation of the use of DQDB within multimedia communications systems for supporting two particular applications: transfer of high speed data packets, and interconnection of LANs situated in remote buildings. In both cases, large amounts of data have to be transferred through DQDB. The performance of DQDB for carrying such applications is then analyzed. In order to evaluate the performance of the systems under study, queueing models have been developed by using the QNAP2 (Queue Network Analysis Package) software [QNAP2] which is summarized in appendix B. QNAP2 is used for queueing system modeling and performance evaluation. It incorporates many of the known analytical solutions of queueing networks, and also a discrete-events simulator that has been used for the performance evaluations presented in this thesis.

This thesis is composed of six chapters. Chapter one just presented the context in which MANs has been developed, summarized the proposed IEEE 802.6 DQDB MAN standard, and discussed multimedia environments where DQDB can be used as a transport network.

Chapter 2 presents an overview of the standards which are involved in the work of this thesis. The OSI Reference Model is presented as well as the IEEE 802 standards' family describing standards for the data link and physical layers of the OSI Reference Model. An overview of the Integrated Services Digital Networks (ISDNs) proposed by the CCITT Study

Group XVIII is also given.

Chapter 3 describes in details the proposed IEEE 802.6 DQDB MAN standard. This chapter also discusses the performance of DQDB for supporting packet switched communications, and proposes a simulation model of the DQDB protocol controlling the access to the asynchronous bandwidth.

Chapter 4 presents a performance study of DQDB used as a high speed network for supporting radiological image browsing between database servers and medical workstations. The elapsed time, measured from the time an image request is issued until the image is displayed in the monitor, is the key parameter considered for evaluating the system performance.

Chapter 5 presents a performance study of an interwork system in which DQDB is used as a backbone network to interconnect low-speed LANs. This system is evaluated regarding two kinds of applications: transfer of short data packets between homogeneous workstations situated on different LANs, and browsing of radiological images by a medical workstation on a LAN serving a hospital, linked with a database server situated in a remote LAN serving another hospital.

Finally, chapter 6 concludes this thesis by reviewing the major characteristics of DQDB, by outlining the advantages and drawbacks of using DQDB in the communications systems defined in chapters 4 and 5, and by proposing further work.

Chapter 2

Review of Relevant Standards

This chapter presents a review of the main standards pertaining to various computer communications networks. This review is intended to show the environment which involves the IEEE 802.6 DQDB MAN standard. The OSI Reference Model, composed of the seven OSI layers, is first reviewed, since this model is often used as a template for designing and implementing communications networks. Then, the next section presents the IEEE 802 standards' family which comprises various system configurations from the physical to the data link layer based on the OSI Reference Model. Thus, the IEEE 802.6 DQDB MAN is involved in this standards' family. Finally, a brief description of the proposed ISDN and B-ISDN standards is given, since DQDB is intended to be compatible with these standards which will play an important role for future implementation of private as well as public telecommunications networks.

2.1 OSI Reference Model

2.1.1 General Overview

In 1977, the International Organization for Standardization (ISO) recognized the special and urgent need for standards for heterogeneous computer networks and decided to create a new subcommittee for "Open Systems Interconnection" (OSI) [ZIM80]. This committee produced a specific layered communications architecture, the OSI Reference Model, which was approved

in 1983 [ZIM83]. This layered model is composed of seven layers (see Fig. 2.1): the physical, data link, network, transport, session, presentation and application layers.

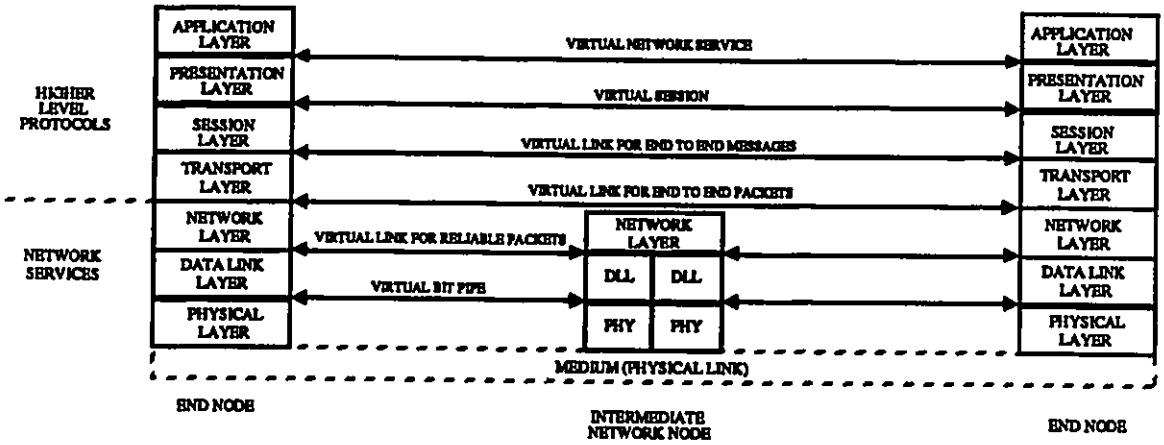


Figure 2.1: OSI Reference Model.

Each layer of the OSI Model is intended to give services to the layer above. Indeed, the (N)-entities in the (N)-layer provide service to (N+1)-entities in the (N+1)-layer above. The N-entity is thus defined the service provider to the (N+1)-entity. The interface between two layer entities is called the "(N)-service access point" (N-SAP). Figure 2.2 illustrates the relations between layers, entities and SAPs. Thus a (N)-entity can serve a number of (N+1)-entities through N-SAPs. On the other hand, each (N)-entity is assumed to have a logical link (or virtual connection) with a peer (N)-entity serving a remote user (with the exception of physical entities which are physically linked). The entities, under the (N)-entity, provide the physical connection between the peer N-entities.

Protocol Data Units (PDUs) are defined to provide the transfer of information between peer entities via a peer protocol. In Fig. 2.3, the different PDUs used in the OSI Reference Model are presented. Data units are first created at the application layer. They are passed to the presentation layer after being encapsulated (by adding some control bits), and then become presentation PDUs containing a presentation service header (SH). This encapsulation process may be repeated at each lower layer. At the destination station, the control bits added

by the sending (N)-entity are only examined and removed by the peer (N)-entity. The rest of the PDU will be sent to next higher layer entity.

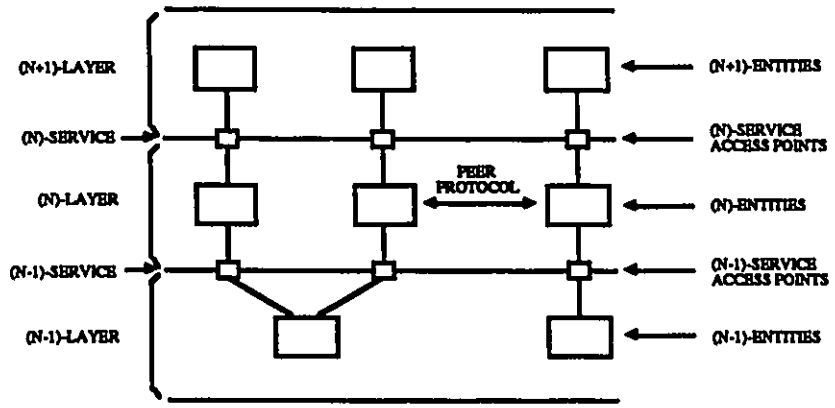


Figure 2.2: Relations between layers, entities and service access points (SAPs).

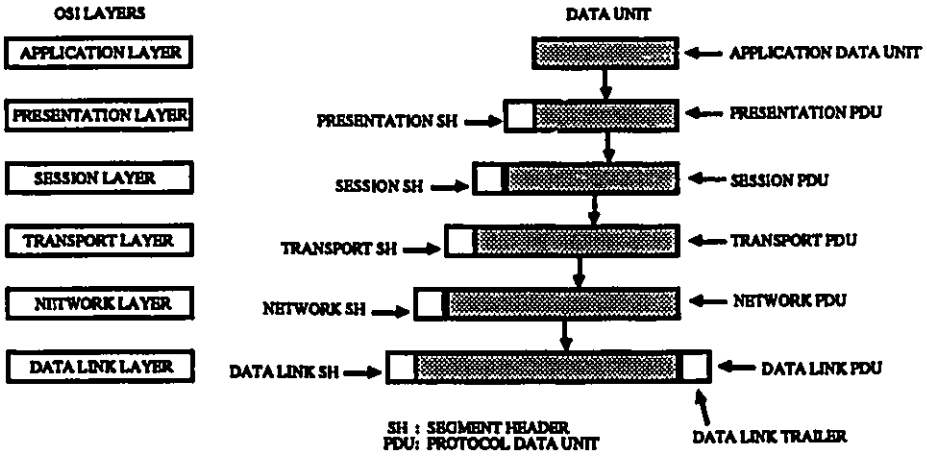


Figure 2.3: Flow of protocol data units (PDUs) through the OSI layers.

2.1.2 The Seven OSI Layers

In this section, the seven OSI layers are summarized. The four upper layers, which provide protocols residing in end stations that are involved in the end-to-end data communication

process, and which carry out the processing required to deliver the data correctly and in a recognizable form to the user, are first presented.

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 semantics or meaning of the information being exchanged between the application entities. Examples of application services are virtual terminal, file and job transfer and manipulation services and protocols [SCHW]. The virtual terminal service is used to provide terminal access to a user process in a remote system. The file service provides for remote access, management and transfer of information stored in the form of files, such as files contained in typical file and/or database servers [GEOR]. The job service is intended to carry out distributed job processing which involves functions such as job submission, job processing and job monitoring.

The **presentation layer** is responsible for the negotiation of transfer syntax of the data unit generated by or addressed to the application layer entities. Indeed, its major functions can be 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.

Under the presentation layer, 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 layer also allows remote users to identify themselves in establishing communications with other users.

The **transport layer** is designed to relieve the session entities from the means of achieving reliable and cost-effective transfer of data. The importance of this layer depends on the design of the lower layers provided. If the lower layers provide virtual circuit services (all packets belonging to a logical connection use the same route), 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 (a connectionless scheme where

each packet generated by the transport layer can be routed differently through the network) is provided by the lower layers, the transport layer protocols may have to assure that packets are delivered in sequence without loss, error or duplication. For this purpose, various flow control schemes can be used, such as the window flow control [KNIG,TANEN]. On the other hand, the transport layer is responsible for breaking messages coming from the session layer entities into smaller packets, and for doing the inverse process at the other end. It can also multiplex several low rate sessions or split a high rate session into multiple sessions at the network layer in order to get higher quality of service. Finally, the transport layer is also used in gateways interconnecting networks using different packet types (different overhead and maximum packet length) and different communication protocols. TCP (Transport Control Protocol), combined with IP (Internet Protocol), is a well-known example [DOD83A,DOD83B].

Whereas the four higher layers are end-to-end communication processes, the three lower are more concerned with the network environment. In fact, these layers are not only used at the end stations but also at the intermediate network nodes (see Fig. 2.1).

The **network layer** mainly carries out the necessary 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 all together in implementing routing and congestion control. This can be done by algorithms using geographically distributed processes. These processes must periodically interchange information in order to adequately react and prevent congestion problems. Examples of congestion control methods used at the network layer are the sliding window mechanism, a scheme where packets sent by the source stations are choked and/or discarded by nodes whose buffers are temporarily full, and also the schemes where buffer preallocation is done at the virtual circuit set up. Furthermore, similar to the transport layer, the network layer can also provide virtual circuit or datagram services, error detection and recovery.

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. In fact, like the two previous layers, the DLL can provide an error and loss recovery procedure by using the window flow control mechanism. Moreover, the DLL provides core services, such as frame error detection (by using a frame error check), delimiting (by using special bits), and addressing in order to route the frames through the network. The DLL PDU produced by the DLL will then be sent, without any other modifications, on the medium through the physical layer. For multiaccess communications performed on star, bus or ring networks, the DLL is usually divided into two sublayers [ANSI85A,ANSI85B, ANSI85C,ANSI85D]: the **logical link control** (LLC), and the **medium access control** (MAC) sublayers (see Fig. 2.4). The LLC is concerned with the link level flow control, delivery without loss, duplication or misordering and the error recovery of PDUs exchanged between logical service access points (LSAPs) which are the interfaces between the LLC and the network layer entities. Serving the LLC entities, the MAC sublayer is intended to provide core services and the management of the multiaccess link so that MAC PDUs can be sent by each node without constant interference from the other nodes. The MAC protocols can differ considerably depending on the method used to access the network. In the next section, I will present a detailed description of various MAC protocols defined by the IEEE 802 committees.

Finally, at the bottom of the OSI layers, the **physical 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.

The OSI Reference Model, comprising the seven layers, serves as a template for implementing any computer network protocol. It must be pointed out, however, that there are some duplicated efforts, in particular in layers 2 to 4 (data link, network and transport layers). Indeed, the utilization of some procedures, such as the one performing PDU error recovery, may seem redundant. Consequently, in some implementations, the transport and network layers may be integrated into a single layer in order to reduce data processing costs. In other configurations, such as the ones based on the frame relaying concept presented in section 2.3.1,

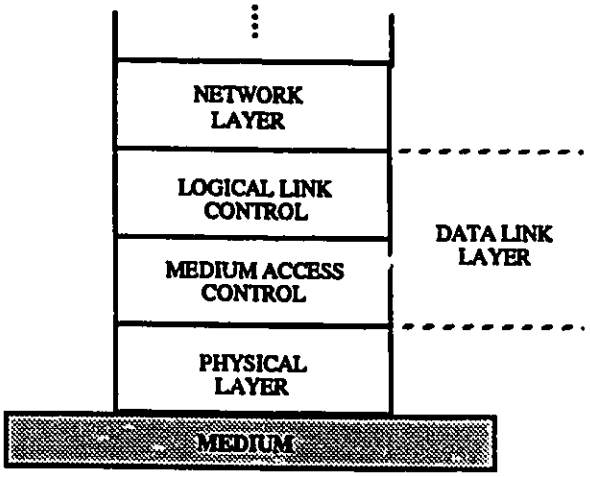


Figure 2.4: The two subdivisions of the DLL for a multiaccess communication link: the LLC and the MAC sublayers

the LLC protocol can be used as an end-to-end protocol between end stations, which allow simplifying the protocol defined at the network layer.

2.2 Family of IEEE 802 Standards

Figure 2.5 shows the different existing as well as emerging standards which are under study by the IEEE 802 committees. They conform to the OSI Reference Model reviewed in section 2.1. At the physical and media access control layers, the IEEE 802.3 (CSMA/CD), the IEEE 802.4 (token bus), and the IEEE 802.5 (token ring) standards have been already established. Moreover, committees are presently working in the definition of the two other standards: IEEE 802.6 DQDB MAN and IEEE 802.9 IVD (Integrated Voice/Data) LAN interface. At the upper part of the data link control layer, there is the IEEE 802.2 Logical Link Control (LLC) sublayer which has been already established. Finally, IEEE 802.1 is presently in preparation and its purpose is to describe the relationships among the previous standards as well as internetworking and network management aspects. In the next subsections, the five physical

and MAC layer standards, and the IEEE 802.2 LLC protocol are reviewed.

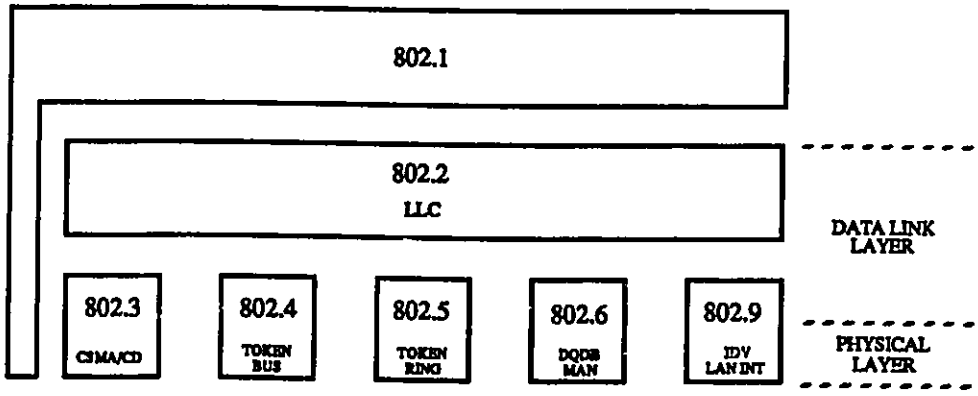


Figure 2.5: IEEE 802 family of standards.

2.2.1 IEEE 802.3 CSMA/CD

General Description

The IEEE 802.3 CSMA/CD (Carrier Sense, Multiple Access with Collision Detection) standard [ANSI85B], often juxtaposed with the Ethernet protocol, is based on the use of a common bus shared by a number of stations (see Fig. 2.6). While the Ethernet specification just calls for a coaxial cable physical channel driven at 10 Mbps, the IEEE 802.3 standard is intended to encompass several medial types and techniques for signal rates from 1 Mbps up to 20 Mbps. In the CSMA/CD protocol, each station, before transmitting, must listen to the bus. If there is no energy detected on the medium during an idle time (slot time), the station can transmit the queued packet. If, after initiating a transmission, the message collides with that of another station, then each transmitting station intentionally sends a few additional bytes (jam signal) to guarantee that all transmitting stations on the network detect the collision. The station remains silent for a random amount of time (backoff) before attempting to transmit again. To perform retransmission of frames, the CSMA/CD protocol uses a truncated binary-backoff algorithm: the delay before retransmitting is assumed to be a random number r of slot times

(defined below) distributed uniformly over the range 0 to 2^n, with n the n^{th} retransmission attempt, until n reaches 10. The retrial interval then remains at 0 to 2^{10} until n reaches 15. At this point the collision event is reported as an error to the higher layers.

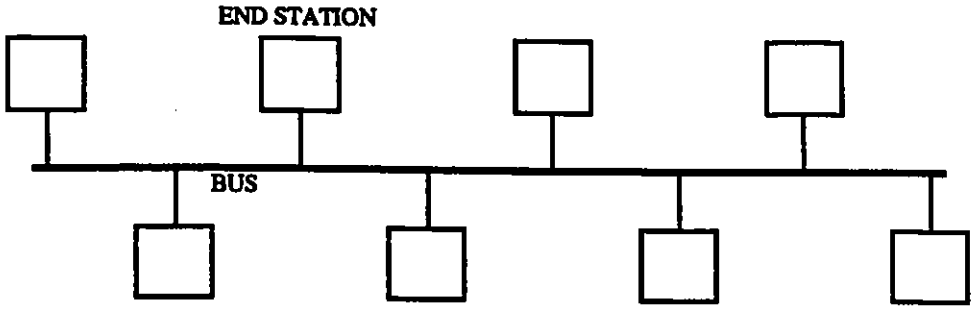


Figure 2.6: CSMA/CD bus topology.

The CSMA/CD protocol defines a slot time which largely determines the dynamics of collision handling. The single parameter describes three important aspects of collision handling:

1. It is an upper bound on the acquisition time of the medium.
2. It is an upper bound on the length of a frame fragment generated by a collision.
3. It is the scheduling quantum for retransmission.

The first aspect represents the minimum allowed length of a CSMA/CD PDU queued for transmission. If the initial CSMA/CD PDU does not have the length required for proper operation of the protocol, a pad field must then be added (see appendix A, section A.1). In fact, the minimum frame length (equivalent to the slot time) shall be long enough in order to get reasonable system performance [BUX81] (this is discussed below). The second aspect illustrates the maximum possible length of the frame fragment generated by a collision involving the stations situated at each end of the bus. The third aspect refers to the slot time used as a scheduling quantum for the retransmission of frames. To fulfill all these aspects, the

slot time shall be larger than the sum of the physical layer round-trip propagation time and the jam time generated by the stations having detected a collision.

Performance of IEEE 802.3 CSMA/CD

In evaluating the CSMA/CD performance, the parameter $a = \tau/m << 1$ has been found to have the most important influence, τ being the end-to-end propagation delay along the bus while m being the frame transmission time [BUX81]. In fact, it has been shown that the parameter a must be kept low in order to get acceptable system throughput. If a decreases, collisions are more readily detected. On the contrary, if a increases towards 1, which means that the propagation delay tends to be as long as the frame transmission time, the probability of collision is increased which will cause system degradation. Thus, the parameter a explains the fact that CSMA/CD implementations like Ethernet must be limited to low speed, short bus length and use a minimum frame length. In the case of Ethernet, the speed has been fixed to 10 Mbps while the bus length has been bound to 1.5 km and the slot time fixed to 512 bits [SCHW]. Finally, it must be pointed out that one of the most important advantage of CSMA/CD on other types of access method is the short access time under low traffic intensity in the system (as it is equal to the waiting time to detect an idle period).

2.2.2 IEEE 802.4 Token-Passing Bus

General Description

The IEEE 802.4 token-passing bus access method [ANSI85C] is intended to be implemented on the same type of medium as the CSMA/CD protocol. However, its access method is based on the use of a token which regulates the access to the bus. The station having grabbed the token has momentary control of the medium. The station can then send data frames until its queues are empty or its token holding timer (THT) expires (timer which prevents any single station from monopolizing the network). Subsequently, the station passes the token to

another station residing on the medium. Each station owns a logical number, which allows transferring the token between stations by following a descending order. Each participating station knows its successor and predecessor. Then, after a station has completed transmitting its data frames, it passes the token to its successor by sending a "token" MAC control frame. This frame contains, among other information, the address of the station generating the token and the address of the station which succeeds in the logical ring. Thus the successor station, hearing the token addressed to itself, grabs the token and begins to transmit data frames. The station having the lowest address will send the token to the station having the largest address. Thus, a logical token-passing ring is formed (see Fig. 2.7).

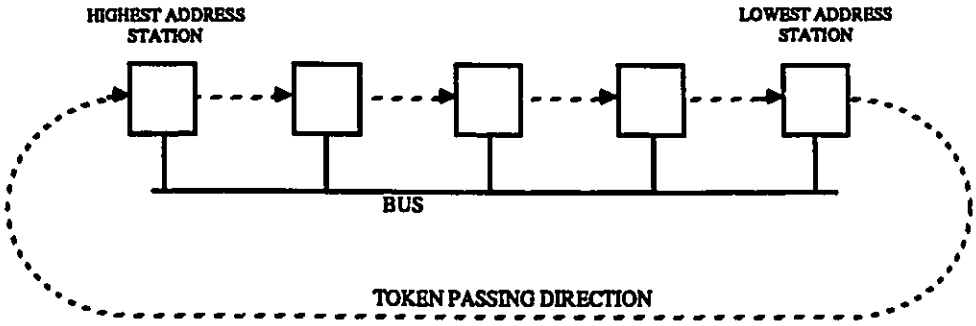


Figure 2.7: Logical token-passing ring.

Various MAC control frames are defined by the IEEE 802.4 standard. One is used for transferring the token between stations. Another is used to reestablish a token lost due to communications problems on the medium. Other control frames are also used to insert new stations or reject physically disconnected stations in or from the logical ring. In particular, each station sends periodically a MAC control frame, called "Solicit Successor", which invites new stations to be integrated in the logical token ring.

The IEEE 802.4 standard also defines eight service classes of priority, class 7 being the highest and class 0 being the lowest. In addition, 4 access classes are defined that correspond to 4 request queues for storing frames pending transmission: access class 0 for service class 0 and 1, access class 2 for service class 2 and 3, etc. (see Table 2.1).

Service classes	*Access classes*	*Priority*
0, 1	0	lowest
2, 3	2	.....
4, 5	4	.....
6, 7	6	highest

Table 2.1: Mapping of the service classes into the access classes in each token-passing bus station.

The priority scheme proceeds as follows. When a station receives the token, it first serves the highest access class queue. Upon the token reception, a fixed value, regardless of current network loading, is loaded in the THT (token holding timer). Then, the station can send frames until this timer expires. If this timer expires before sending all the queued frames, the station must release the token to the next station on the logical ring. On the contrary, if the station sends all the frames of the highest priority queue before the THT expires, then the station begins to serve the lower priority queues.

The service of the three lower access classes is based on the use of a timer called token rotation timer (TRT). The TRT is used to allocate more or less time for transmitting frames depending on the time spent by the token to perform one logical ring rotation. The access class service algorithm consists of loading the residual value from the TRT into the THT and reloading the same TRT with the target rotation time for that access class (thus frames sent by the station, for this access class, are accounted in the access class's next token rotation time computation). Then, if the THT has a remaining positive value, the station can transmit frames at this access class until either the THT expires or the access class queue is empty. When either event occurs the station begins to service the next lower access class. Only after doing the same with all the lower access classes, the token will be passed to the next station. Thus, the TRT method allows lower access class traffic to fill in bandwidth which is not used by higher access class traffic.

On the other hand, bit rates of 1, 5, and 10 Mbps are suggested for implementation by

the IEEE 802.4 standard. For example, the MAP/TOP (Manufacturing Automation Protocol/Technical and Office Protocol) standard adopted the token-passing bus running at 5 or 10 Mbps [MCGUF].

Performance of the Token-Passing Bus

The major disadvantages of the token-passing bus access method over CSMA/CD are the complexity of the protocol and the waiting time for a station which has to send frames under low traffic intensity [BUX81]. However, the token-passing method is more fair and allows the use of several classes of priority. On the other hand, some performance work [PANG88] has showed that, under heavy load conditions, the station service time will oscillate in one of many periodic patterns which motivates further research [PANG89] to dynamically fix the TRT period for improving network sharing and utilization.

2.2.3 IEEE 802.5 Token-Passing Ring

The IEEE 802.5 token passing on a ring [ANSI85D] is mostly based on work on the token ring technique developed by researchers at the IBM Research Laboratory in Zurich, Switzerland [BUX81]. Its concept is quite simple and similar to the token-passing bus, but now a physical ring is used instead of a bus (see Fig. 2.8). Each station, having packets queued for transmission, has to capture the token passing on the ring before performing packet transmission. When the packet transmission is completed, the station passes the token to the next station on the ring. Hence, each station is either in transmit or repeat state. In transmit state, it sends out its own frame after receiving the permission to do so. It transmits until it has completed transmission of all frames queued or until the transmission of another frame cannot be completed before the token holding timer (THT) expires. In repeat state, it outputs a received frame bit by bit back onto the ring. It may copy the frame while repeating it if it recognizes its own address. It may also modify some control bits to indicate that the frame was correctly received.

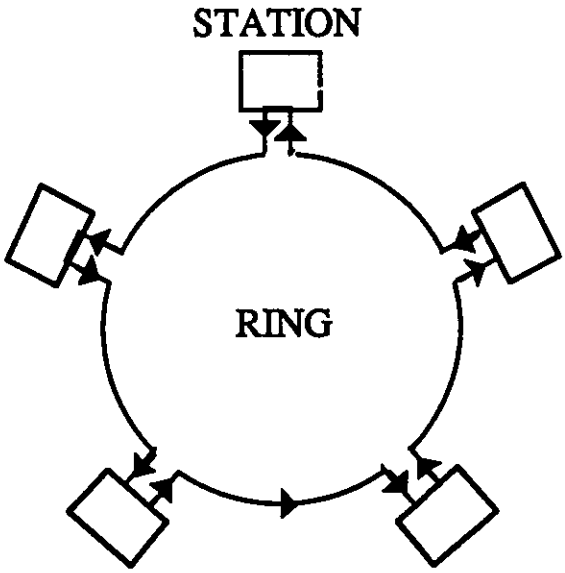


Figure 2.8: Token ring topology.

Like the IEEE 802.4 standard, the IEEE 802.5 standard defines a priority mechanism which comprises eight levels of priorities. However, the IEEE 802.5 priority mechanism is more centralized. In fact, there is only one priority level recognized on the ring at a given time, and this is indicated in the token exchanged between the stations. When stations grab the token, they can only send frames having the same or a greater priority level than the one of the token. In addition to indicate the ring priority in the token, the token ring priority mechanism is based on the use of reservation bits contained in each token ring frame header (see appendix A, section A.3) and on the use of stack registers present in each station. The reservation bits are used by the stations requesting for higher token priority. This may force the next station generating the token to increase the token ring priority. The stack registers are used by the station generating the token to store the new and old priority levels in order to bring back or reset the ring to the original service priority, at a later stage. The term stacked is utilized, since multiple new and old priority level values may be stored if a given station raises the service priority of the ring more than once before the service priority is returned to a lower priority level.

Furthermore, the IEEE 802.5 committee is defining the text of the standard for the "16 Mbps Token Ring" [GLASS]. In addition to this increase in speed, a new early token release which allows multiple frames to be on the ring at a given time (but only one free token) is also proposed. Indeed, workstations will be authorized to transmit a token immediately after sending a frame, without having to wait for the frame to return (which is not the case in the present IEEE 802.5 standard). Efficiency of 95% is guaranteed for a user sending data frames larger than 128 bytes, regardless of ring length [IBM]. Hence, in this thesis, these new improvements are also considered in the performance study presented in chapter 5.

Performance of IEEE 802.5

Some performance studies [BUX81] have showed that better performance is obtained by using the token ring access method compared to the CSMA/CD method, since the CSMA/CD protocol, under heavy traffic, leads to inefficient operation (increase of the number of collisions). The difference between them is amplified if the ratio of propagation delay to packet transmission time (parameter a) is increased, such as in the case of increasing the network speed. On the other hand, under the use of only one priority in the network, the token-passing bus and the token-passing ring access methods should lead to similar performance, since the concept of the logical ring is similarly applied in both cases. There can be difference in results due to the different time needed to exchange the token between the stations, since in one case the medium is a bus and the other a ring. On the other hand, it should be interesting to compare the performance of the priority mechanism used by each network. At a first view, it can appear that the token-passing ring access method is better adapted to allow the sending of all the frames having higher priority access the network before the other ones. Indeed, the token-passing ring access restricts stations to only send frames having equal or higher priority than that of the token, while in the case of the token-passing bus access method, the sending station, if its THT has not expired, can transmit low priority frames, although some other stations may have higher priority frames queued for transmission.

2.2.4 IEEE 802.6 DQDB MAN

As mentioned in the introduction, the purpose of the IEEE 802.6 committee has been to propose a high speed network driving at speeds over 100 Mbps, integrating packet and circuit switching, and allowing to interconnect LANs running at lower speeds such as IEEE 802 LANs. A consensus recently emerged within the IEEE 802.6 committee on the type of protocol to be used as a MAN standard, and DQDB (Distributed Queue Dual Buses) was selected. The architecture of DQDB comprises two unidirectional contra buses which can be joined to become looped buses (see Fig. 1.1). Each node of the network is attached to both buses, allowing full duplex communication between each pair of nodes. The DQDB capacity is dynamically allocated to meet the demand of the isochronous traffic. Capacity that is not allocated to isochronous traffic can be shared by the packet switched traffic. The timing structure of DQDB is based on implicit frames of 125 μs which comprise a given number of fixed slots shared between isochronous and non isochronous communications. Initially, slots of 32 octets were defined [IEEE87A]. However, in order to facilitate the interconnection of DQDB with B-ISDN (Broadband Integrated Services Digital Network) [CCITT89], the slot length has been increased to the ATM (Asynchronous Transfer Mode) cell length which is presently 69 octets [IEEE88A,IEEE88B,MOLL]. In the next chapter, a detailed description of the different characteristics of DQDB is given, since the present thesis is based on the use of this standard. The next paragraph briefly presents the FDDI-II standard which is intended to provide the same types of services as DQDB (high speed transmissions, integrated data/voice)

FDDI-II: Another MAN

FDDI-II (Fiber Distributed Data Interface) is another MAN proposal which has been defined by the Accredited Standards Committee (ASC) X3T9 [ROSS]. Indeed, FDDI-II is a high speed network offering bit rate up to 100 Mbps under a ring architecture, and intended to support circuit and packet switched service modes. Three kinds of traffic are offered by FDDI-II: isochronous, synchronous and asynchronous traffic. Dedicated bandwidth can be assigned to

provide isochronous services. The allocation of the synchronous and asynchronous bandwidth is mainly based on a token-passing method. Synchronous traffic is that traffic where frames can be transmitted inside a restricted time. Indeed, in the FDDI-II protocol, stations can negotiate for a target token rotation time (TTRT) which represent the half of the maximum delay within the frame is assured to be transmitted. By using token rotation timers (TRTs) in each station, a maximum token rotation time can be provided in order to assure frame transmission with a delay not exceeding twice TTRT. Unallocated bandwidth to synchronous traffic can be used on an as available basis for asynchronous traffic.

2.2.5 IEEE 802.9 IVD LAN Interface

For few years, the IEEE 802.9 IVD (Integrated Voice/Data) LAN interface standard committee has been seeking to standardize an interface to an integrated voice/data workstation (IVDWS) providing both ISDN and LAN services [IEEE89]. The major purpose of the standard is to offer potential saving to the business customer in terms of reduced components (one port instead of two), and in simpler management and maintenance (one network instead of two). Thus IVDWSs can be interfaced independently to the integrated communications network. In Fig. 2.9, the IVD interface is represented. The connections between the access unit (AU) and the IVDWSs is a star environment. The interface is intended to provide high bandwidth packet and circuit switching on a full duplex interface to each IVDWS. The unshielded telephone twisted pair wiring in a physical port-to-port configuration is proposed as medium. On the other side of the AU, there may be connections to an existing ISDN switch, IEEE 802 LAN, and/or an IVD LAN such as DQDB and FDDI-II.

The IVD interface consists of 4 full duplex channel types (2B+D+P+C channels) over which different types of services may be provided. Figure 2.10 represents the allocation of the bandwidth to IVD channels. The channel types are:

- *B-channel*: 64 kbps channel intended to support digitized voice and other subscribed ISDN services.

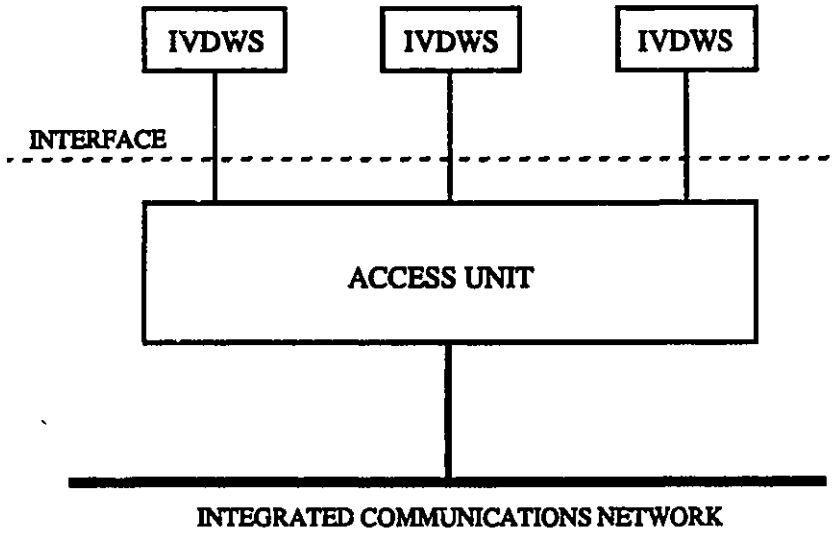


Figure 2.9: IEEE 802.9 IVD LAN Interface.

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

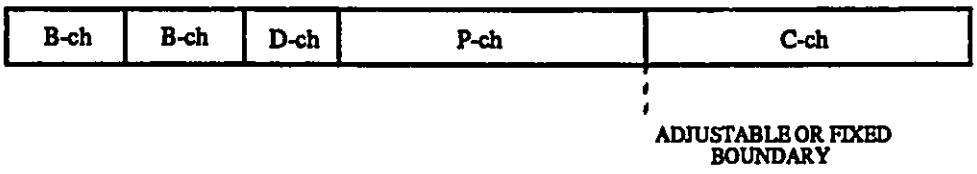


Figure 2.10: Bandwidth allocation to IVD channels.

The bandwidth allocated between the P- and C-channels is currently being studied by the IEEE 802.9 committee. The use of fixed and dynamic bandwidth allocations are presently

being considered [MOUG]. In the case of dynamic bandwidth allocation, a temporarily fixed bandwidth must be allocated to the C-channel for the duration of the session. On the other hand, regarding the use of the P-channel, one of the purpose of the IEEE 802.9 committee is to define a MAC layer supporting IEEE 802 LLC (described in the next section) as well as LAPB (link access procedure, balanced) defined in the X.25 protocol [RYBC], and LAPD (link access procedure for the D-channel) defined in the ISDN protocol for sending packets through B- and D-channels [CCITT88]. IEEE 802.2 LLC, LAPB and LAPD are protocols performing more or less similar functions for the recovery of lost frames.

Furthermore, an important part of the proposed IEEE 802.9 standard defines the access to the P-channel. A symmetrical request/grant protocol has been proposed [IEEE89]. This protocol is intended to be established between any single IVDWS and the AU. The idea is that two control bits, periodically exchanged between the two stations implicated in the protocol, are used for managing the sending of MAC frames on the medium (two unidirectional links running in opposite way). The "request" bit is used by a sender (IVDWS or AU) having frames queued for transmission in order to request the right to transmit on the medium. The "grant" bit is used by the destination device for authorizing the sending of frames by the requesting station. Moreover, since the AU concentrates packets from a number of IVDWSs to the IVD LAN, a flow-control mechanism must exist for the P-channel access in order to prevent congestion at the LAN interface. However, this last issue has not yet been defined.

Comments about the 802 MAC and Physical Layer protocols

This concludes the summary of IEEE 802 standards defined or presently proposed at the physical and MAC layers. Since these standards define similar frame structures (see appendix A), it can be relatively simple to interconnect these networks together. On the other hand, when one of these networks must be selected for implementation in a computer communications system, the choice must be based on various requirements. The bandwidth requirements, the types of applications which must be supported, and obviously the budget available to

buy the network are factors which must be considered. DQDB is certainly a network that provide good performance and services, since it is a high speed network supporting circuit and packet switching and it is intended to be used over large areas. However, in some situations where the required network does not have to support heavy load traffic and particular traffic requirements, it may be sufficient to use a CSMA/CD network. If priority classes are required, token-passing schemes will be rather considered (as well as DQDB). If users want to interconnect remote LANs together, then DQDB must be considered. Finally, if users want to integrate voice and data communications, IEEE 802.9 and DQDB represent the logical choice. The next section present the IEEE 802.2 LLC protocol which is compatible with any of the standards previously reviewed.

2.2.6 IEEE 802.2 Logical Link Control

General Overview

The IEEE 802.2 Logical Link Control (LLC) standard [ANSI85A] describes the functions of the top layer of the the Data Link Layer (DLL), and is common to the various medium access methods that are defined and supported by the IEEE 802 activities (see Fig. 2.5). It provides peer-to-peer protocol procedures that are defined for the transfer of information and control between any pair of logical service access points (L-SAPs) on a network (one LLC entity can support one or several L-SAPs). To satisfy a broad range of potential applications, three types of data link control operations are presently defined. The first one, called type 1, provides a data link connectionless service across a data link with a minimum protocol complexity. This type of operation is useful when higher layers provide any essential recovery and sequencing services. In addition, it can prove useful when it is not essential to guarantee the delivery of every data unit. The second type of operation, called type 2, provides a data link connection-oriented, where setting up, maintaining and disconnecting operations of a link are performed. For this type, sequence numbered information frames and supervisory frames are used to provide acknowledgement, retransmission and flow control actions. A

window mechanism is also defined to detect and recover lost data units. A third type of operation, called type 3, has been recently proposed [ISO]. This type of operation is also called "Acknowledgement Connectionless Service", since it provides acknowledgement with minimum protocol complexity. In the next sections, the LLC PDU frame format is presented and a detailed description of the three LLC types of operation is done.

LLC PDU Frame Format and Functions

Figures 2.11 and 2.12 show respectively the frame format and control field of the LLC PDU. The LLC PDU frame contains the DSAP (destination SAP) address, the SSAP (source SAP) address, the control, and the information fields. The address fields identify the one or several SAPs for which the LLC information field is intended (7 bits for 128 L-SAPs). Three control fields are defined to identify three types of frames (Fig. 2.12): I-frames supporting numbered information transfer; S-frames supporting numbered supervisory transfer; and U-frames supporting unnumbered information transfer. The I- and S-frames are only used by the LLC type 2 operation while the U-frames can be used by all three types of operations. The I-frame control field, in order to identify numbered information transfer, contains a send sequence number N(S). Both I- and S-frames also contain a receive sequence number N(R) which is interpreted by the receiving LLC as being the expected number of the next received I-frame by the remote LLC. The S-frame shall be used to perform data link supervisory and control functions. In fact, there are three types of S-frames distinguished by the SS bits: RR-frame (receive ready), RNR-frame (receive not ready), and REJ-frame (reject). In each case, N(R) acknowledges all I-frames up to and including N(R)-1. The RNR-frame also provides the flow control for a temporary suspension of I-frame transmissions while the REJ-frame contains a N(R) which rejects all I-frames from N(R). The U-frame is used for initialization and termination of a data link connection (for LLC type 2), and for transferring unnumbered information and diverse control messages (for type 1 and 3). Its control field is shorter than the other ones since it does not contain any sequence number.

Figure 2.11: Logical Link Control Protocol Data Unit (LLC PDU) frame format.

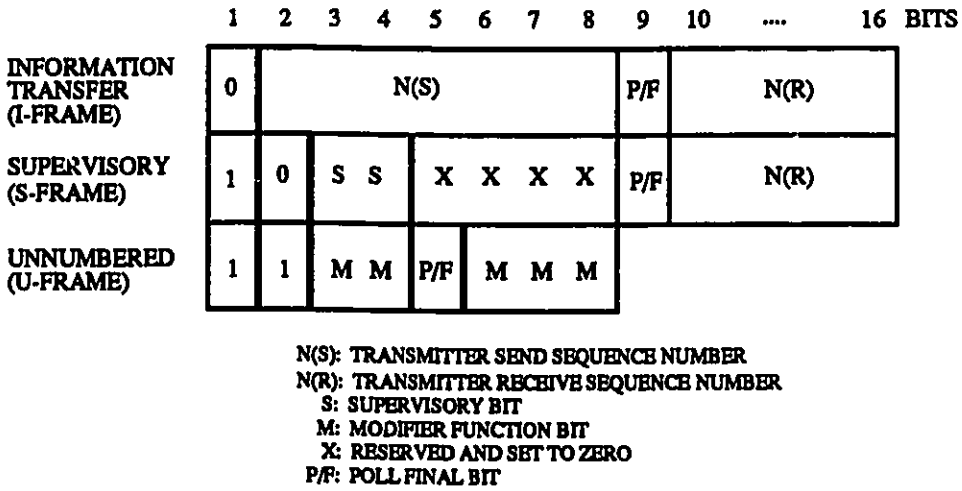


Figure 2.12: LLC PDU control field.

In each control field, there is also the P/F (poll/final) bit. The poll (P) bit shall be used to solicit (poll) a response from a remote LLC. Then the final (F) bit shall be used to indicate the response frame sent as a result of the soliciting (poll) command. This polling/response function of the P/F bit can be used for checking if the data link is operating properly, for forcing an early acknowledgement of an I-frame or an early REJ-frame, or for preparing to take a link down. A timer, called the P-bit timer, is started when a frame is sent with the P-bit set to 1. The P-bit timer recovery conditions shall be cleared when the LLC receives a valid frame from the remote LLC with the F-bit set to 1. Else, if the P-bit timer expires out, the LLC should resend a frame with the P-bit set to 1 and restart the P-bit timer. This can be repeated until a given time, after which a transmission error is reported to the higher layers.

LLC Types of Service

This section summarizes the protocols characterizing each of the three types of operations defined by the IEEE 802.2 committee.

The **LLC type 1** provides less overhead connectionless information facility than type 2. For this purpose, three U-frames are defined: unnumbered information (UI) frames and two management frames, called XID (exchange information) and TEST (for link testing) frames. Since the type 1 defines the transfers of information over UI frames, without the use of send sequence number, and does not provide particular frames for acknowledging information frames, there is neither a flow control mechanism nor a recovery scheme performed.

The **LLC type 2** provides connection-oriented information transfer for two LLC entities. Compared to LLC type 1, a logical link must be established between logical SAPs before exchanging I-frames. By using N(S) and N(R) in I- and S-frames, the type 2 protocol can provide an in-sequence information transfer without loss or duplication of data. As previously mentioned, S-frames are used for acknowledgement, retransmission and flow control purposes. Besides, the "Go back n" protocol [BERT] is used as a retransmission and flow control strategy.

The idea of the "Go back n" protocol is that the sending station can send up to a number of n I-frames without having to wait for the acknowledgement of the first sent I-frame. The expression "window of size n" is used to represent the physical environment containing the unacknowledged frames. When the first frame in the window is acknowledged, the window is slided by one, and another frame can be sent. By using the sequence number N(S) with a modulus M ($M = 128$ for 7 bits field), a window of at most M minus 1 frames can be employed for preventing any ambiguity in the association of sent I-frames with sequence numbers during normal operation and/or error recovery action. The case where $n = 1$ is called the Stop-and-Wait protocol [BERT].

Figure. 2.13 presents an example of "Go back n" protocol where $n = 4$. An example of the use of REJ-frame is also depicted. The REJ-frame forces the retransmission of I-frames having a send sequence number equal and greater to N(R) (N(R) = 2 for the example). The send state variable V(S) denotes the sequence number of the next in-sequence I-frame to be sent, and the receive state variable V(R) denotes the sequence number of the next in-sequence I-frame to be received. These variables shall take on a value between 0 and M minus 1. V(S) is incremented by one with each successive I-frame transmission, but shall not exceed N(R) of the last received I- or S-frame by more than the window size n minus one. However, V(S) may be set to the value contained in N(R) if a REJ-frame is received (see example in Fig. 2.13). V(R) is incremented by one whenever an error-free in sequence I-frame is received whose send sequence number N(S) equals V(R).

The LLC type 2 uses four timers:

- *acknowledge timer*: used to define the time interval during which the LLC shall wait for an acknowledgement of an I-frame.
- *P-bit timer*: used to define the time interval during which the LLC shall wait for a reply to a previous poll command.
- *REJ-timer*: used to define the time interval during which the LLC shall expect to receive

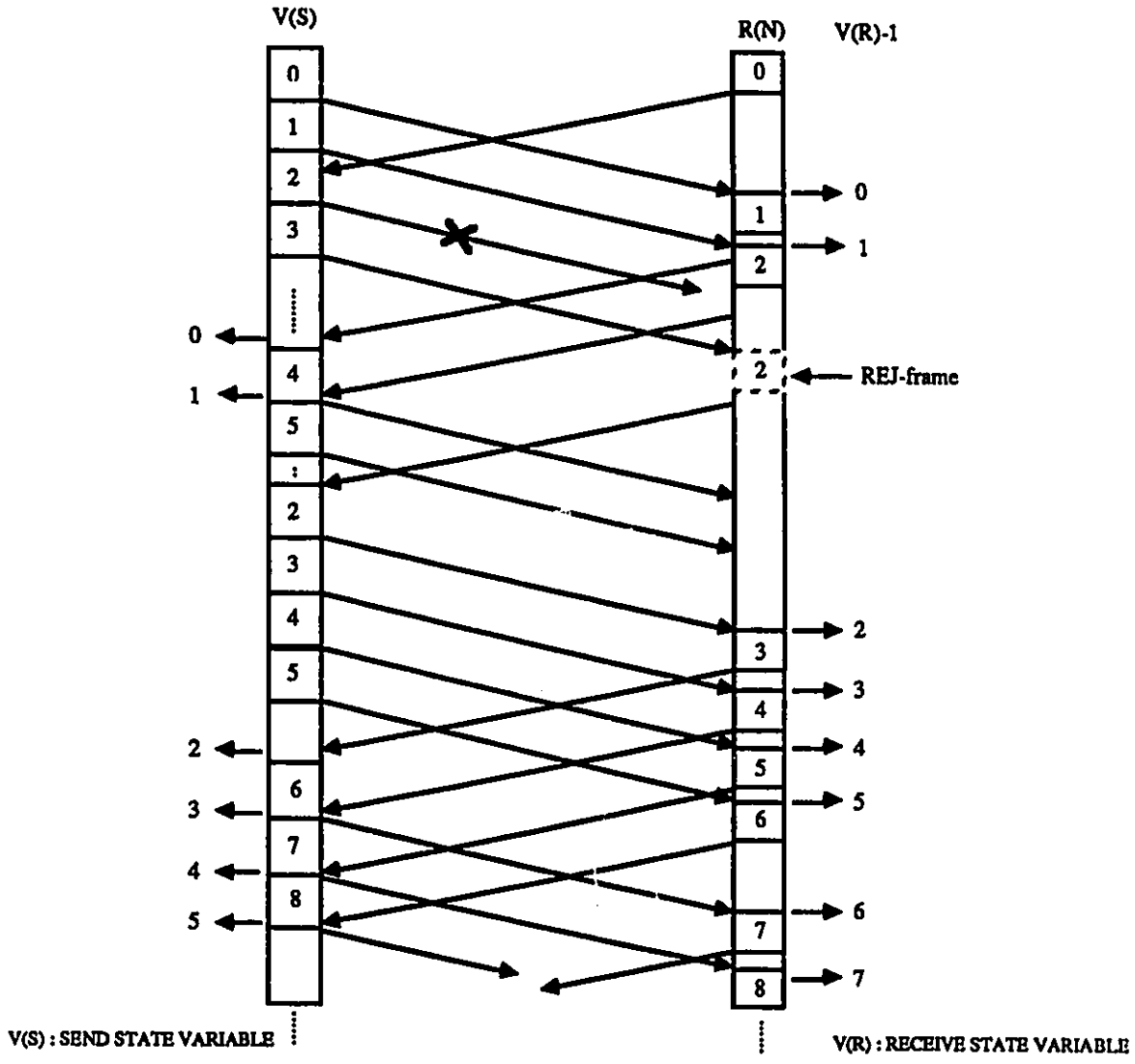


Figure 2.13: Example of the "Go back n" protocol with $n = 4$.

a reply to the sent REJ-frame.

- *busy-state timer*: used to define the time interval during which the LLC shall wait for an indication of the clearance of a busy condition at the remote LLC (used when the LLC has received a RNR-frame).

Finally, the **LLC type 3** has been recently proposed [ISO] and it is intended to provide a "send one frame, receive one acknowledgement" service [FIELD] for applications that require acknowledgement of data unit transfer but wish to avoid the complexity of a full connection service like the one offered by the LLC type 2. The LLC type 3 uses two U-frames: AC0 and AC1 (acknowledgement connectionless information sequence 0 or 1). For instance, if a sending LLC entity sends an AC0 (AC1) frame with an information field, the remote LLC entity acknowledges its receipt by means of an AC0 (AC1) with a null service data unit. If an acknowledgement is not received within a timeout period, the AC0 (AC1) frame is retransmitted. Another frame cannot be sent until the first one has been acknowledged. The sending LLC alternates between AC0 and AC1 frame on new transmission to permit the remote entity to distinguish between retransmission and transmission of the next frame. This scheme is similar to the of the Stop-and-Wait protocol (window size n equal to 1).

2.3 Integrated Services Digital Networks

In the sections 2.2.4 and 2.2.5 where the proposed IEEE 802.6 DQDB MAN and IEEE 802.9 IVD LAN interface standards have been presented, the existence of the ISDN (Integrated Services Digital Network) and the B-ISDN (Broadband ISDN) standards have been mentioned. ISDN involves a set of standards whose main objective is the end-to-end digital connectivity between end user. It offers bit rate from 64 kbps up to 600 Mbps. In particular, N-ISDN (Narrowband ISDN) and B-ISDN have been proposed. N-ISDN is intended to provide bit rates up to 1.536 Mbps (2.048 Mbps in Europe) while B-ISDN is intended to provide 150/600 Mbps link per user for supporting broadband applications such as video communications. In

the next sections, the major concepts of ISDN and B-ISDN standards are reviewed.

2.3.1 ISDN

ISDN is a data communications network that is digital from one end to the other, i.e end-to-end digital connectivity [ANDR]. It is intended to carry all types of information including digitized voice, data, and image. The architecture of a potential ISDN network is briefly given in Fig. 2.14. On the ISDN access side, there is the network terminator (NT) which corresponds to the layer 1 of the OSI Reference Model (line transmission termination, interface termination) [JULIO]. It can be accessed by one or several terminal equipments (TEs) through a passive bus. A large number of NTs can be physically attached in a full duplex way to the ISDN switch. On the network side, the ISDN switches can be interconnected through packet and/or circuit switched networks or networks composed of ISDN switches. In fact, similarly to the IEEE 802.9 standard, each end user can be independent of the transport network by using an ISDN switch as a network interface.

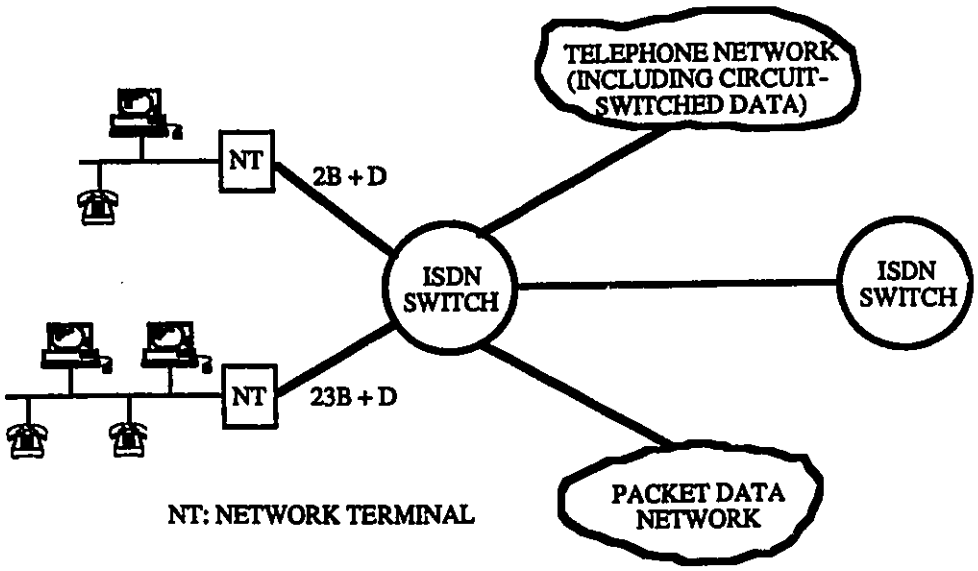


Figure 2.14: A potential ISDN architecture.

Two ISDN interfaces are presently defined: the basic rate interface, composed of 2 B-channels (64 kbps/channel) plus one D-channel (16 kbps), for an overall data rate of 144 kbps; the primary rate interface, composed of 23 B-channels and one D-channel of 64 kbps for an overall data rate of 1.536 Mbps (30B + D in Europe). The D-channel is mainly used for transferring control signaling in order to establish communications between end users while the B-channel supports circuit and packet switching communications. The ISDN standards also define the H-channels which have a bandwidth in multiples of 64 kbps. The characteristics of each channel are summarized in Table 2.2. Hence, these channels represent the types of bandwidth offered to each NT connected to an ISDN switch.

Channel type	*Bandwidth (kbps)*	*Purpose*
D	16 or 64 kbps	carry signaling information plus packet switching
B	64 kbps	circuit and packet switching
H0	384 kbps	
H11	1.536 Mbps	
H12	1.920 Mbps	
H21	32.8 Mbps	
H22	44 Mbps	
H22	70 Mpbs	
H3	140 Mbps	

Table 2.2: ISDN channel types.

Figure 2.15 shows the layered protocol structure at the ISDN user-network interface (ISDN switch). At the layer 2, the LAPD (Link Access Protocol for the D channel) has been proposed to support packet switching communications over the D, B and H channels [CCITT88]. This protocol provides error and loss recovery as well as core functions and multiplexing of several connections coming from the layer 3. Related to LAPD, the Study Group XVIII has recently proposed frame relaying services [CCITT88]. The main purpose of using frame relaying is to reduce the frame processing in intermediate nodes (interconnecting switches providing services to end users) by limiting the functions of LAPD to the ones of core services (see Fig. 2.16 and also [LAI]). Thus, error and loss recovery is only performed at the end nodes. With end user

performing these functions end-to-end, there is lesser or no need for a network layer to do the same. Frame processing at the network layer can then be considerably reduced. Finally, at the layer 3 of the ISDN user-network interface (UNI) layered structure (Fig. 2.15), the Q.931 recommendation is defined for providing establishment, maintenance and clearance of network connections at the ISDN UNI [KANO].

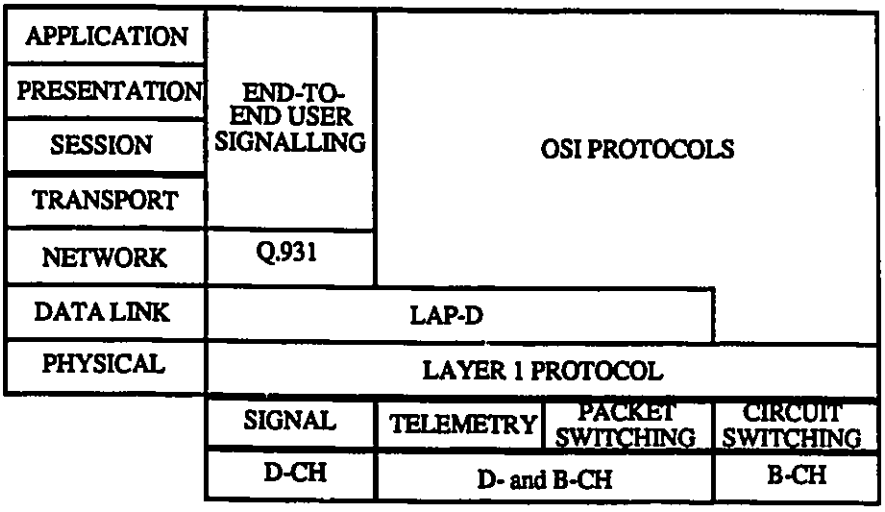


Figure 2.15: Layered protocol structure at the ISDN user-network interface.

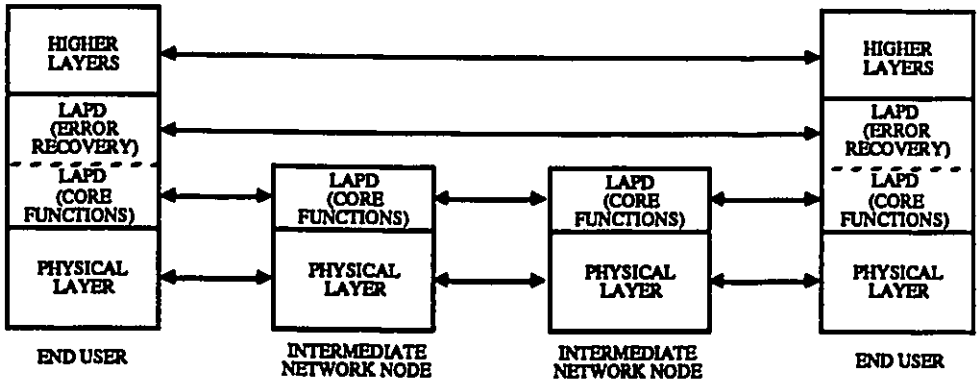


Figure 2.16: Data transfer procedures in frame relaying links.

2.3.2 Broadband ISDN

With the introduction of new services requiring large bandwidth such as high speed data packet (high volume file transfer services) and motion video services (broadcast/multipoint video, video telephony, video conference), there is a need to define a broadband network able to integrate these services requiring bandwidth higher than 2 Mbps. Thus, the CCITT Study Group XVIII [CCITT89] is defining the B-ISDN standard which is intended to provide 150/600 Mbps interface to end users or customer premises networks (see fig 2.17). All types of services will be packetized at the terminal adapter (TA), and multiplexed at the user-network interface (UNI), and then transferred through a route comprising a given number of switching nodes connecting the source and destination UNIs.

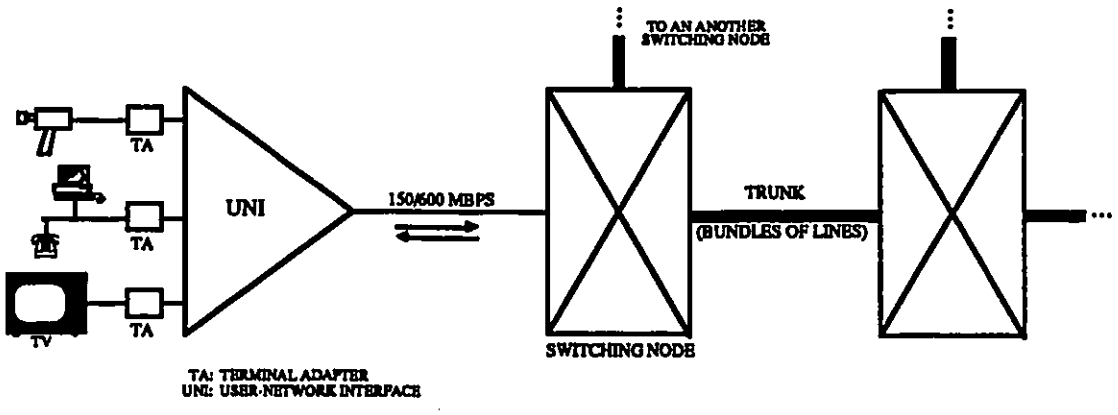


Figure 2.17: B-ISDN architecture.

The CCITT Study Group XVIII has adopted the ATM (Asynchronous Transfer Mode) concept [RYDER,THOM] as transport mechanism for B-ISDN. In ATM, all information is conveyed within fixed-size slots called cells. Figure 2.18 illustrates the structure of the ATM cell. This is quite similar to the structure of the cells (slots) defined by the IEEE 802.6 DQDB MAN standard. Each ATM cell contains in its header a VCI (virtual channel identifier) which allows the switching nodes to identify and switch the cell through the network. The VCI is determined at the virtual connection establishment. The term "asynchronous"

is used since ATM is a packet switching based network. ATM cells are filled depending on the actual demand from each connection, which means that cells used by a given connection can exhibit an irregular recurrence pattern. ATM is then different than schemes like STM (Synchronous Transfer Mode) where dedicated bandwidth is preallocated for each connection (circuit switching concept). Thus, the major advantage of ATM is flexibility, since any kind of traffic, already or eventually defined, will be integrated. For instance, ATM will be advantageous to support bursty traffic (such as compressed video) which is characterized by important variations of the bit rate generated during the maintenance of a virtual connection.

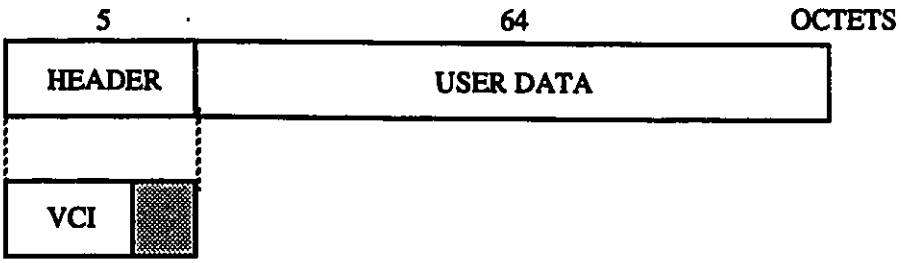


Figure 2.18: A potential ATM cell structure.

2.4 Concluding Remarks

Figure 2.19 depicts an example of a typical scenario of a future public telecommunications network interconnecting public and private networks. Indeed, B-ISDN will be used as a large public network interconnecting various types of LAN, integrated voice/data interfaces, and DQDB based networks. DQDB will be used as concentrator for regrouping LANs as well as integrated voice/data interfaces. This mainly explains why the IEEE 802.6 committee has adopted the same slot length and structure as the cell presently defined for the ATM protocol transferring information through B-ISDN.

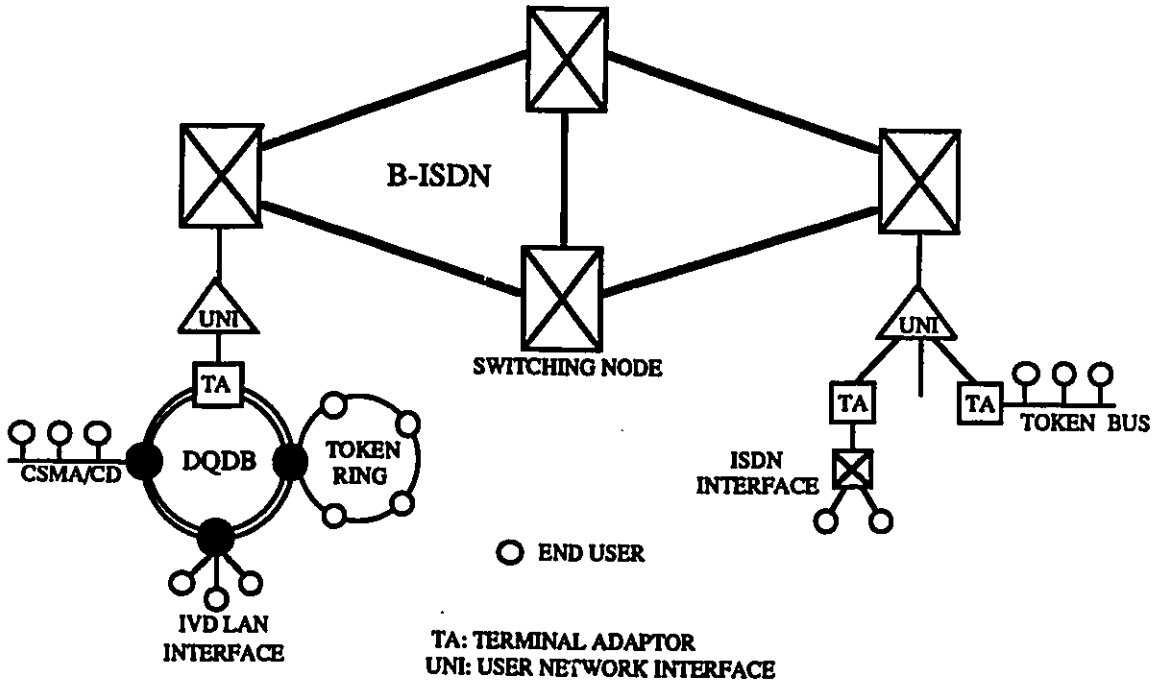


Figure 2.19: Future telecommunication network mainly supported by B-ISDN.

Chapter 3

IEEE 802.6 DQDB MAN

This chapter presents the proposed IEEE 802.6 DQDB MAN which is in the process of being standardized [IEEE88A,IEEE88B]. An overview of the evolution of the proposal is first presented. Then the architecture, the timing structure and the access protocols of DQDB are described. Finally, the performance of DQDB for supporting packet switched communications are discussed and analyzed. This includes a proposal for a simulation model of DQDB performing the allocation of asynchronous bandwidth.

3.1 Evolution of DQDB

3.1.1 QPSX: The precursor of DQDB

In 1986, QPSX (Queue Packet and Synchronous Circuit Exchange) was proposed by a research group of the University of Western Australia [BUD86,NEW86,NEW88B]. The idea of QPSX is mainly based on previous work done by Budrikis on an integrated data/voice switch [BUD84] as well as on the concepts defining the Fastnet network [LIMB]. In 1987, with the support of Telecom Australia, a company, named QPSX Communications Pty. Ltd, was founded to develop QPSX products. Starting in 1990, pre-commercial pilot networks are to be implemented to demonstrate the full capabilities of QPSX technology and its scope for applications [SCOTT].

3.1.2 DQDB as an IEEE 802.6 Proposal

In November 1987, the IEEE 802.6 Working Group, responsible for standardizing a MAN (Metropolitan Area Network), adopted the QPSX concepts under the name DQDB (Distributed Queue Dual Buses) [IEEE87B]. They made that choice, since QPSX allows the integration of voice and data, provides high speed data transmissions, and can be deployed over a large area (metropolitan area) such as a city or suburbs. Since then, some modifications to the initial proposal has been done (slot size and structure) but the major ideas concerning the access control mechanisms of QPSX have been retained. The proposed IEEE 802.6 standard is to be ratified and sent to IEEE for formal approval sometime during 1989.

3.2 The Architecture of DQDB

As mentioned in chapters 1 and 2, the architecture of DQDB comprises two unidirectional contra buses which can be joined to form looped buses (see Fig. 3.1 and 3.2). Each node of the network is attached to both buses, allowing full duplex communications between every pair of nodes. Fixed-length slots, recurring periodically in implicit frames of 125 μs and shared between isochronous and non isochronous communications, are generated by the head of each bus. Two types of slots are defined: NA (non-arbitrated) slots for supporting isochronous communications and QA (queue-arbitrated) slots for supporting non isochronous communications. After being generated by the head of bus, slots are passed successively to each node where they may be modified. They are later destroyed at the end of the bus. In the case where buses are joined to form a loop, both heads of bus are located in the same device. The major advantage of using a looped bus topology appears when there is a link failure in the system. Each node adjacent to the link failure becomes temporary head of bus while the true heads of bus will bypass slots to allow the restoration of the communications between nodes situated at each side on the link failure (Fig. 3.3).

Nodes attach to the bus via a *Write* connection and a *Read Tap* placed ahead of the *Write*

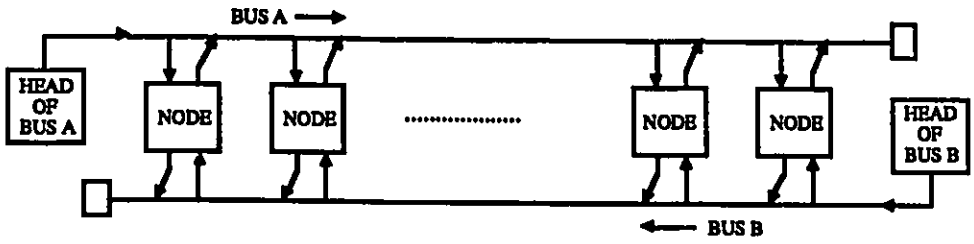


Figure 3.1: IEEE 802.6 DQDB MAN open bus topology.

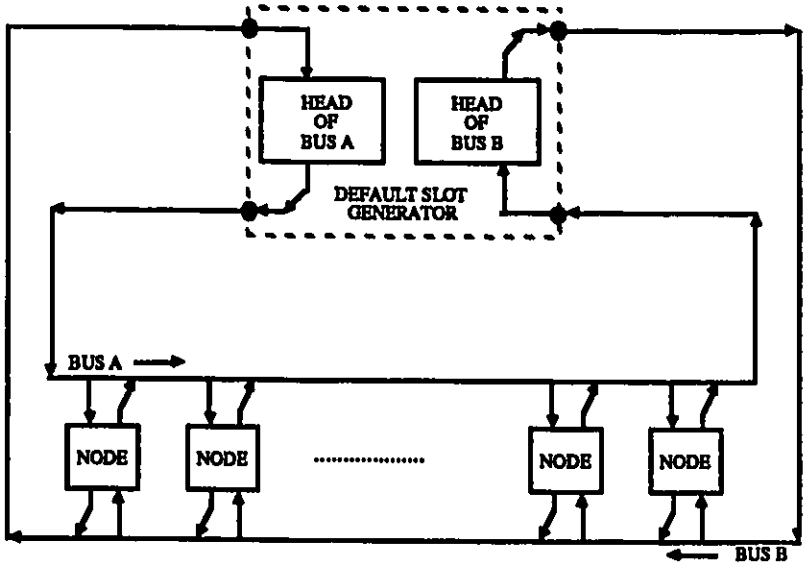


Figure 3.2: IEEE 802.6 DQDB MAN looped bus topology.

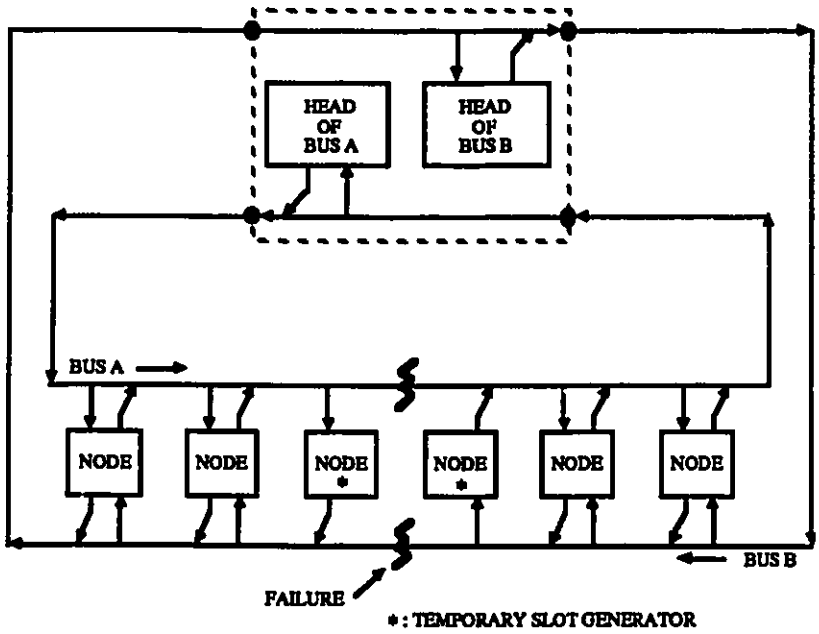


Figure 3.3: IEEE 802.6 DQDB MAN looped bus topology under healed conditions.

connection (see Fig. 3.4). The writing on to the bus is by logical OR of the data from upstream with the data from the node. The read connection allows all data to be copied from the bus unaffected by the node's own writing.

Figure 3.5 shows the internal structure of the DQDB MAC and physical layers. At the upper part of the structure, there are the entities responsible for managing the transfer of data through DQDB. Regarding the management of the transfers of asynchronous data, there is first the ATC (asynchronous transfer control) which is intended to provide an asynchronous connection-oriented service for data transfer to all ATSUs (asynchronous transfer service users) over QA slots. The ATC is also responsible for the mapping of connection end-points and VCIs (virtual channel identifiers) contained in each QA slot header. The MCP (MAC convergence protocol) is a particular ATSU defined to support connectionless MAC service which is consistent with the MAC services defined by other IEEE 802 standards. For instance, the MCP must perform segmentation and reassembly functions in order to adapt the service

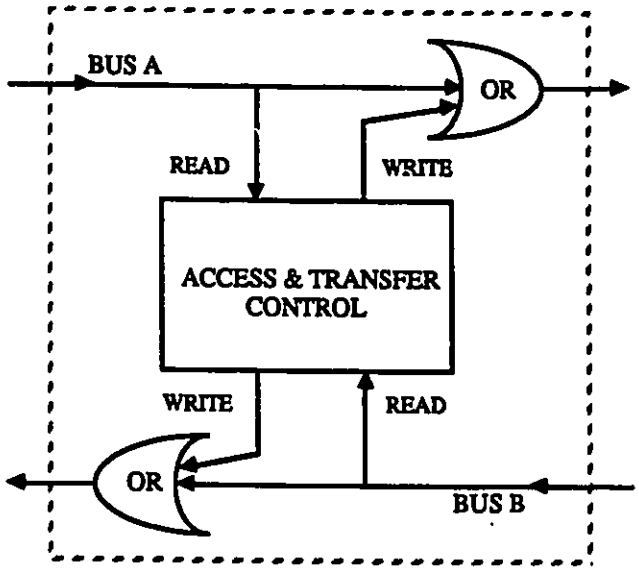


Figure 3.4: DQDB node architecture.

offered by the ATC. Under the ATC, there is the queue-arbitrated access control (QA_AC) which is responsible to control the access of the QA slots to the bus. The QA_AC will be described in more details in section 3.4.2.

In order to provide services for supporting isochronous channels over an isochronous connection, the ICP (isochronous convergence protocol) and the ITC (isochronous transfer control) are defined. The ICP is a smoothing function matching the isochronous rate required by the isochronous service user to the actual rate offered by the ITC entity (the ICP can simply be a buffer). The ITC is responsible for controlling the transfers of isochronous data between the ICP and the non-arbitrated access control (NA_AC). Indeed, when the NA_AC receives a NA slot, it extracts the contents of the slot (VCI and isochronous octets) and sends it to the ITC. If the VCI maps with some connection end-points supported by the ITC, the ITC will read and/or insert octets from and/or to the appropriate octet positions in the NA segment payload. Then, the NA segment payload is returned to the NA_AC in order to be reinserted in the NA slot and transmitted to the next node.

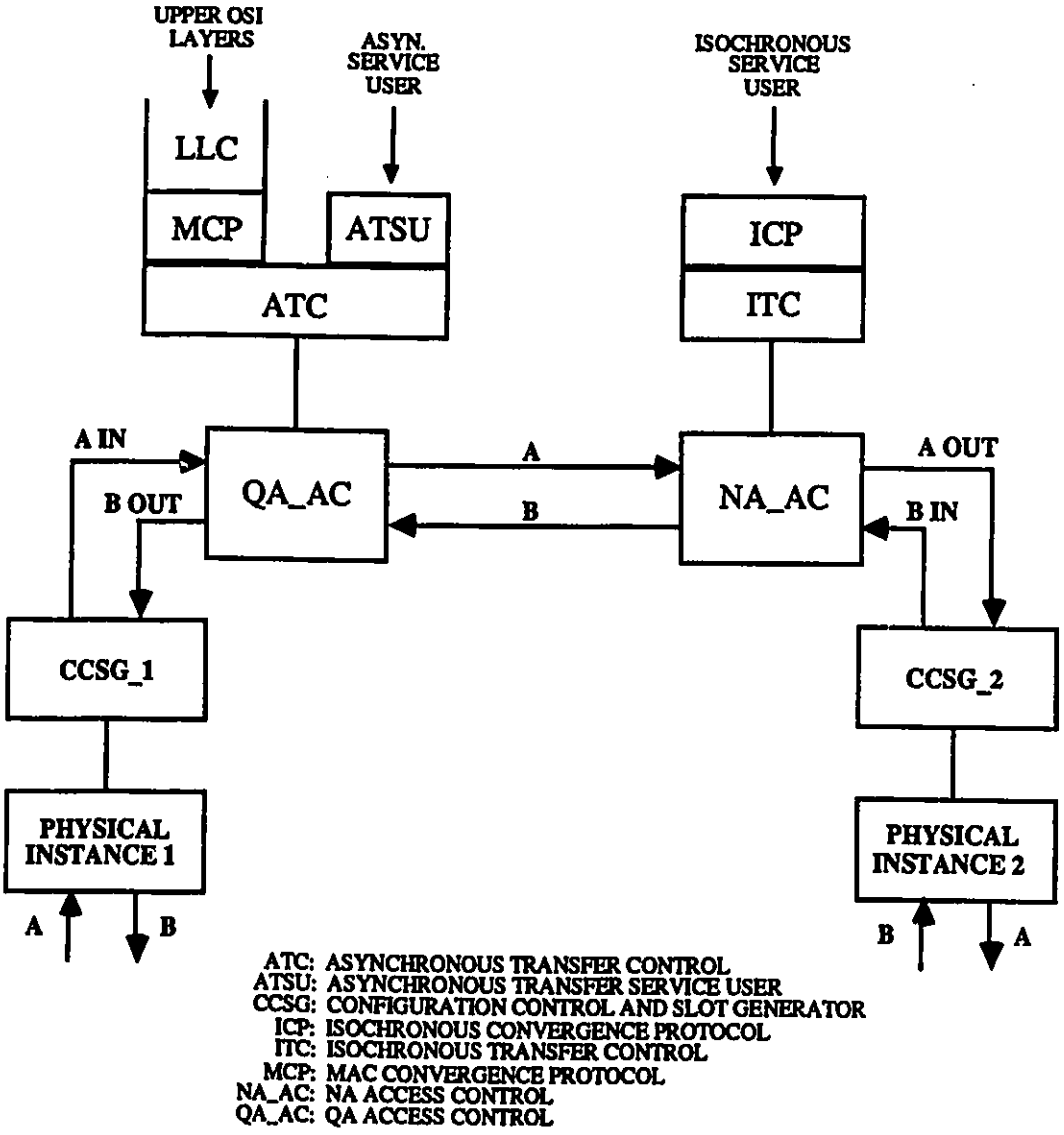


Figure 3.5: DQDB MAC and physical layer structure.

In the layered structure presented in Fig. 3.5, there is a configuration control and slot generator (CCSG), at the physical layers, which is mainly defined for allowing any node to perform temporary slot generation upon occurrence of a link failure or any other faults within the network. Under normal operation, the CCSGs situated in the head of bus are responsible for generating slots and providing time synchronization.

3.3 Timing Structure in DQDB

The timing structure of DQDB is based on implicit frames of 125 μs comprising a given number of fixed slots shared between isochronous and non isochronous communications (Fig. 3.6). Circuit switching through NA slots, and packet switching through QA slots, can then be provided. The network capacity is dynamically allocated to meet the demand of the isochronous traffic. Capacity that is not allocated to isochronous traffic can be shared by the packet switched traffic. Slots of 69 octets are presently defined in order to facilitate the interconnection of DQDB with B-ISDN [IEEE88B,MOLL]. Hence the DQDB bandwidth is fixed by the number of slots in the frame. Each slot insertion brings an additional bandwidth of 4.416 Mbps (69 bytes/125 μs). However, bit rates of 45 Mbps for DS3 (digital signal number 3) circuits and 155 Mbps for SONET (Synchronous Optical Network) circuits seem to be candidates [MOLL].

The first octet of each slot is the access control field (ACF) used for slot identification and access control. The ACF contains the BUSY bit for indicating if the slot is used, the TYPE bit for indicating if the slot is a NA or QA slot, the PSR (previous segment received) bit whose use is for further study by the IEEE 802.6 working group, and finally the four REQ (request) bits which are used in the operation of the four priority level distributed queues. The use of the ACF field will be explained in more details in section 3.4.2.

The next four bytes of each slot corresponds to the segment header. This comprises the virtual channel identifier (VCI) representing a logical link between two MAC entities, some

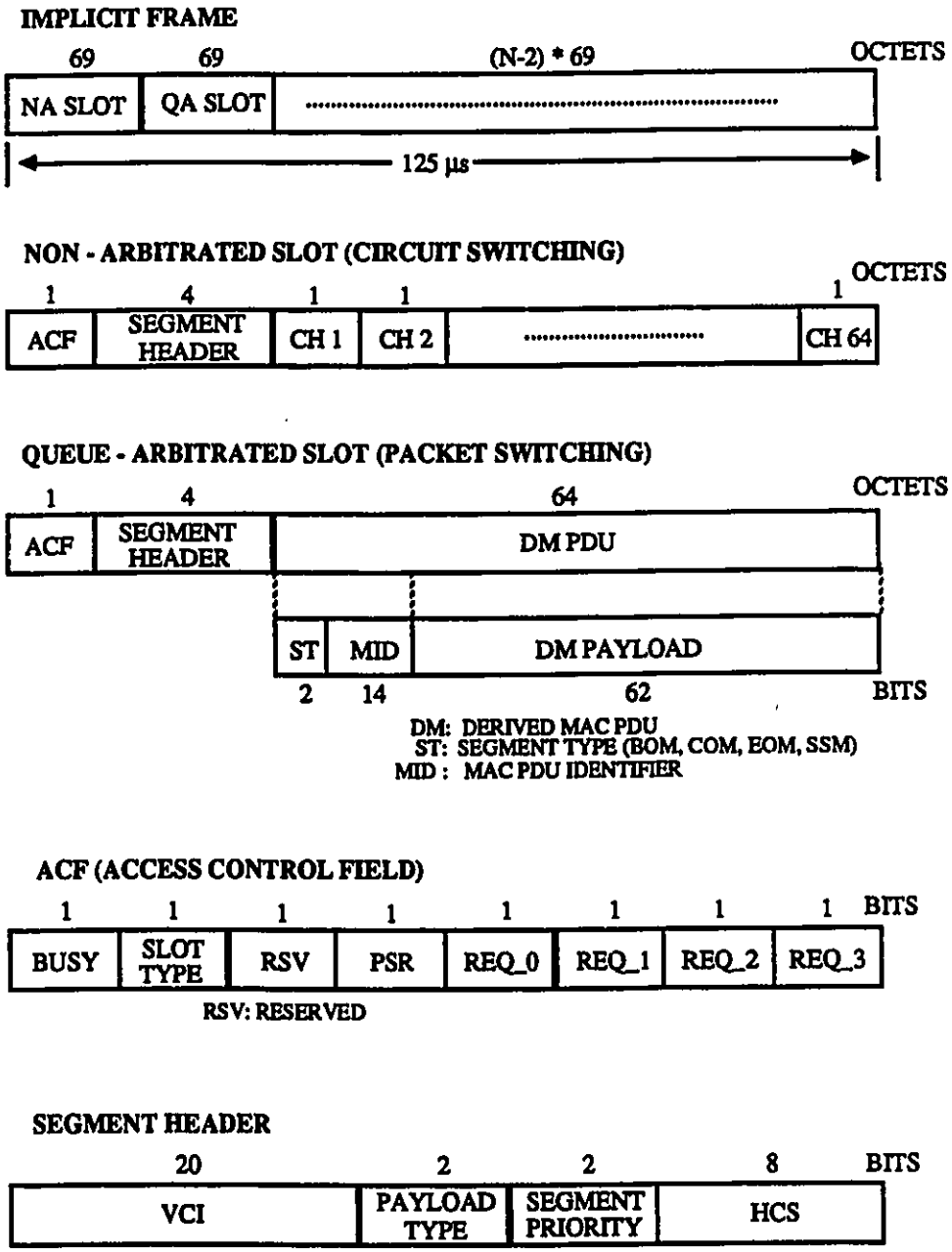


Figure 3.6: Timing structure of DQDB.

bits used for protocol service identifications, and one byte containing the segment header check sequence (HCS).

Every NA slot consists of 64 octets where each of them corresponds to a voice channel ($\frac{8\ bits}{125\mu s} = 64\ kbps$). The isochronous channels are allocated to nodes by the head of the bus. In fact, each node must send a request to the head of bus to have access to one or several channels. Each isochronous channel is identified by an implicit sequence number reflecting its position in the implicit frame of 125 μs. Thus the i^{th} NA slot in the frame contains the octets with identification numbers $(i-1)L + iL$ where L denotes the number of isochronous channels in each NA slot.

QA slots are used to transfer asynchronous segments previously generated by a longer message segmentation. Of the 69 octets that comprise a QA slot, 64 are used to convey each derived MAC PDU (DM PDU) generated at the MCP (MAC convergence protocol) following a MAC PDU segmentation. Each DM PDU comprises a header which contains bits for identifying the DM PDU type and the initial MAC PDU. In fact, there are 4 types of DM PDU: the beginning of MAC PDU (BOM), the continuation of MAC PDU (COM), the end of MAC PDU (EOM), and the single segment MAC PDU (SSM) which conveys MAC PDUs whose length is shorter than the maximum segment payload length (62 octets). Figure 3.7 shows an example of MAC PDU which must be segmented into several DM PDUs in order to be transferred through DQDB. A detailed description of the DQDB MAC PDU is given in appendix A, section A.4.

3.4 Access Protocols in DQDB

The IEEE 802.6 DQDB MAN standard defines two access control mechanisms for supporting circuit and packet switching services: isochronous octets access control in NA slots and QA slot access control. Both access controls are independently executed.

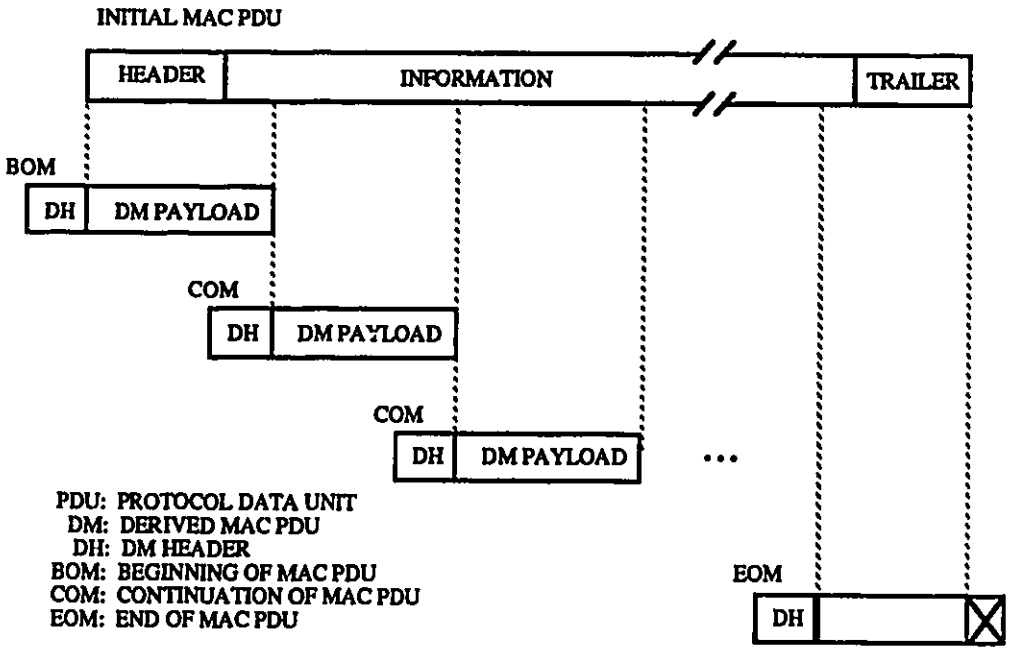


Figure 3.7: MAC PDU segmentation performed by the MAC Convergence Protocol (MCP).

3.4.1 Isochronous Octets Access Control in NA Slots

The allocation of isochronous octets in NA slots is performed by centralized protocol procedures situated in the heads of bus. Stations have to send a circuit request to the head of the bus on which they have to send isochronous information. On request receipt, the head of bus verifies if the request demand can be satisfied among the empty isochronous channels available in the NA slots already defined. If not, the head of bus defines a new NA slot in order to satisfy the request. The requesting isochronous channels is then informed of the NA slot (identified by a VCI) and the isochronous octets in the slot it can use. The circuit allocation policy which is currently applied in QPSX is the *First-Fit* algorithm [ZUK87] (for DQDB, this is not defined since this action is assumed to be performed by the upper layers). Under this policy, for each incoming isochronous octet request, the head of bus assigns the octet with the smallest identification number among all the empty octets in the NA slots.

3.4.2 QA Slot Access Control

The allocation of QA slots is performed by a distributed queueing protocol established between all nodes connected to the buses. The operation of this protocol is based on two control bits contained in the ACF, the BUSY bit which indicates whether the slot is used and the REQ (request) bit which is sent whenever a node has an asynchronous segment waiting for access to one of the two buses. Figures 3.8 and 3.9 show the scenario where the access protocol is used to transmit segments onto the bus A. During the idle state (when there is no segment queued at the access node), only the REQ counter is used: it is incremented for each REQ bit received on bus B, and decremented for each non busy QA slot that passes by the station on bus A. Hence, each node can determine the number of downstream stations which have a segment queued to send on bus A. When the station has a segment to send on bus A, it moves to the countdown state (Fig. 3.9 and 3.10). The value of the REQ counter is passed to the countdown (CD) counter, and the REQ counter is cleared. At the same time, the node must set the next empty REQ bit passing on bus B to 1 in order to inform the stations ahead that it has a QA slot queued to send on bus A. During the countdown state, the REQ counter is incremented for each BUSY bit received on bus B, and the countdown counter is decremented for each non busy QA slot passing on bus A. This counting establishes a single ordered queue across the network for access to the bus. The segment will gain access as soon as capacity becomes available. Once the countdown counter reaches zero, the station sends the queued segment in the next non busy QA slot. While in the countdown state, the station uses the REQ counter for keeping track of the number of downstream stations which enters in the distributed queue after it. In addition to the idle and countdown states, the standby state is also defined. Receiving a segment for transmission, the station enters in standby mode if its REQ counter is equal to zero. If the next QA slot passing on bus A is empty then the segment is immediately transmitted, else the station must move to the countdown state.

The four REQ bits comprised in the ACF (Fig. 3.6) may be used to handle up to 4 levels of priority (0, 1, 2, and 3, 3 being the highest). Thus, in each node, four access priority queues

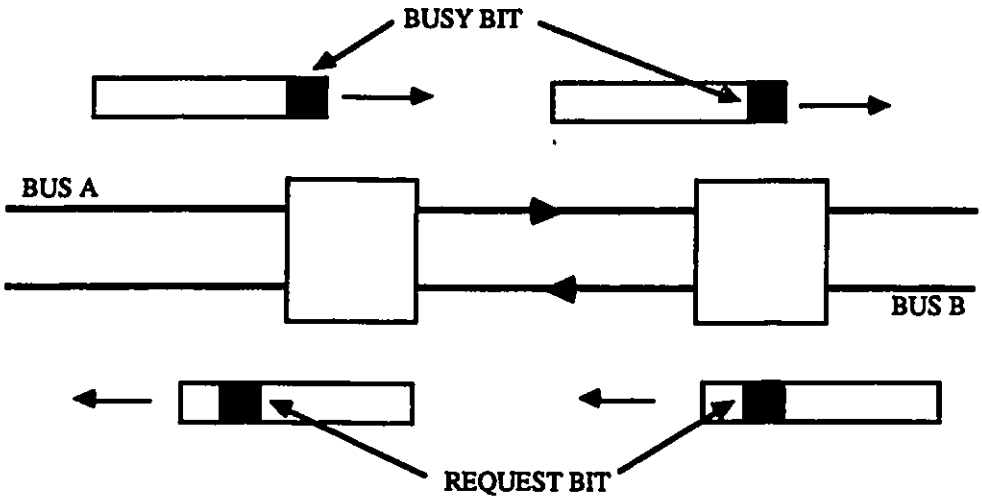
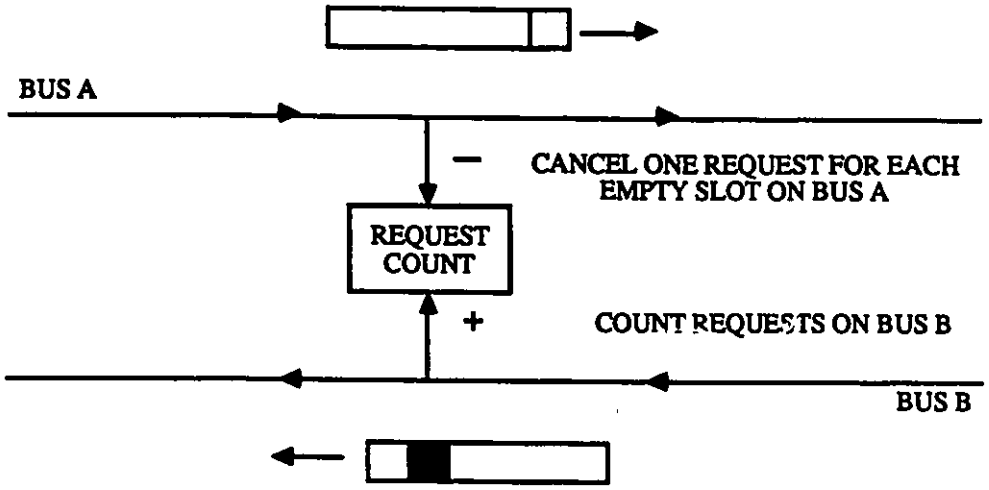


Figure 3.8: Elements used for the queue formation on bus A.

per bus can be defined where each access queue (AQ) is associated with a REQ counter and a countdown counter which can be now incremented or decremented. Hence segments waiting in higher access priority queues are transmitted before the others. Figure 3.10 shows the distributed queue state machine of the access queue at priority level I (AQ_I) performing the sending of segments on bus A. During the idle state, the REQ counter for priority level I (RQ_I) is decremented for each empty QA slot passing on bus A, incremented each time a REQ bit having a priority equal or greater than I passes on bus B, and incremented each time another AQ_J (with J > I) within the node receives a segment to transmit on bus A. For the last event, a SELF REQ bit, having a priority J, is generated by the AQ_J for informing the other AQs within the same node. The AQ_I remains in idle state until it receives a segment for transmission. Upon segment receipt, the AQ_I enters in standby state if the RQ_0 is equal to zero or if the AQ_0 has no segment queued (else the AQ_I enters in countdown state). Having entered in standby state, the AQ_I must send the queued segment in the next QA slot (if this slot is empty), else it will move to the countdown state. Moreover, if during the time the AQ_I is in standby state, a REQ bit passes on bus B or a SELF REQ bit is generated within the node, then the AQ_I will enter in countdown state. In countdown state, as well as in standby

A) NODE NOT QUEUED TO SEND (IDLE STATE)



B) NODE QUEUED TO SEND (COUNTDOWN STATE)

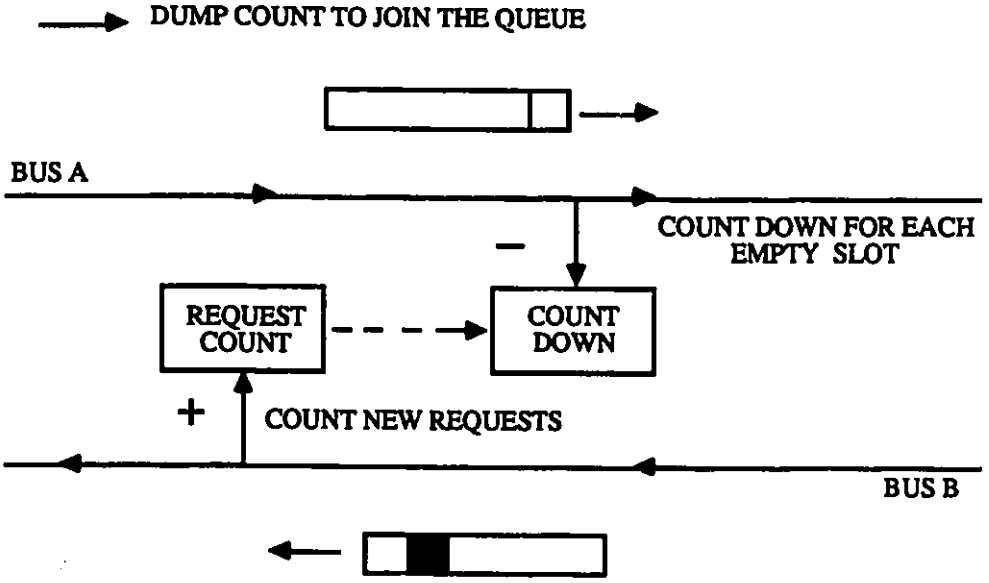


Figure 3.9: a) Node in idle state b) Node in countdown state.

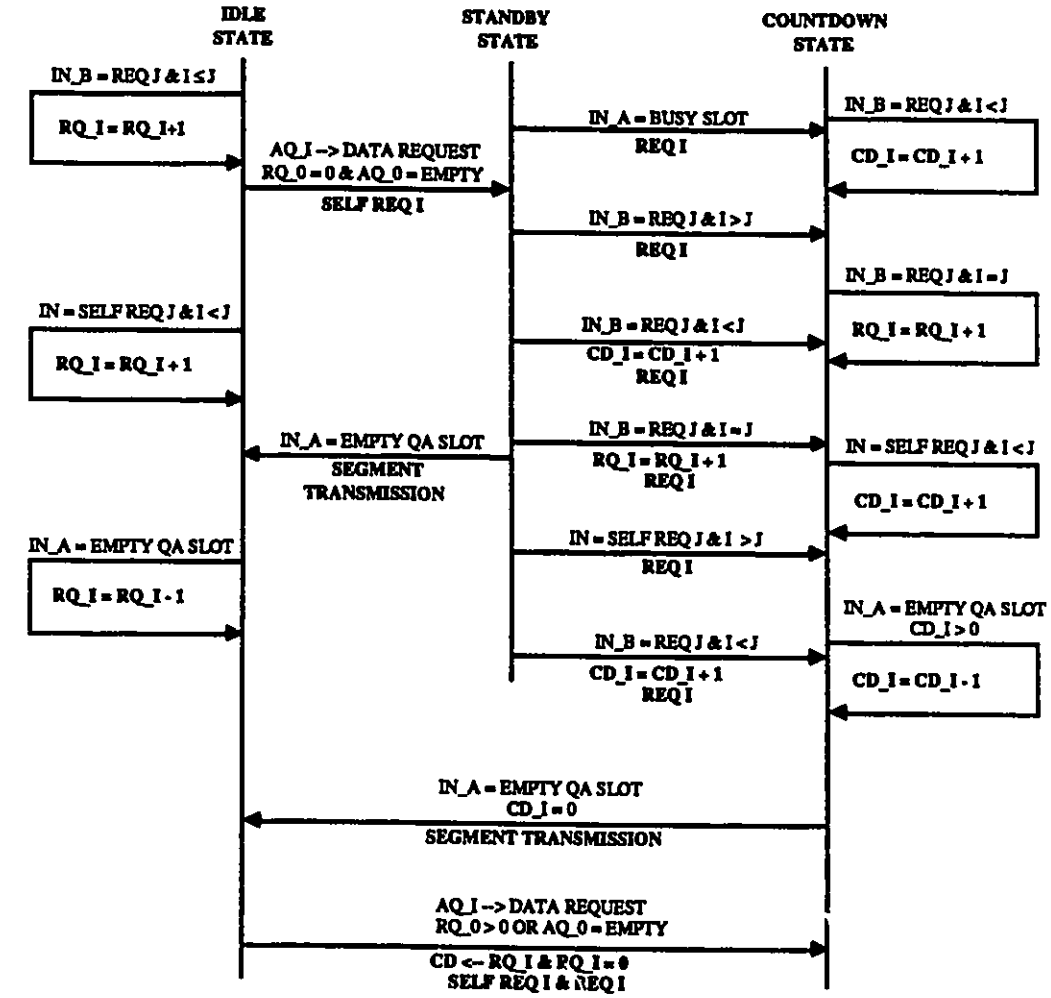
state, the RQ_I is incremented each time a REQ bit, having a equal priority, passes on bus B. At the same time, the countdown counter for priority level I (CD_I) is decremented for each empty QA slot passing on bus A, incremented each time a REQ bit, with a higher priority, is read on bus B, and incremented each time a SELF REQ bit, with an higher priority, is generated within the node. In countdown state, the segment can be transmitted when the CD_I is equal to zero and there is an empty QA slot passing on bus A.

3.5 Performance of the DQDB Access Protocols

This section first reviews performance studies previously done on the DQDB access control methods. Most of these studies mainly discuss the performance of DQDB for transferring asynchronous segments over the QA slots. Then, a simulation model of the distributed queue defined by the IEEE 802.6 standard is presented. This model will be used in the following chapters presenting simulation studies of multimedia communications systems supported by DQDB.

3.5.1 Circuit Switching versus Packet Switching Capacities

A point of interest has been to evaluate the effects of isochronous circuit allocation on packet switching capacity [ZUK87,ZUK88]. The DQDB bandwidth is first allocated to meet the demand of the isochronous services. Capacity that is not allocated to isochronous traffic can be shared by the packet switching traffic. NA slots are defined to contain isochronous channels. However, because of the slotted structure imposed by DQDB, there can be waste of bandwidth when the isochronous octets in NA slots are not completely used. For instance, it can happen that even if only one octet, out of the L available in a NA slot, is allocated to circuit switching, an entire new slot must be designated as isochronous. Thus, this new slot becomes unavailable for packet switching even if only one octet is used for an isochronous communication. This waste, measured in octets, must then be considered in the performance evaluation of DQDB. Another source of waste is the occurrence of "holes" in the NA slots. Indeed, since each



AQ: ACCESS QUEUE
AQ_I: AQ AT PRIORITY I
CD_I: COUNTDOWN COUNTER AT PRIORITY I
IN: INPUT FROM OTHER AQ SITUATED IN THE SAME NODE
IN_B: INPUT FROM BUS B
REQ I: NODE WILL SET TO NEXT FREE REQ BIT ON BUS B
RQ_I: REQUEST COUNTER VALUE AT PRIORITY I
SELF REQ I: USED BY AQ_I FOR INFORMING OTHER AQ WITHIN THE SAME NODE ABOUT SEGMENT QUEUED

Figure 3.10: Distributed queue state machine for priority level I access to bus A.

connection always uses the same channel(s) during a session (no reassignment of channels following a termination of a circuit usage), empty octets occur in the NA slots.

As technique to allocate the isochronous octets, the first-fit circuit allocation policy has been proposed [ZUK87]. Under this policy, for each incoming circuit request, the octet with the smallest sequence number among all the empty octets is assigned. Zukerman [ZUK87] ha developed an analytical model showing the effect of circuit allocation on packet capacity under the first-fit circuit allocation.

A blocking probability scheme has been also proposed [ZUK88] for reducing the generation of a new NA slot upon receiving a circuit request. In fact, the idea of the scheme is that a circuit request may be blocked, according to a given probability, with the hope that when the next request (or a repeated attempt of the same request) arrives, there will be some unused octets within the first i slots, and accepting the next request will not increase the number of NA slots from i to $i+1$. The higher is i, the higher is the probability that the next request will not be blocked. Consequently, the blocking probability can be dynamically adjusted according to the number of NA slots defined per period of 125 μs. This, it has been showed [ZUK88] that when the number of isochronous channels in one NA slot is large compared to the number of NA slots defined in one implicit frame of 125 μs, the proposed blocking probability policy, even by assuming a low probability of request rejection (less than 1%), allows to improve the packet switching capacity (more than 5%).

3.5.2 Performance of the QA slot Access Control Method

Another point of study has been to evaluate the sharing of the asynchronous bandwidth between the DQDB nodes. In fact, the distributed queue defined in DQDB has been approximated by the M/D/1 queueing system [KLEIN] adding extra delay resulting from the slotted structure imposed by DQDB [NEW86,NEW88B]. The average access delay, W, of the approximate queueing system has been defined as:

$$W = \frac{\bar{x}\rho}{2(1-\rho)} + D_{slot} \tag{3.1}$$

where:

ρ = network utilization

$\bar{x} = \frac{125\ \mu s}{number\ of\ QA\ slots}$ = service time of the distributed queue

D_{slot} = delay caused by the slotted structure imposed by DQDB

However this approximation neglects the effect of the propagation delay along the bus of the REQ bit used to put a station in the distributed queue. Indeed, such delay causes the REQ bits to inform less rapidly upstream stations about the fact that some downstream stations have segments queued to send on the other bus. Therefore an upstream station may send a segment before other stations already having queued segments before it. Simulation studies [CONTI,DAVIDS,NEW88A] have shown that there is effectively unfairness between the stations along the buses, and the unfairness is amplified with the increase of the network load (some results are presented in the next section). In fact, under higher load conditions, the mean access delay of the stations adjacent to the heads of bus is quite low compared to the one of the stations situated on the middle of the buses. On the other hand, the presence of an important number of NA slots compared to the number of QA slots improves fairness, since NA slots also carry REQ bits which can be used to queue stations [DAVIDS].

3.5.3 A Simulation Model of DQDB

In this section, a model for simulating the DQDB "distributed queue", controlling the QA slot access, is presented. This model will be used in the next chapters for evaluating the performance of multimedia communications systems supported by DQDB. Figure 3.11 shows the model that is evaluated by using the QNAP2 package [QNAP2] which is briefly described

in appendix B. The simulation program of the DQDB "distributed queue" is presented in appendix D.1.

In the QNAP model, QA and NA slots are represented by customers. In the case of the NA slot, the octets contained in the segment payload are not really represented since only the access to the QA slots is evaluated. Customers are first created at each head of bus, then passed successively to each node, and finally destroyed at the end of the bus. Each head of bus creates N customers (slots) per cycle of 125 μs. Each customer is associated with three variables symbolizing the BUSY, REQ and TYPE bits. The BUSY and REQ bits are examined by the MAC of each node which adjusts its counters in order to keep track of the state of the network at any time [IEEE88B]. The TYPE bit is used to distinguish QA and NA slots. Finally, since a DQDB MAN network which can operate over a large area is assumed, the propagation delay for transferring slots between each node shall be represented.

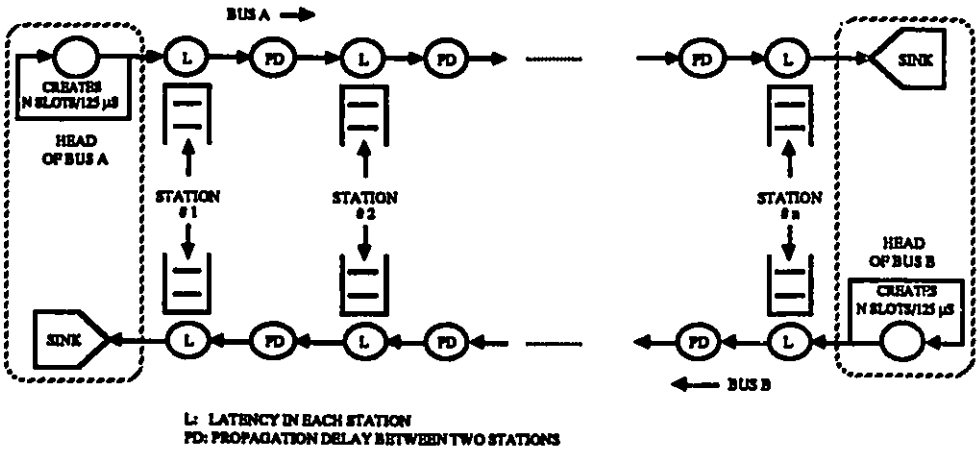


Figure 3.11: Simulation model of the "DQDB" distributed queue.

Assumptions

The main assumptions used in the simulation model of DQDB are now presented. The number of generated slots per period of 125 μs is fixed to 5: 1 NA slot and 4 QA slots. The NA slot

is assumed to support isochronous communications while the QA slots are shared by the stations generating asynchronous segments. Each station generates the same load (number of asynchronous segments per unit of time), and each generated segment is sent to one of the other stations with the same probability (the traffic generated by each station on one bus compared to the one on the other bus is proportional to the number of stations which can be reached through each bus). The stations are assumed to be uniformly distributed along the buses and their number is fixed to 25. The station latency is assumed to be one octet delay. On the other hand, three bus lengths are compared: 1, 10 and 150 km. The propagation delay on each bus is assumed to be 5.085 μs/km since the use of optical fibers as medium is assumed.

Analysis and Results

Eq. (3.2) represents the analytical approximation of the mean delay for accessing a QA slot. The service time of the distributed queue is given by Eq. (3.3).

$$W = \frac{\bar{x}\rho}{2(1-\rho)} + \frac{q}{N}\frac{T}{2} + \frac{n}{N}\frac{1}{n}\sum_{i=1}^{n}(2i+1)\frac{T}{2} \tag{3.2}$$

$$\bar{x} = \frac{125\ \mu s}{q} = \textit{ service time of the distributed queue} \tag{3.3}$$

where:

q = number of QA slots per period of 125 μs

n = number of NA slots per period of 125 μs

$N = q + n$

T = slot transmission time

For instance, if there are 3 QA slots available per period of 125 μs, then there will be an average of one QA slot available at each 41.67 μs. In Eq. (3.2), the waiting delay caused by the use of a slotted structure and the presence of NA slots in DQDB is also defined. It is assumed that in each implicit frame of 125 μs, the NA slots first appear, then the QA slots follow. Two delays are defined: 1) the mean waiting time which results when the next slot passing on the bus will be QA and which is represented by the ratio $\frac{T}{2}$ (where T is the transmission time of one slot); 2) the mean waiting delay which results when the next slot(s) passing on the bus will be NA before to eventually see the next QA slot and which is represented by the term $\frac{1}{n}\sum_{i=1}^{n}(2i+1)\frac{T}{2}$. These delays are shown in Fig. 3.12. Each delay is respectively multiplied by the percentage of QA and NA slots generated per period of 125 μs. Eq. (3.2) is obviously valid in situations where the bandwidth allocated for isochronous and non isochronous traffics do not vary during the time the system performance is evaluated.

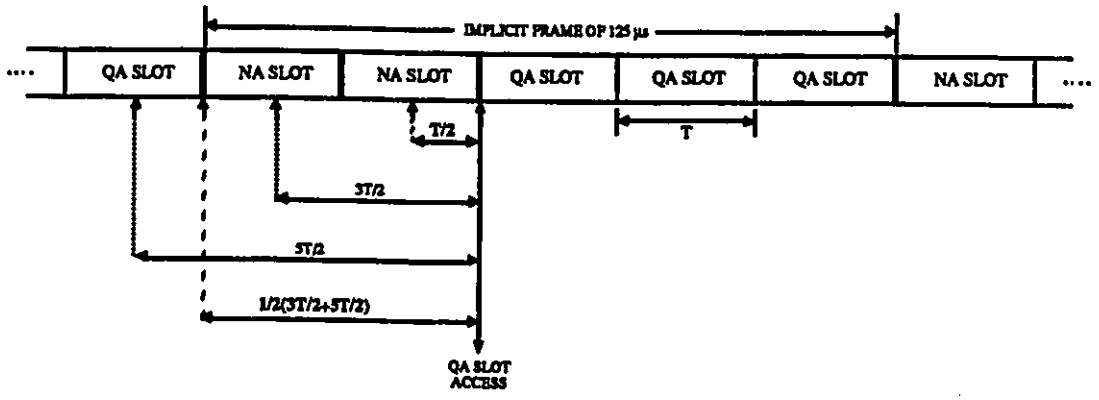


Figure 3.12: Mean waiting delay caused by the slotted structure and the presence of NA slots in DQDB.

Figure 3.13 shows the results obtained from simulation for a bus length of 150 km compared to the Eq. (3.2). The method considered for validating the results is the one described in appendix C. The results obtained from Eq. (3.2) are quite similar to the simulation results.

In Fig. 3.14, the mean access delay as a function of the position of the station along the 150

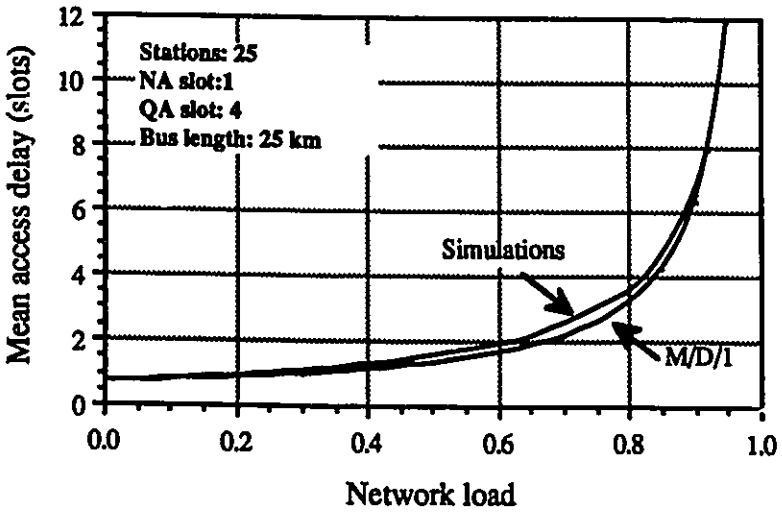


Figure 3.13: Mean access delay as a function of the network load.

km buses, under a network load of 80%, is shown. As it has been mentioned in section 3.5.2, the mean access delay of the stations adjacent to each head of bus is lower than the mean access delay of the other stations. A polynominal approximation (2nd degree) has been used to trace the curve in Fig. 3.14.

Figure 3.15 shows the effect of the network load on the unfairness between the stations. Under higher load conditions, the unfairness between the stations increases since downstream stations on the bus (those which are on the side opposite to the one where slots are generated) see more busy slots before being able to send a segment on the bus. Finally, Fig. 3.16 depicts the mean access delay for different bus lengths (1, 10 and 150 km). Clearly, this figure suggests that the performance of DQDB is not significantly affected by a variation of the bus length, even under heavy input traffic load.

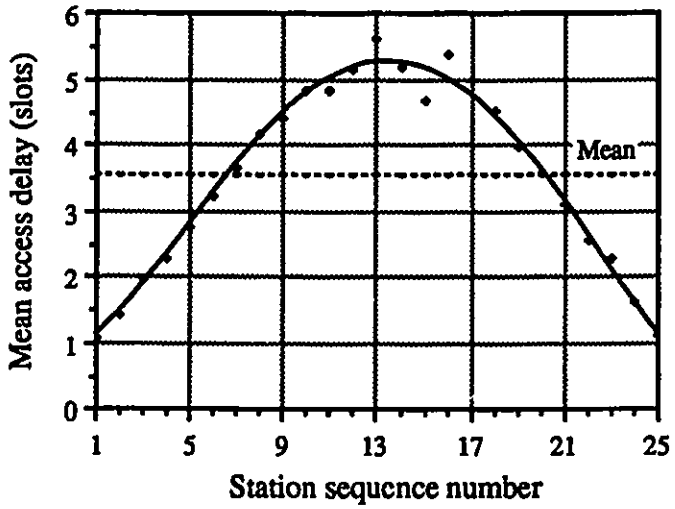


Figure 3.14: Mean access delay as a function of the positions of the stations on the buses (under a network load of 80%).

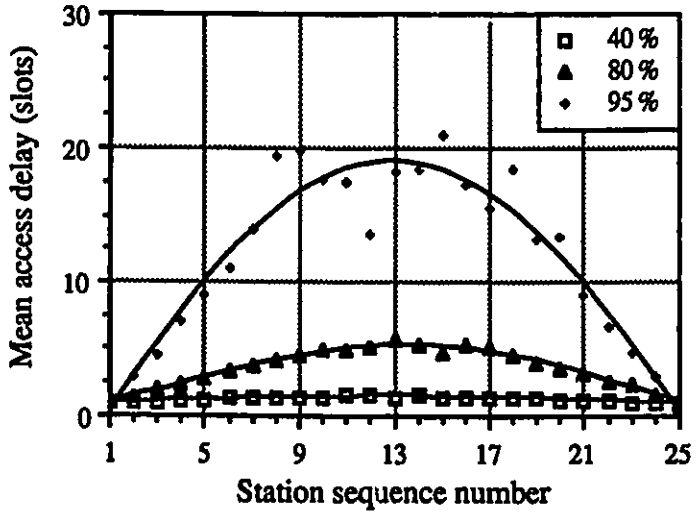


Figure 3.15: Mean access delay as a function of the positions of the stations on the buses (under different network loads).

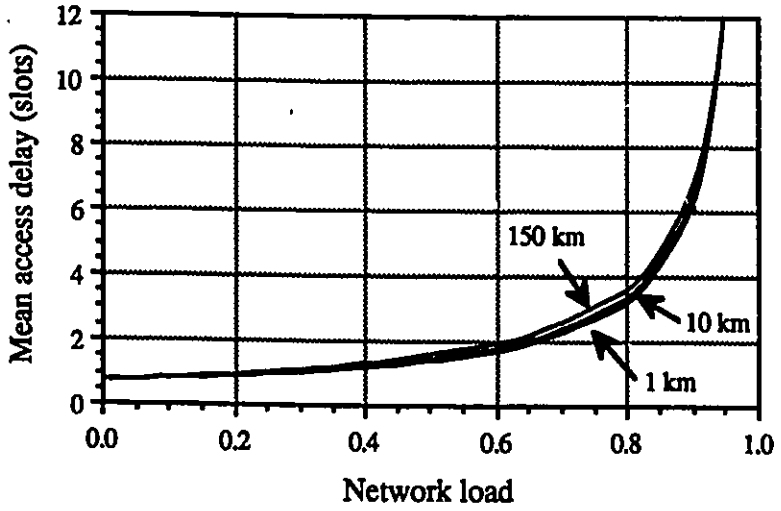


Figure 3.16: Mean access delay as a function of the network load (for different bus lengths).

3.6 Concluding Remarks

In chapter 2, it has been mentioned that there is another standard, called FDDI-II (Fiber Distributed Data Interface), which is intended to provide the same services as DQDB (integrated voice/data, high speed data transmission, interconnection of low-speed LANs). Hence, it may be interesting to compare them. For instance, some simulation studies [WAIN] have compared their performance. In particular, it has been shown that under configurations where a small area is covered by the network, better performance is obtained for FDDI-II (more bandwidth available), since in DQDB there is more bandwidth allocated to the segment overhead (7 bytes in each segment) as well as the slotted structure used. However, when the bus length of the network is considerably increased, the performance of FDDI-II decreases because of the high access delay even under lower load traffic. Indeed, in FDDI-II, each station must wait for the token before transmitting onto the medium. As the bus length of the network is increased, the waiting time for the token becomes significant. On the contrary, as it has been discussed in the previous section, the global mean access delay is not significantly affected by important

increases of the bus length in DQDB. Another advantage of DQDB on FDDI-II is also the possibility to use an open bus topology (Fig. 3.1), which, in some situations, can allow to reduce the length of the physical links used to transfer data.

It must be also pointed out the effects of using short slots in DQDB. Indeed, long MAC PDUs have to be fragmented into multiples of segments in order to be transferred on DQDB. This segmentation process, under a configuration where DQDB offers a large bandwidth (over 100 Mbps), can be the bottleneck of the system. In the next chapters, such behavior should be noticed when DQDB will be used as a high speed data network transferring radiological image files and used as a backbone network for interconnecting low-speed LANs.

Chapter 4

DQDB Supporting Radiological Image Browsing Applications

As mentioned in previous chapters, DQDB is mainly intended to provide high speed data transmission between nodes located within areas of up to 50 km in diameter. In this chapter, the use of DQDB is then evaluated within a communications system supporting radiological image transfers between stations situated in remote hospitals. In particular, the system performance for transferring images browsed by medical workstations logically linked to remote database servers is examined.

The first section of this chapter discusses applications involving image browsing. Then, in order to support such applications, a communications system is proposed in the second section. Finally, the last section presents a study of the proposed system performance. The emphasis is put on the delay to transfer images within different system configurations.

4.1 Image Browsing

As mentioned in the introduction of this thesis, studies are under way to implement systems facilitating the storage and retrieval of radiological images in hospitals. For instance, radiological images can be taken, digitized and stored in a database server. Then radiologists and physicians, using a medical workstation connected through a network to the database

server, can view X-ray images. Recently, browsing of images has been explored to improve the efficiency of radiologists and physicians. By browsing of digitized images, this means that a user can take a quick look through a set of images, with the possibility of stopping for a longer period of time on images which capture the attention. An example of browsing activities in medical environments is the case of a radiologist searching through the database for images with specified characteristics. Some simulation work has already been done on systems involving browsing of radiological images [MAYD].

Systems supporting image browsing have to deal with important transfers of data in short periods of time. A network such as DQDB is then a potential candidate for transferring the images between remote stations. Furthermore, since high resolution is required for representing radiological images, compression techniques may be used for reducing the image transmission delay. Hence, in this chapter, the manipulation of original and compressed images is also compared. In particular, such as a compression technique, the use of the reduced-difference pyramid data structure [WANG] is considered. In addition to providing image compression, the use of such structure allows performing progressive image transmission which can be useful within browsing activities. For instance, the user can browse a determined series of images with reduced-resolution, since a quick look is performed for each image. Moreover, the user can take more time on an image whose resolution can be progressively increased up to the full resolution defining the original image.

4.2 Image Browsing System

The system is composed of interconnected database servers and medical workstations using DQDB for transferring information between them (see Fig. 4.1). For the performance study, several established logical links, each of them corresponding to an image browsing session between a database server and a medical workstation, are assumed. This means that database servers have to segment each requested image into DM PDUs (derived MAC PDUs) in order to transfer the image through DQDB. By sending control information, the workstation requests

the database server to send the next image, or more information about the currently observed image if progressive image transmission is used. The other workstations, which do not have an established logical link with any database server, are not considered in this study, since only the delay for transferring each image during a browsing session is under interest (which also assumes that these other workstations do not generate significant traffic). Real-time voice communications are also assumed to be supported by DQDB through NA slots. In the next subsections, the data structures used to store and transfer radiological images in and from the database server are presented, and two image browsing activities which can be supported by the system are defined.

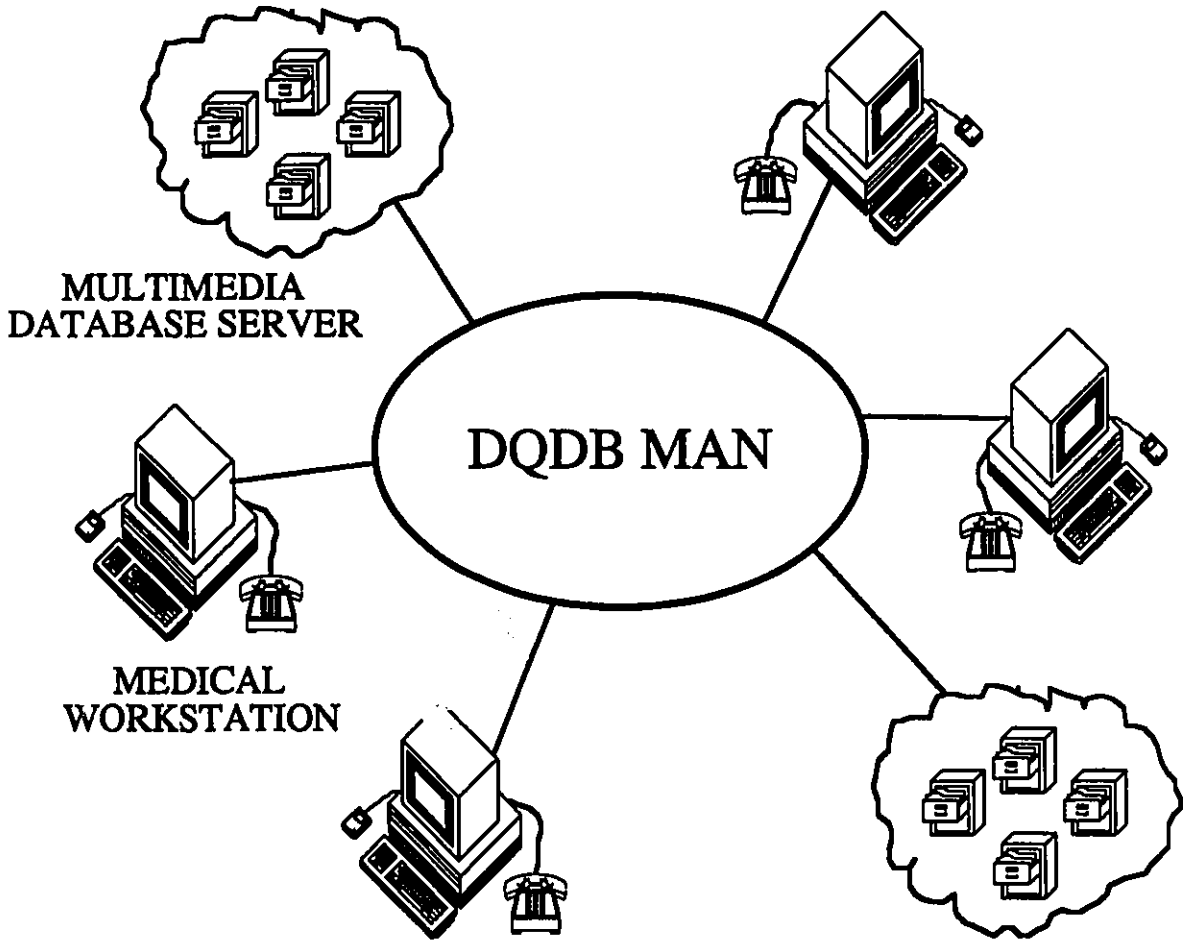


Figure 4.1: DQDB MAN supporting image transfers and voice communications

4.2.1 Full Resolution Images and Pyramid Data Structures

For storage and transmission purposes, two types of images are compared in the present case: a) original raster scanned and b) compressed images. In the former case, full resolution images without any compression are used. The images are stored and transferred as scanned. However, since high resolution images are required to allow accurate doctor's diagnosis (1000 x 1200 pixels x 8 bits for 14" by 17" film [GEOR]), a large amount of memory is needed and long delays to segment and transfer the image result. To overcome these problems, compression techniques can be considered [JAIN].

One solution may be the use of the compression technique called the reduced-difference pyramid [WANG]. The structure defined by this technique is based on the mean pyramid data structure [TANI] which is formed by successively averaging over 2 x 2 nodes (while pixels are the elements representing the original image, and the nodes represent the compressed image elements). In the pyramid structure, the image is represented by a hierarchy of levels where immediate levels correspond to reduced-resolution approximations (see Fig. 4.2). The top level is the average of the image while the bottom is the original (full resolution) image. To perform progressive image transmission, the pyramid is transmitted starting from the top level. Within browsing activities, this type of transmission allows the user to glance at a series of reduced-resolution images and to request for further data, if he desire to increase the image resolution.

However, the mean pyramid needs more data than the original image, since the number of nodes required is almost 1/3 more than the number of pixels of the original image. Indeed, each reduced-resolution generated from a higher resolution of n by n nodes needs to be represented by $\frac{n}{2}$ by $\frac{n}{2}$ nodes. This means that for an original image of k by k pixels, $\sum_{i=0}^{l} 4^i$ nodes will be required to store the image according to the mean pyramid data structure. The parameter l represents the number of levels needed in the mean pyramid data structure representing the image and comes from the relation $k = 2^l$. The term 4^i represents the number of nodes required to represent the i^{th} level of the pyramid. Thus, for a large l, the mean pyramid will

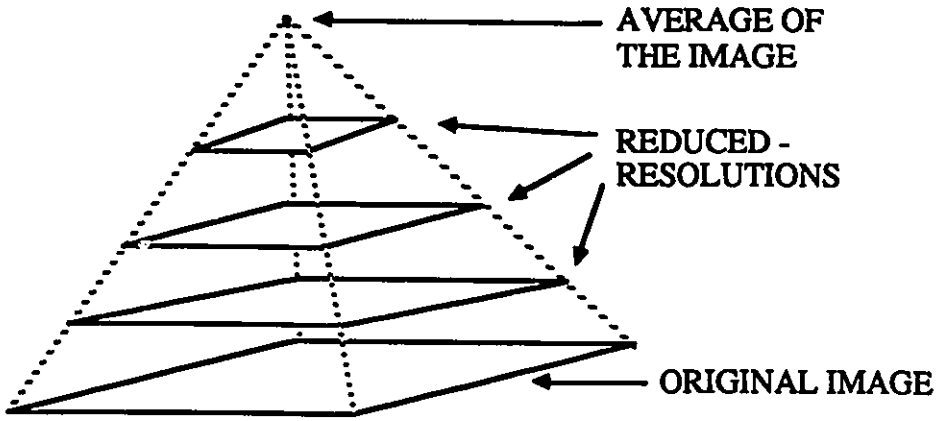


Figure 4.2: Pyramid data structure.

approximately need $\frac{4}{3}k^2$ nodes to be stored, which is 1/3 more than as required to store the original image. This is represented in Eq. (4.1).

$$\sum_{i=0}^{k} 4^i = k^2 \sum_{i=0}^{k} \frac{1}{4^i} = k^2(1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \cdots) = k^2(\frac{1}{1 - \frac{1}{4}}) = \frac{4}{3}k^2 \qquad (4.1)$$

The reduced-difference pyramid data structure has then been proposed [WANG] in order to reduce the number of nodes representing the mean pyramid. This structure is formed by calculating the difference between the 2 x 2 nodes at the same level in the mean pyramid and retaining only three out of four differences (see Fig. 4.3). Hence, this structure offers several advantages compared to the mean pyramid. The number of nodes at each level in the reduced-resolution pyramid is reduced by 1/4. Therefore, the total number of nodes in the pyramid is the same as the number of pixels in the original image. Moreover, by calculating the differences between neighboring nodes at the same level, the mean number of bits for coding each node is considerably reduced compared to the number of bits associated with each pixel in the original image.

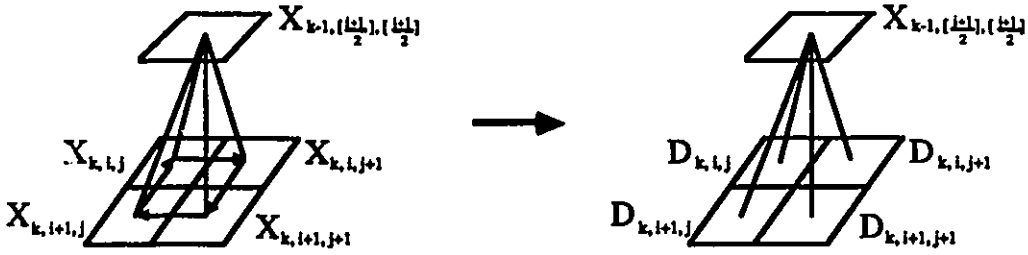


Figure 4.3: Reduced-difference pyramid data structure formation.

4.2.2 Image Browsing Activities

Two types of browsing activities are defined. The first one involves successive transmissions of radiological images having the same size, with a random time between each image sending. This time comprises the time necessary to display the image at the workstation, and for the user to read the image and ask for the next image. The second type of browsing uses the concept of progressive image transmission mentioned in the previous sections. For every image, a reduced-resolution version is first transmitted and then, upon user request, the database server may send more data for increasing the resolution of the image. Here, there must be a particular communication protocol between the database server and the workstation at the application layer. For instance, a timer may be started at the database server after the transmission of an image has been completed. During the timeout period, the workstation user may then issue a request to the database server to send the next image. If the database server does not receive any request before the timer expires, it will send data improving the resolution of the current image read by the workstation user (see Fig. 4.4). This procedure is repeated each time a reduced resolution image is sent. Therefore, the resolution of the image is improved as long as the user does not send any request. One or several levels in the pyramid data structure can then be sent before the last level detailing the full resolution image.

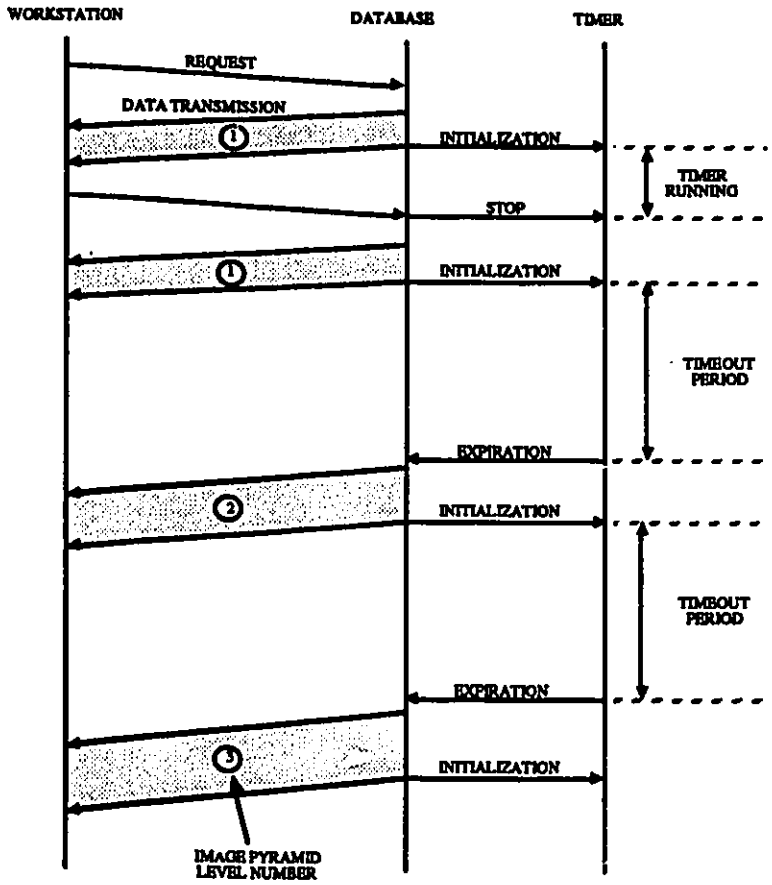


Figure 4.4: State diagram of the protocol between the database server and the workstation in the case of browsing including progressive image transmission.

4.3 Performance Study

In this section, the simulation model of the communications system defined in the previous section is first presented. Then, the assumptions underlying the system configuration are defined, and the results obtained through simulation are presented and analyzed.

4.3.1 Simulation Model

The simulation model of the proposed system is mainly based on the DQDB model presented in section 3.5.3. However, each node generating asynchronous segments is replaced by a queue simulating the services of each database server. The simulation program of the complete system is presented in appendix D.2. The workstations are not directly represented in the simulator. Indeed, the behavior of the workstation is involved in the model representing the database server with which the workstation is doing a browsing session. Indeed, the algorithm simulating the services of the database server non only involves the transmission of image segments but also the the time during which the workstation user is assumed to wait for an image, to read it, and to optionally inform the database server to send the next image.

The first type of browsing presented in section 4.2.2 involves successive transmissions of radiological images having the same size, with some delay between each image transfer. In this case, the database server first sends all the image. Then, having completed the image transmission, the database server waits for an exponentially distributed period of time before transmitting the next image. Only full resolution images are assumed to be sent in this first type of browsing.

Figure 4.5 presents the flowchart of the algorithm simulating the operation of the database server for the case of browsing including progressive image transmission (see also Fig. 4.4). At the beginning, the database server sends the first reduced-resolution version of the image. After having sent all the version segments, the database server waits during a period of time exponentially distributed. During this time, the workstation user is assumed to receive and

read the reduced-resolution version of the image, and to choose between receiving a higher resolution of the image or requesting for the next image. In QNAP2, a bimodal function can be used to represent two alternatives which can be done with unequal probability for the event of each of them. Thus, after this waiting period, the database server performs one of the two previously mentioned events.

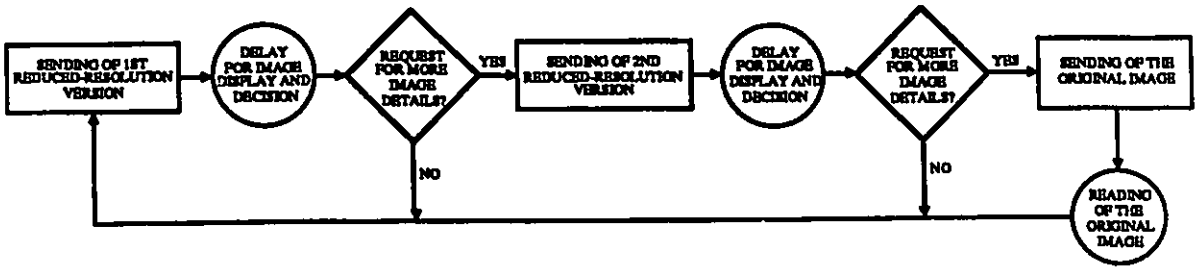


Figure 4.5: Algorithm of the protocol between the database server and the workstation in the case of browsing including progressive image transmission.

4.3.2 Assumptions

The values of the different parameters of the model have been chosen by considering the recommendations stated in the proposed IEEE 802.6 standard [IEEE88B], and the values reported within performance studies [KOSIT,VAL89A] and tests done in laboratories [GEOR] involving manipulation of radiological image files. The DQDB bit rate is fixed by the number of slots in each frame of 125 μs. Each slot insertion brings an additional bit rate of 4.416 Mbps. In the present study, various number of QA slots to transfer radiological images are considered, and one NA slot per frame is assumed to support real-time voice communications between different users. The other assumptions concerning DQDB and made in section 3.5.3 are used. Furthermore, since DQDB is assumed to interconnect several hospitals (at the same time, several database servers and a large number of medical workstations), we assume a DQDB looped bus topology whose circumference length is important (25 km). Along the buses, database servers are considered to be equally distributed and their number is fixed to 8. Finally, as mentioned in the previous section, the simulation model only represents the

workstations which have a logical link established with a database server for performing image browsing.

In the database server, images are stored on a Mercury 8512-84 SCSI-II 8 inch Winchester disk drive, using a SCSI interface, which allows retrieving images of 1000 x 1200 pixels x 8 bits (for 14" x 17" film) from the disk in under 1 second [GEOR]. Furthermore, from tests in the laboratory, it has been measured that the image segmentation into packets of 9 kbytes can be done in approximately 1.5 seconds. In fact, an image of 1000 x 1200 pixels x 8 bits, stored in the form it was originally scanned, thereby comprising 1.144 Mbytes, must be divided into DQDB MAC PDU whose maximum length can be 9 Kbytes (minus 32 bytes of overhead) [ANSI85A,IEEE88B]. For instance, the original image, comprising 1.144 Mbytes, must be divided into 131 DQDB MAC PDUs (130 PDUs using the maximum length). Once this is done, each MAC PDU must be segmented into DM PDUs (derived MAC PDUs) to be then inserted into the QA slots. Hence, for the image previously defined, the MAC PDUs must be divided into a total number of 19469 DM PDUs. Images of 2000 x 2400 pixels are also used.

Furthermore, when the reduced-difference pyramid data structure is applied on radiological images having more than 1 Mbytes, 4 bits per node are assumed which has the effect of reducing by 2 the total image manipulation size. This approximation has been deduced from simulation results involving different types of images [WANG]. However, in order to easily implement the pyramid data structure, images of 1024 x 1280 and 2048 x 2560 pixels are assumed when image browsing including the use of the progressive image transmission is performed.

Finally, the delay comprising the time necessary for the reception and the display of the image at the workstation, as well as the time necessary for the workstation user to inform the database server to send the next image, is assumed to be exponentially distributed with a mean of 2 seconds. In the case of browsing including progressive image transmission, the same thinking time, between each reduced-resolution image, is assumed.

4.3.3 Analysis and Results

The results for the system described in the previous sections are now presented. Two scenarios are considered: 1) image browsing, in which fixed size images are sequentially sent; 2) image browsing with the use of progressive image transmission (in which reduced-resolution images are browsed with the possibility, for the workstation user, to increase the resolution of an image that captures his attention). For each scenario, 8 preestablished logical links are considered, where each of them corresponds to a browsing session between a database server and a workstation. Moreover, several DQDB asynchronous bandwidths as well as full resolution images of 1000 x 1200 pixels by 8 bits and 2000 x 2400 pixels x 8 bits (for the new generations of high resolution monitors) are compared.

Figure 4.6 shows the results of the first scenario (the method used for validating the results is described in appendix C). It can be observed that a small number of QA slots can limit the sending of image segments and cause significant image display delay. This delay can be reduced by increasing the DQDB asynchronous bandwidth (i.e. the number of QA slots per frame period of 125 μs). However, by increasing gradually the DQDB asynchronous bandwidth, the segmentation process in the database servers eventually become the bottleneck of the system. Thus the minimum possible delay is 1.50 seconds for 1000 x 1200 pixels images (19469 DQDB segments) and 6.0 seconds for 2000 x 2400 images (82345 DQDB segments). From the point of view of the user performing image browsing, it is important to determine what is the minimum acceptable delay in order to avoid wasting of time by waiting for images. If 5 seconds is considered as the maximum acceptable waiting delay, the present system will require the use of a minimum of 2 QA slots per frame (DQDB asynchronous bandwidth of 8.83 Mbps) for browsing 1000 x 1200 pixels images. However, for 2000 x 2400 pixels images, such delay requirement cannot be reached. Therefore, it may be inappropriate to browse images having such resolution.

Figures 4.7 and 4.8 correspond to browsing activities including progressive image transmission where images compressed according to the reduced-pyramid data structure are used.

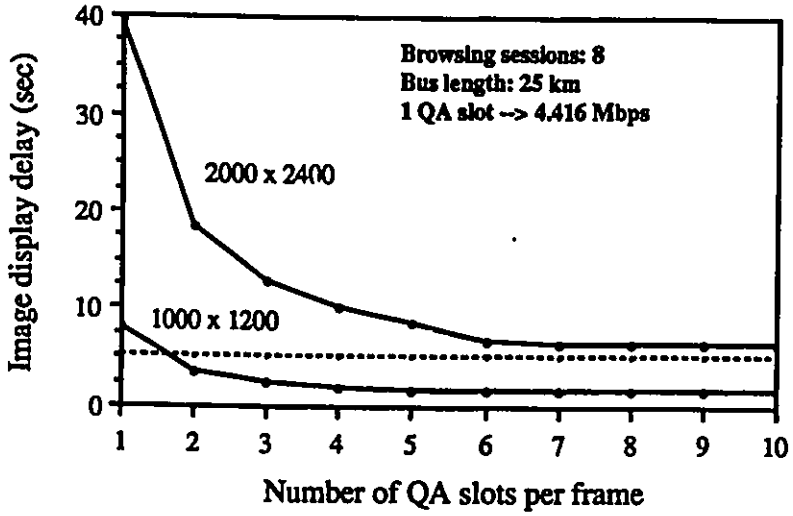


Figure 4.6: Mean image display delay for full resolution images of 1000 x 1200 pixels x 8 bits and images of 2000 x 2400 pixels x 8 bits.

The curves show the mean image display delay for each pyramid level compared to the number of QA slots per frame. Results for originals image of 1024 x 1280 pixels x 8 bits are shown on the former figure. In this case, reduced-resolution images of 256 x 320 and 512 x 640 pixels are considered (256 x 320 as a minimum resolution to represent radiological images). In fact, workstation users, currently performing a browsing session, initially browse images of 256 x 320. If a particular image captures their attention, the users decide to get an increase of the image resolution. In the present study, it is assumed that the users decide, with a probability of 20%, to bring up to 512 x 640 the resolution of the current displayed image. Then, having read the improved image, the users decide, according to a probability of 80%, to receive the data representing the full resolution image of 1024 x 1280 pixels (see Fig. 4.5). Results show that the mean image display delay, for each pyramid level, is quite low compared to the first scenario, since less segments are needed to represent each level.

Figure 4.8 depicts the results for the same scenario as the one of Fig. 4.7, but the use of images of 2048 x 2560 pixels x 8 bits is now considered. Furthermore, reduced-resolution images of 512 x 640 and 1024 x 1280 are used to perform the progressive image transmission. Simulation results show that 2 QA slots per frame are sufficient for getting mean image display

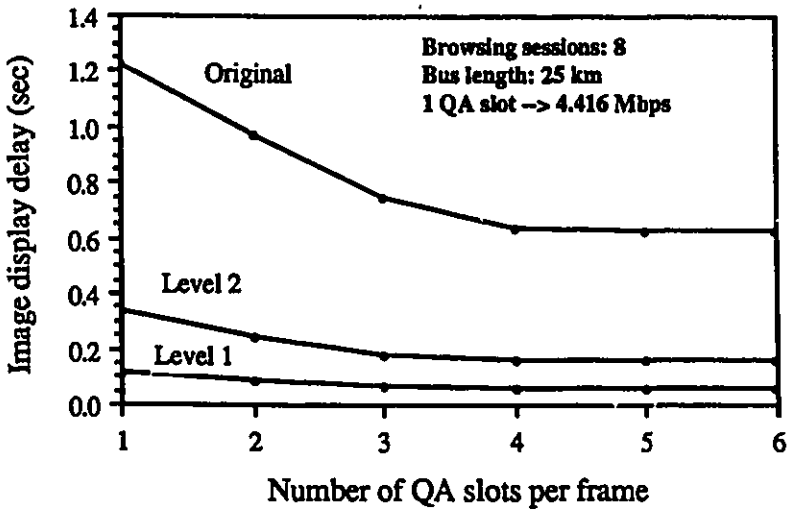


Figure 4.7: Mean image display delay for transferring the three higher levels in the reduced-pyramid data structure representing images of 1024 x 1280 pixels x 8 bits.

delay under 5 seconds. With 4 QA slots, the mean image display delay of the full resolution image is already under 3 seconds. Thus the use of the reduced-difference pyramid allows getting reasonable mean image display delay for the browsing of reduced-resolution images whose resolution may be eventually increased up to 2048 x 2560 pixels.

It must be pointed out that if the number of active browsing sessions is increased, more QA slots will have to be defined per frame period of 125 μs in order to obtain similar delays.

4.4 Concluding Remarks

This chapter has presented an evaluation of the performance of image browsing activities within a medical environment. The proposed IEEE 802.6 DQDB MAN standard has been chosen as communications support for these applications. The use of images of 1000 x 1200 and 2000 x 2400 pixels (8 bits/pixel) has been compared. From simulation results, it has been shown that workstation users can browse adequately full resolution images of 1000 x 1200 pixels but not 2000 x 2400, since the segmentation time of the second type of images is quite important (6.0 sec). The use of a compression technique called the reduced-difference

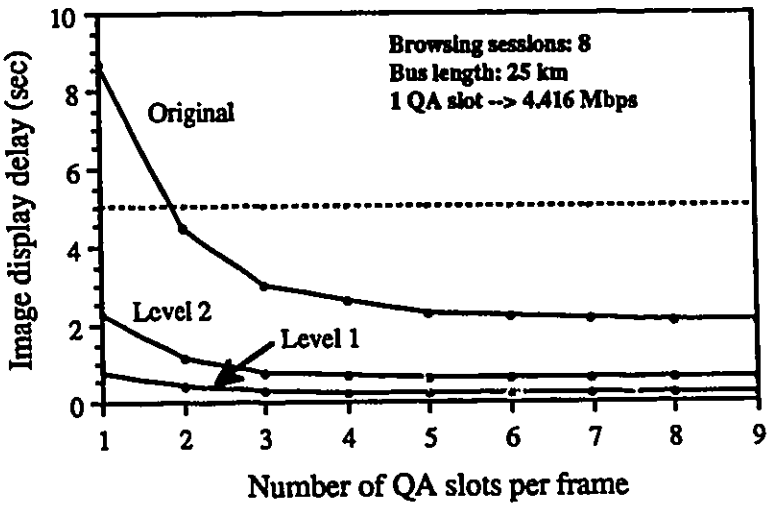


Figure 4.8: Mean image display delay for transferring the three higher levels in the reduced-pyramid data structure representing images of 2048 x 2560 pixels x 8 bits.

pyramid has been also examined. In addition to reducing the amount of data used to represent images, this technique allows performing progressive image transmission. Such transmission permits workstation users to browse over a set of reduced-resolution images with the option of halting on images and progressively increase their resolution. Under such scenario, it has been determined that the minimum image display delay when there is a sufficient number of QA slots per frame period of 125 μs.

Thus the use of DQDB is recommended for performing image browsing, since this network can support the required bit rates. However, future work and research must be done in order to reduce the time delay caused by the reading and segmentation of images in the database servers (in particular for images of 2000 x 2400 pixels).

Chapter 5

DQDB as a Backbone Network for Interconnecting LANs

This chapter presents a performance study of DQDB used as a backbone network in packet switched mode to interconnect token rings operating as specified in the ANSI/IEEE 802.5 standard. Interconnection between DQDB and token rings is provided by gateways which offer various protocol services to ensure proper network interworking. A network architecture, in which the IEEE 802.2 LLC type protocol is employed as end-to-end protocol between stations on the token rings, is also considered. Several LLC window sizes, different priority schemes, and various traffic intensities and bit rates in token rings are considered. The system performance is examined for two different applications: data packet transfer and browsing of radiological images. The emphasis is put on the system efficiency for transferring I-frames between token rings through DQDB. Results show that the proposed system can support various activities. The allocation of higher access priority to the gateways in the token rings is also recommended.

5.1 Context

As introduced in this thesis, DQDB can be used as a backbone MAN for interconnecting low-speed LANs. Low-speed LANs, such as those presented in section 2.2, are usually running between 1 and 20 Mbps and supporting local traffic within a campus area. For the study

presented in this chapter, the token rings defined in the ANSI/IEEE 802.5 standard [ANSI85D] have been chosen. The main reason of this choice is the possibility to use different priority access modes in token rings.

Interconnection between DQDB and token rings is provided by gateways which must perform segmentation and reassembly, as the interconnected networks do not use the same type of packets (different overhead and maximum packet sizes [SHOCH]). In case of congestion, the gateways, having limited buffer size, discard frames they cannot handle momentarily [BUX85,GER88]. To perform the recovery of lost frames, the IEEE 802.2 LLC type 2 protocol [ANSI85A], which has been reviewed in section 2.2.6, is considered as end-to-end protocol. This also allows reducing the packet processing time in the gateways.

Bux and Grillo have done a similar study by considering a system architecture comprising several token rings interconnected by a token ring backbone [BUX85]. They have shown that the performance of this interconnection architecture can be improved by using a dynamic flow control mechanism in the IEEE 802.2 LLC protocol and by assigning a higher priority to the bridges. They have also found that buffer space allocation techniques do not overcome the problem of congestion. In their study, bridges are used to transfer packets between rings without any protocol conversion. The present study differs from theirs in two points: a) the interconnection of two different networks is considered: slotted ring (DQDB) and token ring interconnected via gateways providing protocols services, such as packet segmentation and reassembly, to ensure proper network interworking; and, b) the system is evaluated by considering two applications: short message and image file transfers between stations, such as workstations and database servers, on any token rings.

5.2 The Interwork System

The present work considers the use of DQDB as a backbone network interconnecting two token rings comprising a given number of stations (Fig. 5.1). Gateways are used to interconnect the

token rings with DQDB. Communications between gateways, on DQDB, are assumed to be done over QA slots (packet switching). Logical links are assumed to be established between every pair of stations. The next few subsections describe one by one the architecture of the different elements of the interwork system supported by DQDB, namely, token rings, gateways and end stations on the token rings.

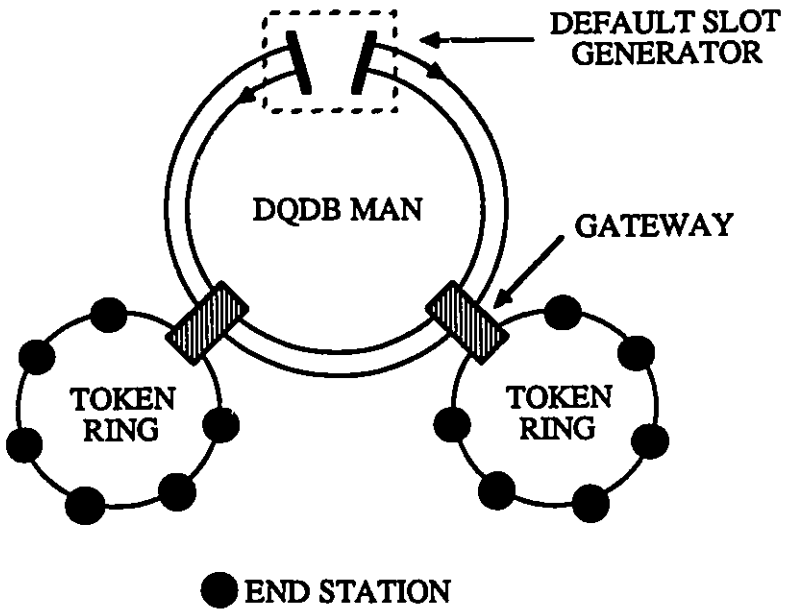


Figure 5.1: An interwork communications system.

5.2.1 Token Rings

This study involves the use of the token ring configuration standardized by the IEEE 802.5 committee [ANSI85D] which defines a priority mechanism with eight levels of priorities. Two modes of priorities are studied: a non priority mode in which all stations and gateways on the ring have the same ring-access priority and may send one frame per token arrival (non exhaustive service); a priority mode in which higher access priority is allocated to the gateways. In priority mode, the gateways, whenever they have a frame queued for transmission, will either seize the low priority token or make a reservation for high priority level. The second

event forces the currently transmitting station, after having sent its frame, to issue a priority token which will be then used by the gateway. After the gateway has completed its transmission, the low priority token is granted to the next station downstream from the one which last transmitted. In this mode, gateways are allowed to send all frames queued in their MAC server, then assuming a very long token-holding timeout. All the other stations continue to be limited in sending just one frame each time they capture the token (non exhaustive service). On the other hand, as the IEEE 802.5 committee is preparing the text of the standard for the 16 Mbps Token Ring [GLASS], this speed shall be also considered in the present study.

5.2.2 Gateways

To interconnect DQDB with token rings, gateways which comprise two independent processors have been defined (Fig. 5.2). One processor is used for converting token ring frames into DQDB MAC PDU and for segmenting the DQDB MAC PDU in multiple segment payloads or DM PDUs, while the other is used for performing the inverse operation (see Fig. 5.3). Having been formatted by the processors, each packet is then queued and processed by the MAC. Moreover, in order to reduce the processing delay in the gateways, it has been assumed that the gateways do not perform any error recovery action or flow control function. In order to perform these tasks, the LLC type 2 (connection-oriented) protocol [ANSI85A] is used as end-to-end protocol between end stations (see Fig. 5.4). Finally, equally divided buffer sizes are distributed between the two processes (segmentation and reassembly). Hence, packets are discarded when they do not find a sufficient number of free memory segments upon their arrival at a gateway [BUX85,GER88].

5.2.3 End Stations

Two types of stations are considered on the token rings: workstations and database servers. Figure 5.4 shows the protocol layering structure of both types of stations for the case where both of them are connected through the interwork system. At the layer 7, there is the message

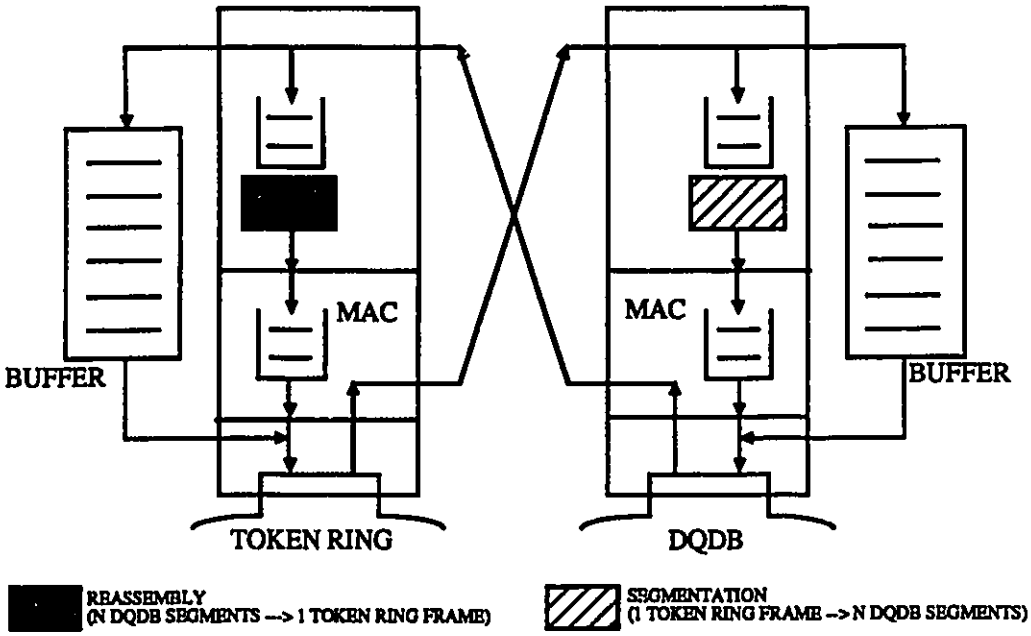


Figure 5.2: Model of the gateway interconnecting DQDB and token ring.

and file transfer applications. In the case of a file transfer application, the file is read from the database disk and eventually displayed at the workstation. The layer 6 (presentation) is used for any image compression and decompression techniques. Then, the lower layers are the same for all types of stations. They are processed in the network adapter which allows interconnecting the stations through the internetwork. The session and transport layers are not directly considered in this study. The network layer is not also represented, since its use can be quite simplified by using the LLC protocol as end-to-end protocol between end stations (similar to the concept of frame relaying [LAI]). At the LLC sublayer, the presence of the LLC type 2 protocol is assumed. In particular, the concepts of the I-frames for transferring information data, S-frames (supervisory frame) for indicating frame acknowledgement (RR-frame) or rejection (REJ-frame), and the P/F (poll final) bit for reestablishing data link connections have been used. The "Go back n" protocol is used to throttle the generation of packets by the stations in order to reduce congestion problems in the system. Finally, by using timers, the LLC protocol can reestablish a communication which has been interrupted

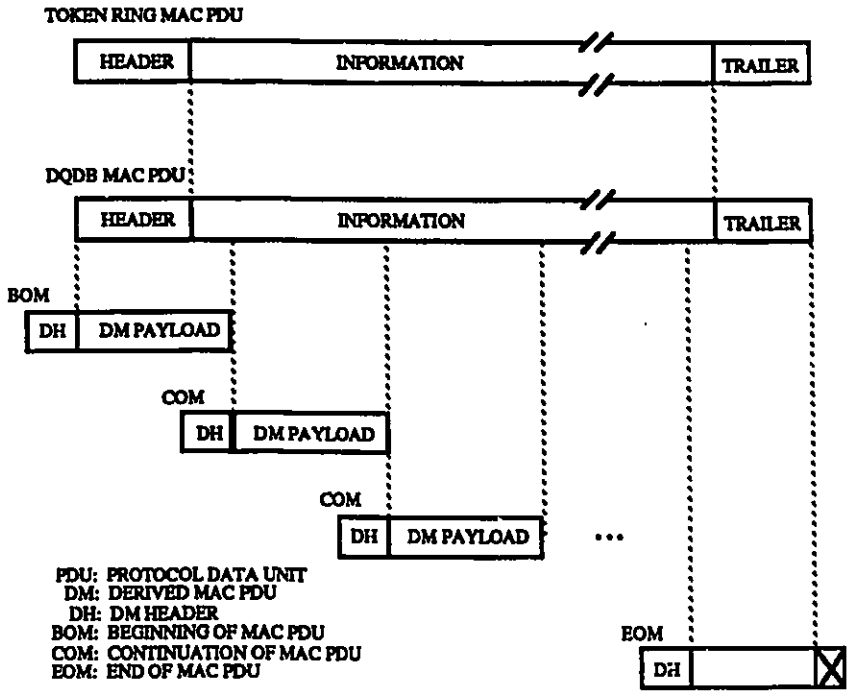


Figure 5.3: Conversion and segmentation of token ring frames in the gateway.

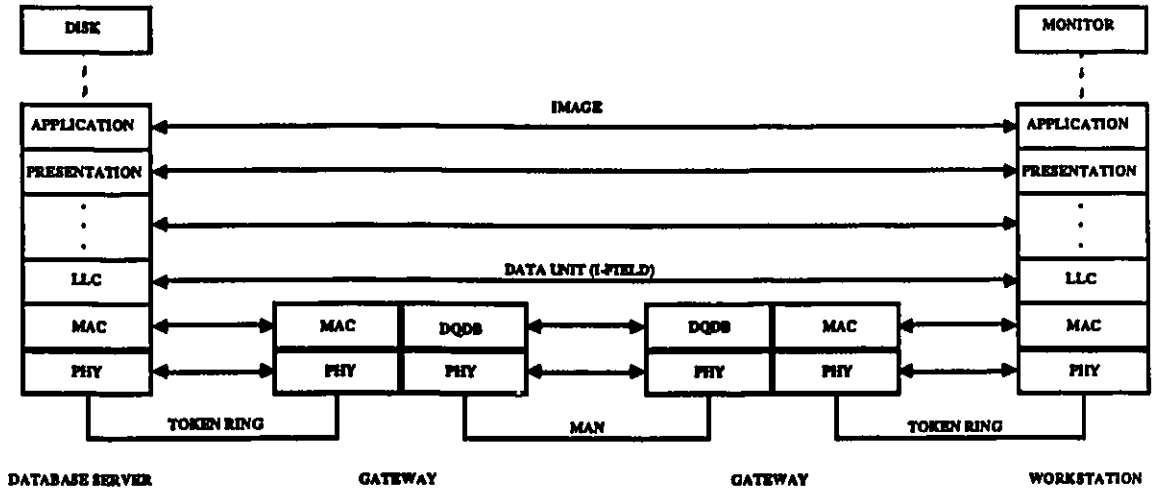


Figure 5.4: Protocol layers considered in the interconnection scenario.

following rejection of packets in the gateways. At the MAC and physical layers, there is the token ring protocol (and DQDB protocol in the gateways) as previously defined. However, operations at the physical layers are assumed to not have any consequence on the system performance (insignificant processing times).

5.3 Performance Study

This section presents an evaluation of the system performance by examining two particular applications. The first one is the transfer of data between homogeneous stations situated on different token rings. The second one is the browsing of radiological images by medical workstation users. In each application, the simulation model is first presented, then the underlying assumptions, and finally the results and their analysis.

5.3.1 Data Packet Transfer between Homogeneous Stations

In this first application, full-duplex communications, preestablished between pair of stations situated on different token rings (see Fig. 5.1), are assumed. Each station generates data packets in sequence.

Simulation Model

The QNAP program for simulating the whole interwork system is presented in appendix D.3.1. The part of the model simulating DQDB is the same as the one used in section 3.5.3 and 4.3.1. However, only two nodes appears on DQDB. These are the gateways interconnecting DQDB with the token rings. Figure 5.2 shows the simulation model of the gateway that must perform segmentation, reassembly and transmission of formatted packets to the next network. Both processors are simply represented by a queue, and by a number of memory partitions which are only released when affected packets are transmitted by the gateway MAC. Arrival segments at the gateway responsible for the reassembly must be counted for determining later if the

whole MAC PDU can be correctly reassembled when the last DM PDU (EOM segment) is received [IEEE88B].

Figure 5.5 shows the simulation model of the token ring operation. The exchange of the token within the ring is represented by a flag. Only one token may be circulating through each token ring network at any time, since the token ring has a short rotation time compared to the transmission time of the packets sent on it. A monitor is responsible for passing successively the token from one station to another one [BUX81]. In the case of the priority mode, the monitor, before passing the token to the next station, verifies if the gateway MAC has a packet queued for transmission. If so, the monitor passes it the token. After the gateway MAC has emptied its queue, the monitor returns the token to the next station downstream from the one which last transmitted before the gateway MAC.

Figure 5.6 shows the model of the typical end station used in the case of the first application. At the application layer the station generates fixed length data units at time intervals 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 the LLC or being queued to it. The value of 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 as I-frames. The threshold M is then used in a flow control technique called "backpressure" (Fig. 5.6).

Then after being queued at the high layer server, data unit packets are sent to the LLC where the packets are formatted as I-frames. However, the sending of data unit packets by the high layer is throttled down 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 are not yet acknowledged is kept in the buffer server.

Figure 5.6 clearly shows that the LLC server is a key part at the end station. It has to send and receive I- and S-frames (and if necessary to resend I-frames), to slide the window when it

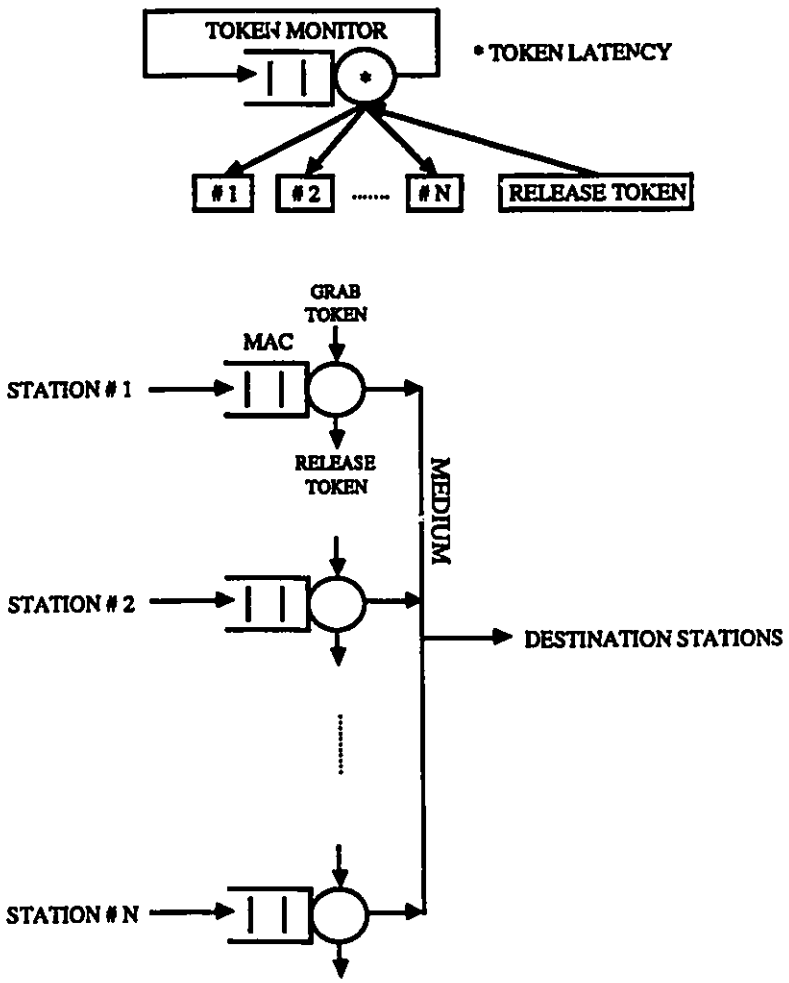


Figure 5.5: The model to simulate the token ring.

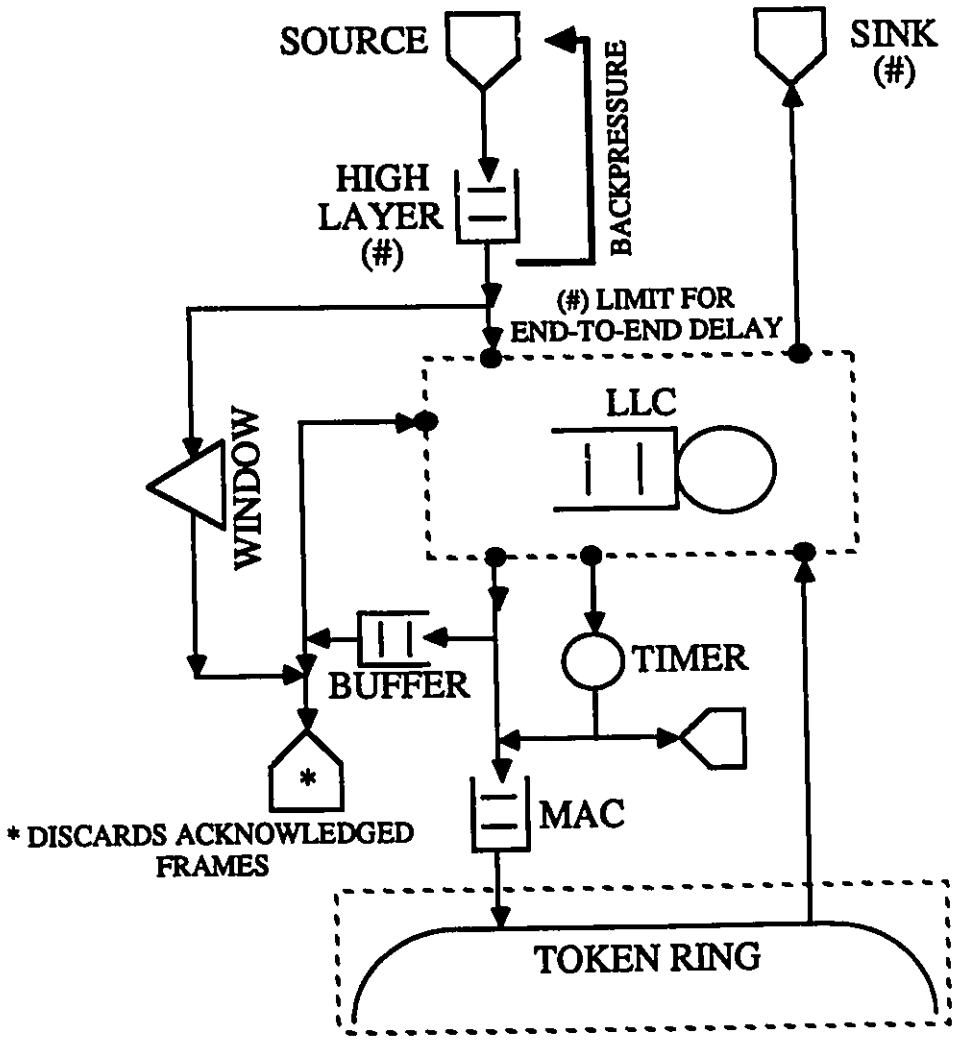


Figure 5.6: Model of end station.

receives acknowledgement for I-frames and to control the timer. Indeed, the LLC queue can reinitialize or interrupt the timer at anytime. If the timer expires before being affected by the LLC, a RR-frame to the MAC with the P-bit set to 1 will be sent to the remote LLC. Finally, at the bottom of the model presented in Fig. 5.6, there is the MAC server which queues I- and S-frame for transmitting them later on the token ring.

Assumptions

The values for the model parameters have been chosen by considering the recommendations stated in different standards [ANSI85A,ANSI85D,IEEE88B], the values reported within performance studies [KOSIT,VAL89B,VAL89C] and tests done in some laboratories [GEOR], and some assumptions made in section 4.3.2. The DQDB bit rate is assumed to be 45 Mbps for compatibility with DS3 circuits. This is translated to the equivalence of 10 slots per frame period of 125 μs. One of these slots is reserved in order to support real time voice communications between users. The remaining are available for transferring data packets between gateways connecting token rings and DQDB. A DQDB looped bus topology whose circumference length is 25 km is assumed. The attached nodes are supposed to be equally distributed within the network and their latency delay is equal to one octet delay. Furthermore, pairs of token rings running at 4 and 16 Mbps [GLASS] are examined. The delay for transferring the token between stations is assumed to be negligible.

At the end stations as well as at the gateways, the use of COMPAQ DESKPRO 386/20 computers which are actually employed in some laboratories [GEOR] are considered. The I-field and S-field frames used on the token rings are assumed to have respectively a fixed length of 1 kbytes (plus 25 bytes for overhead) and 25 bytes. The choice of 1 kbytes is mainly justified by the fact that the utilized COMPAQ is running on XENIX which allocates memory partitions by multiples of 1 kbytes. As stated before, DQDB uses QA slots having a length of 69 bytes: 7 for identification and control, and 62 for containing asynchronous segments furnished by the fragmentation of longer DQDB MAC PDU. Therefore, at the gateways, a

token ring I-frame is segmented into 17 QA slots after being converted to DQDB MAC PDU. The S-frame is transferred in just one QA slot. The processing time for converting the token ring frames into DQDB MAC PDU (as well as for the reverse process) is fixed at 150 μs, and at 31 μs to format each QA slot of 69 bytes. These times are provided 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 reassembly the whole MAC PDU, a time of 20 μs is assumed for decapsulating each derived MAC PDU. On the other hand, 2 sets of buffers having a size of 8 kbytes are comprised in each gateway. This value has been used in similar performance studies [BUX85]. One set is used by each direction of data flow in the gateway.

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 timeout value is 250 ms. The backpressure threshold value M is set to the window size plus 4. This gives a source node sufficient flexibility to prepare I-frames for later transmission also during times when the LLC window is closed.

Analysis and Results

In the gateways, the minimum time for processing token ring I-frames containing 1 kbytes of information has been determined to be 677 μs, which is more costly in time than the reassembly process, since the latter does not manipulate slot overhead. Thus, a maximum of 1477 packets of 1 kbytes per second can be processed. In the case of token rings running at 4 Mbps, one I-frame is transmitted in a minimum time of 2.09 ms, which means that a maximum of 477 packets can be transmitted per second (523 μs for token rings running at 16 Mbps). Table 5.1 summarizes these times.

Servers	*time/packet (ms)*	*packets/sec*
gateway (seg)	0.677	1477
token ring (4 Mbps)	2.094	477
token ring (16 Mbps)	0.523	1912
database	1.280	781.3

Table 5.1: Service times of servers which are susceptible to be system bottlenecks (the database service time is used in the case of the second application).

From these values it can be anticipated that 4 Mbps token rings will tend to saturate under heavy load. Congestion problems will particularly occur in the gateway MAC, since this will be the server which will have to process the largest number of packets. The use of higher ring access priority allocated to the gateway will tend to reduce these problems. On the other hand, bringing up the token ring bit rate to 16 Mbps will increase the performance of the system but the utilization rate of the processor responsible for segmenting token ring frames in the gateway will be increased.

For the present simulation study, three scenarios are then examined: 1) non priority mode in token ring, in which upon the token arrival a node (corresponding to a station or a gateway) may transmit only one packet (non exhaustive service); 2) priority mode in token ring, in which gateways have a higher ring-access priority and allowed to empty the queue of the MAC server after having seized the token (exhaustive service); 3) same as 2), but the token ring speeds are brought up to 16 Mbps. For each scenario, the system throughput (bits per second) as a function of the offered load (bits per second) and the mean end-to end delay of delivered data units will be measured. The mean end-to-end delay is defined as the time elapsed between the moment when the data unit packet is inserted in the LLC window at the source station (converted in an I-frame) and the moment when the packet is accepted by the destination station (see Fig. 5.6). The offered load is the reference used and it is expressed in number of bits generated per unit of time by all the token ring stations under the condition that the generation is not halted by backpressure.

The results for the first scenario are shown in Fig. 5.7 to 5.9 (the method considered for validating the results is the one described in appendix C). Figure 5.7 presents the throughput as a function of the offered load. Several window sizes used by the flow control mechanism of the LLC protocol at the end stations are compared. Under low load, there is no noticeable effect of the window size on the throughput which increases linearly. However, under heavy offered load, results differ according to the size of the window. For W=1 (STOP and WAIT protocol [BERT]), the throughput is bound by the interwork system delay corresponding to the round trip delay for each packet and its acknowledgement (service time plus queueing delay). In fact, the speed of the token rings is the throughput bottleneck (the system throughput is bound around 3.8 Mbps) and contributes to the most of the delay. There is no packet rejection in this first case. When the window size is increased, the system throughput increases linearly as a function of the offered load as before until a point (around 3.0 Mbps for W=4, and 2.6 Mbps for W=8) where rejection of frames appears and becomes important. At this point the system is congestion-prone [GER80], i.e, an increment in offered load causes a reduction in throughput. Finally the system stabilizes from a given offered load.

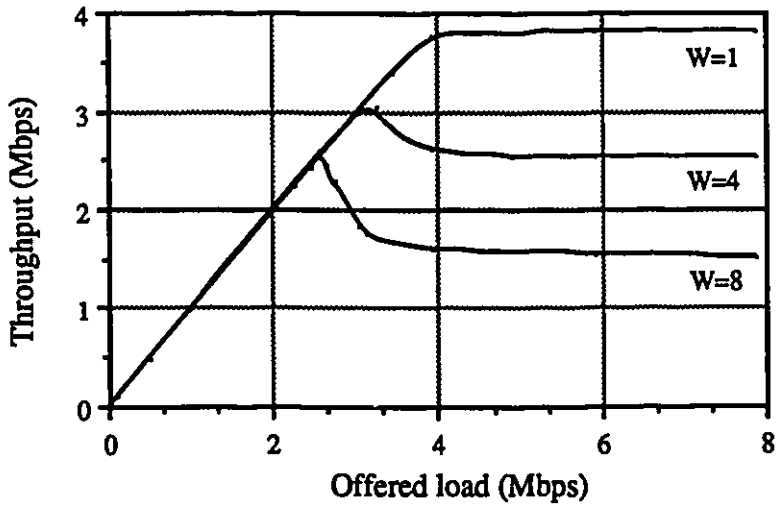


Figure 5.7: System throughput vs offered load (non priority mode in 4 Mbps token rings).

The mean utilization of the buffer used by the gateway for transferring packets from DQDB

to token rings is shown in Fig. 5.8. The retransmission of frames that are discarded due to buffer overflow causes the buffer utilization to increase, which in turn further increases the overflow probability. This positive feedback causes abrupt increases in the buffer utilization and the fraction of the network capacity used for transmission. As a result there is a sudden drop in the throughput. When the throughput is stabilized, the buffer utilization stays around 81%. In Fig. 5.9 shows the mean end-to-end delay versus the system throughput. Under low load, the delay is almost the same for each window size. However, when the system becomes congested, as in the case of W=4 and W=8, the delay increases very steeply. This is caused by the important period of time between the first and the subsequent transmissions of same packets.

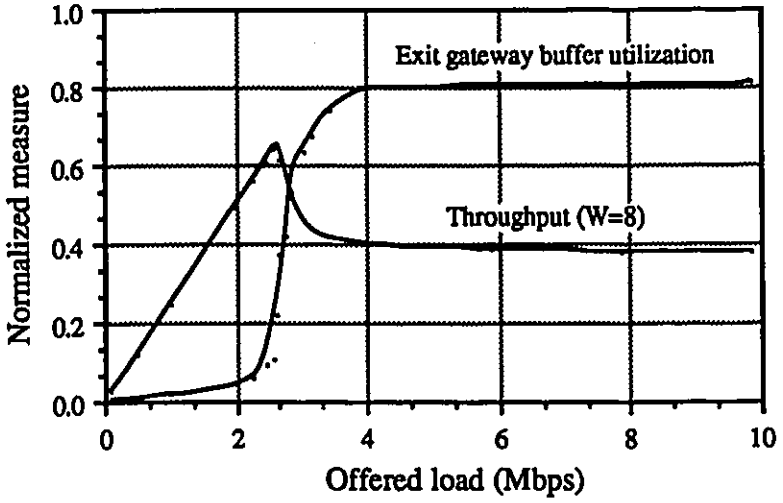


Figure 5.8: Normalized system throughput and gateway buffer utilization vs offered load (non priority mode in 4 Mbps token rings).

Thus the use of larger windows implies that more packets can circulate in the system and causes discarding of packets in the gateways. In this case, a large number of packets need to be retransmitted (even retransmitted three or four times for W=8 after the system throughput is stabilized under heavy load), which decreases the total throughput and increases the mean end-to-end delay.

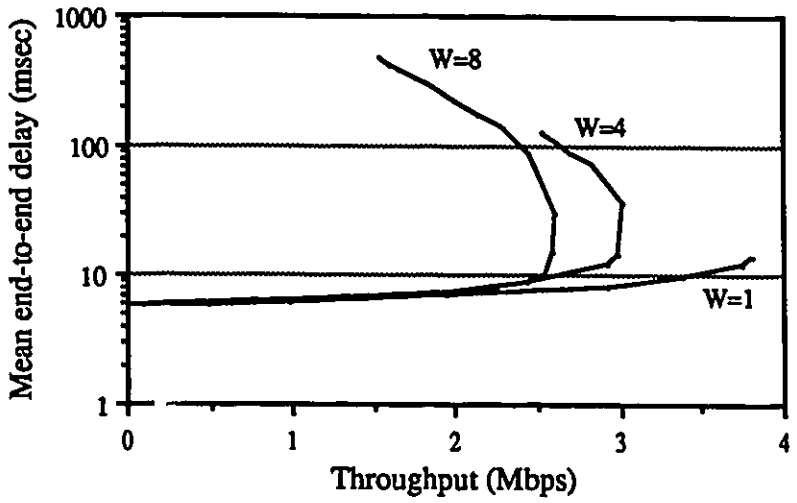


Figure 5.9: Mean end-to-end delay versus system throughput (non priority mode in 4 Mbps token rings).

Furthermore, Fig. 5.7 also shows that the use of backpressure on the application layer generating packets and the use of the LLC windows at the end stations (see Fig 5.6) allow to reduce the lost of frames under congestion prone condition (see the area where the curves are stabilized under heavy traffic). In fact, when the LLC window is closed during an important time since amount of frames have to retransmitted, the generation of data units is restricted at the application layer. In other words, this means that the activity of the application layer can be temporary blocked since the layers under it cannot accept more data units.

Figures 5.10 and 5.11 show the results for the second scenario. For W=1, the same results as above are obtained, but there is a slight decrease of throughput after gaining the saturation point (backpressure effect). For W=4 and 8, significant throughput improvements are obtained by introducing the defined priority mode in the token ring. Gateways are now decongested (in particular the token ring MAC server), avoiding the rejection of packets. Furthermore, the sending of I- and S-frames by the end station MACs is slowed down because gateway MACs, having higher ring-access priority and being able to empty their queues, have a very long token-holding timeout compared to the end station MACs. The long waiting

time before the seizing of the token by the end station MACs acts as a flow control process avoiding congestion problems in the gateways. Moreover, for W=4 and 8, the maximum global throughput obtained is around 3.8 Mbps.This is near of the maximum bit rate supplied by 2 interconnected token rings running at 4 Mbps and in which 100% of the traffic generated is sent to the remote LAN. On the other hand, the throughput obtained for W=4 and 8 is slightly better than for W=1, since the end stations can transmit pending I-frames. Regarding the mean end-to-end delay (Fig. 5.11), values are lower than those in the first scenario (for W=4 and 8), since there is no frame retransmission. When the window size is larger, the mean delay increases because higher traffic load is applied on the interwork system.

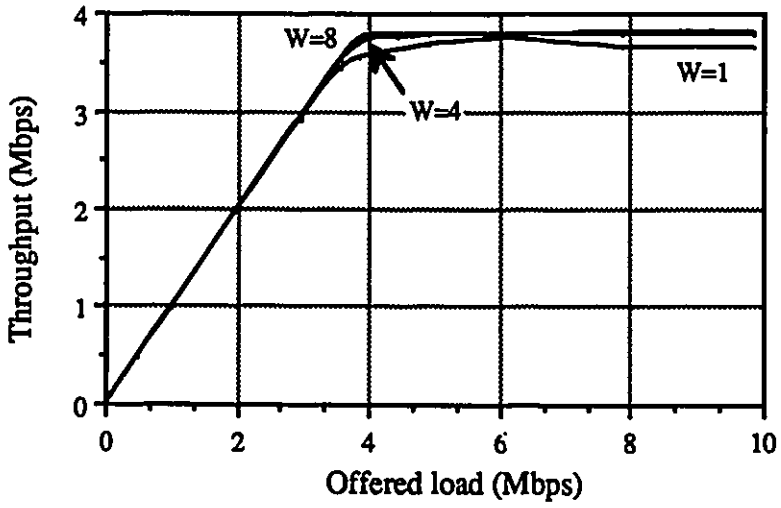


Figure 5.10: System throughput vs offered load (priority mode in 4 Mbps token rings).

In the third scenario, the same characteristics as for the second one are considered but the token ring speed is brought up to 16 Mbps. This increase in speed has a positive effect on the performance of the system (Fig. 5.12 and 5.13). The global throughput is improved and the mean end-to-end delay is considerably reduced. The throughput is now bound around 15 Mbps for W=4 and W=8, and 12 Mbps for W=1. The shape of the curves is similar to those obtained in the previous scenario but better performance is obtained for W=4 or W=8. Indeed, since the global system delay is considerably reduced, larger window size schemes

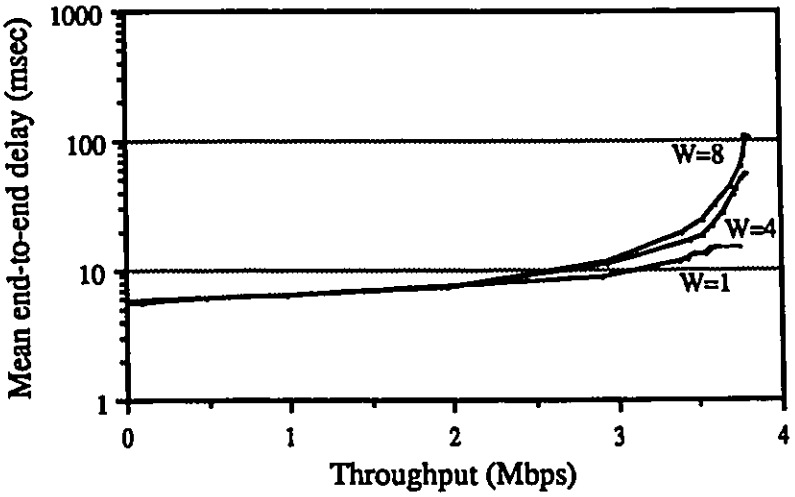


Figure 5.11: Mean end-to-end delay vs system throughput (priority mode in 4 Mbps token rings).

are more efficient. It must be mentioned that in the case of W=8 an utilization rate of 75% is obtained for the token ring frame segmentation processor under heavy load. If faster networks were used instead of 16 Mbps token rings, some congestion problems would occur in the gateway part where the frame segmentation is done. Such situation is represented in Fig. 5.14 and 5.15, where a 32 Mbps token rings is used instead of 16 Mbps. Except for W=1, the increase in the token ring speed has a negative effect on the performance of the system. For W=4 and 8, the system is congestion-prone [GER80] after gaining the saturation point. In fact, by increasing the token ring speed, end station MACs can now send more frames to the gateway and overcharge the processor performing the frame segmentation. Long queues filling up the buffers and causing frame rejections are formed in gateways. Thus increasing the token ring speed is not a way for improving the system performance, except for the case of W=1, since the interwork system delay is considerably reduced compared to the previous scenarios.

Figures 5.16 and 5.17 show the results for the same scenario as the third one (priority mode within 16 Mbps token rings), except that the use of the 32 bytes slots previously defined

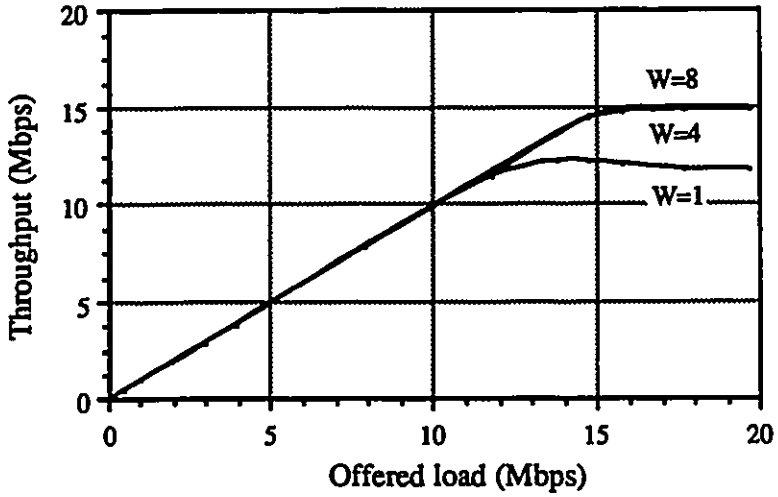


Figure 5.12: System throughput vs offered load (priority mode in 16 Mbps token rings).

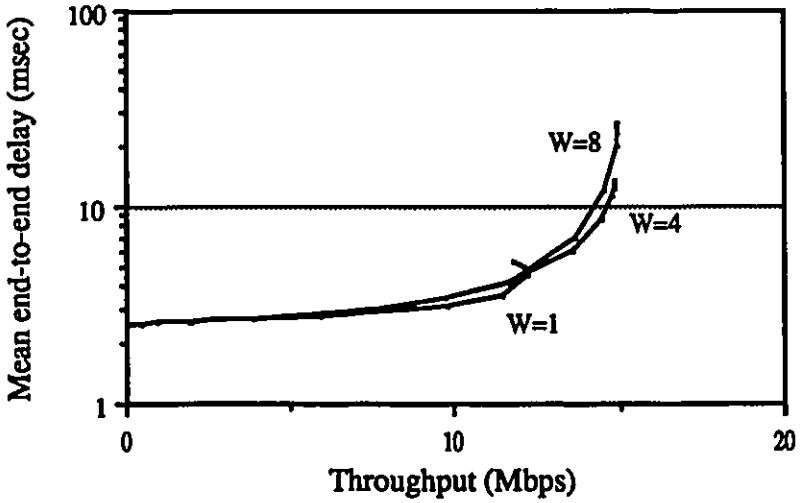


Figure 5.13: Mean end-to-end delay vs system throughput (priority mode in 16 Mbps token rings).

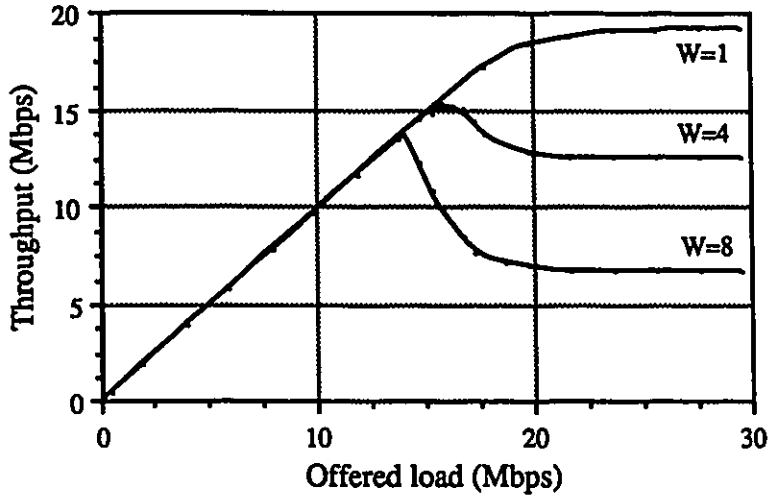


Figure 5.14: System throughput vs offered load (priority mode in 32 Mbps token rings).

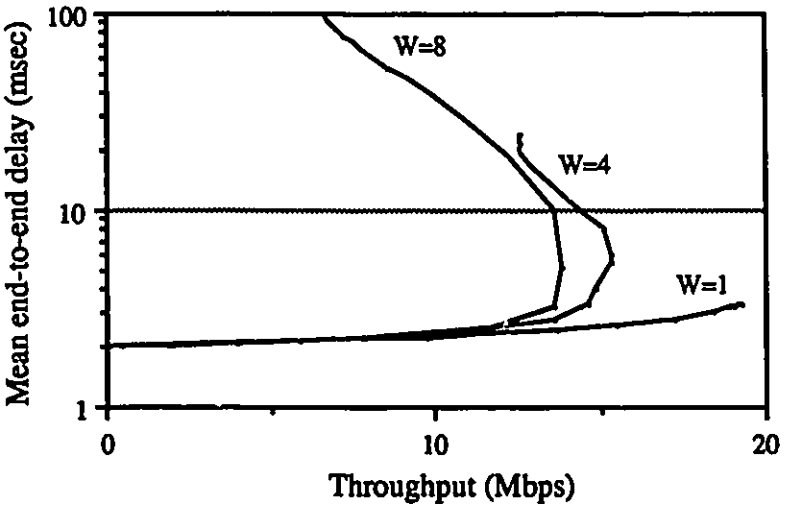


Figure 5.15: Mean end-to-end delay vs system throughput (priority mode in 32 Mbps token rings).

by the IEEE 802.6 committee [IEEE87A] is considered instead of the 69 bytes slot which are presently defined [IEEE88B]. In the gateways, 40 DM PDUs must be now generated for each segmentation of 1 kbytes token ring frames. The results look the same as in the previous situation where 32 Mbps token rings were used (with 69 bytes slots). The processor performing the segmentation of frames, under higher loads, becomes congested, and leads to eventual frame rejections. In fact, by formatting a larger number of smaller DM PDUs, there are more overhead manipulations which causes more processing time for segmenting each token ring frame.

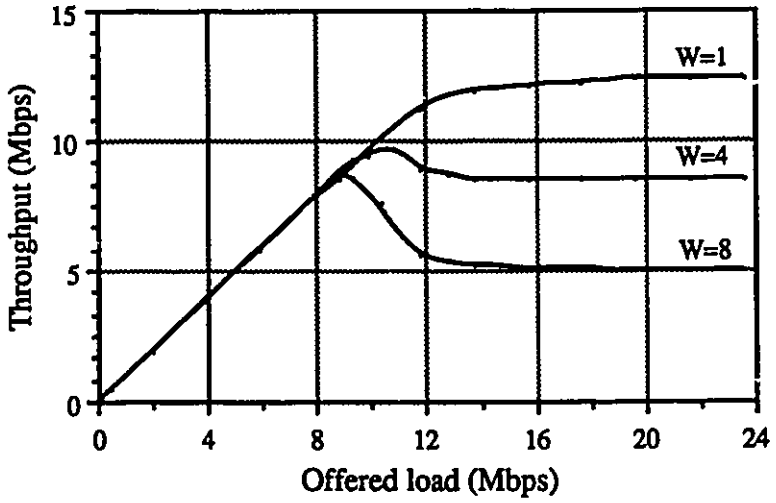


Figure 5.16: System throughput vs offered load (priority mode in 16 Mbps token rings, and 32 bytes slots in DQDB).

5.3.2 Inter-Hospital Image Browsing

The proposed interwork system is now used to provide image browsing between remote hospitals. Each token ring is employed for allowing local traffic in a hospital while DQDB is the backbone network providing inter-communications between hospitals (Fig. 5.18). The system is then evaluated for supporting image browsing performed by medical workstations, in a hos-

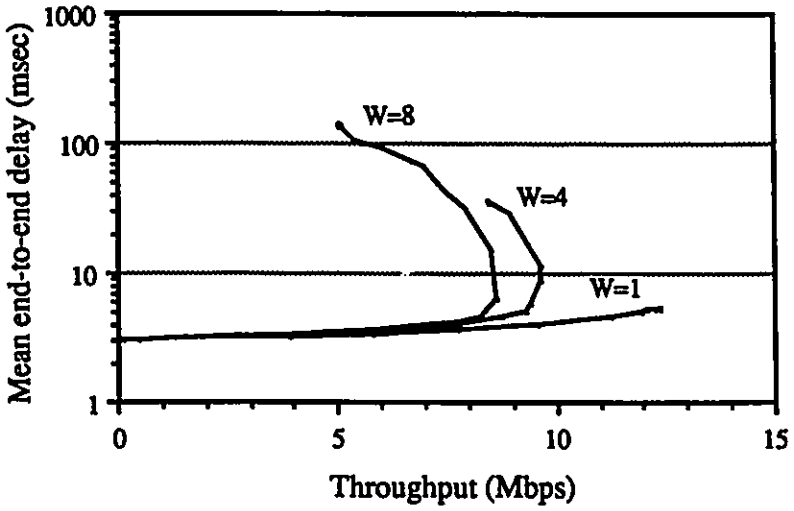


Figure 5.17: Mean end-to-end delay vs system throughput (priority mode in 16 Mbps token rings, and 32 bytes slots in DQDB).

pital, that is linked with a database server situated on a remote token ring serving another hospital. By sending control information, the workstation requests the database server to send the next image or higher resolution version of the current observed image if any progressive image transmission technique is used (as described in chapter 4). At the same time, to simulate requests to the database from other devices, background load is assumed to be generated by other nodes (similar to [HAUS]) which generate image requests to any database in the interwork system (see Fig. 5.18). In this application, the emphasis is put on the elapsed time measured from the time an image browsing request is issued at the measure workstation until the image is displayed on its monitor. Furthermore, the two types of browsing defined in the previous chapter are considered. The first one involves successive transmissions of radiological images having the same size, with a given time between each sending. The second type of browsing uses the concept of progressive image transmission.

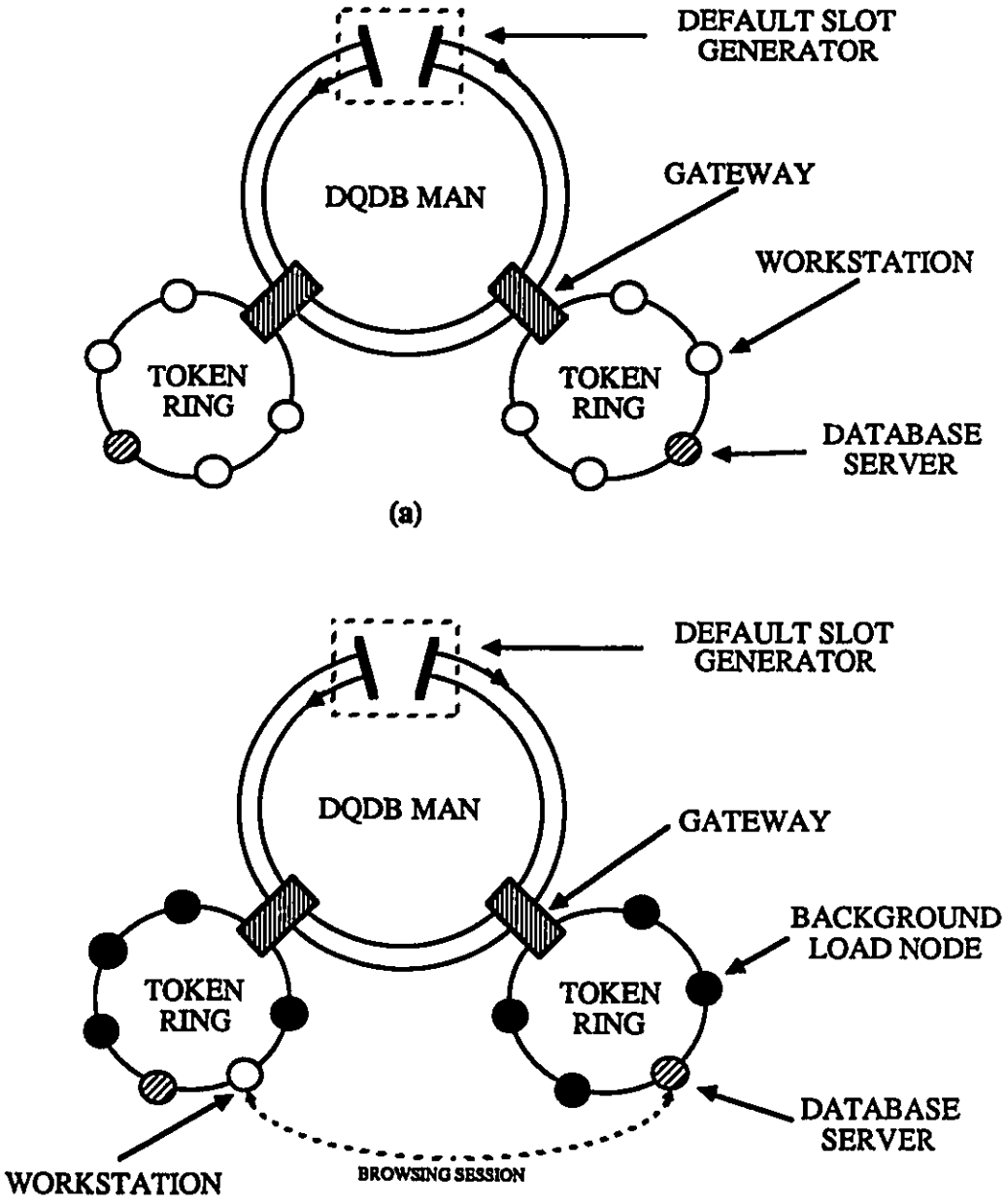


Figure 5.18: (a) Interwork system allowing transfer of information intra and/or inter hospitals; (b) Testbed internetwork configuration where performance of one workstation is evaluated according to the image request generation rate from the background load nodes.

Simulation Model

The QNAP program for simulating the whole interwork system is presented in appendix D.3.2. The same model as in the first interwork system application is used, except that the LLC protocol is simplified at the end stations. Indeed, the process recovering the loss of I-frames is not modelled since sufficient buffer size in the gateways has been assumed in order to avoid frame retransmission and then to keep low the delay between each browsed image. Figure 5.19 shows the models of end stations attached to token rings. After a delay representing the workstation user thinking time, an image request is sent to the database server. Upon receiving the request, the database server retrieves the image from the disk, and sends its segments to the network adapter. A single queue is used to represent all these operations. In the case of browsing incorporating the progressive image transmission technique, a timer must be included in the database model. Thus, once the database has completed the transmission of the reduced-resolution, it initializes the timer. If the database does not receive any request before the timer expires, it will send the next pyramid level of the currently observed image (see Fig. 4.2 and 4.4). Otherwise, the database server will send the next browsed image. At the network adapter, the LLC and MAC layers are simulated. After being processed by the LLC, image segments, now formatted as I-frames, are queued at the MAC for eventual transmission onto the token ring.

Figure 5.19 also shows the model of a typical background load node. This model is similar to the one of the workstation, except that the upper layers are simply represented by a source since the background load node continually request images at a rate selected at the beginning of the simulation. This means that the background load node does not wait to receive images before sending the next request [HAUS]. Thus only the performance of the defined "workstation" browsing images is evaluated.

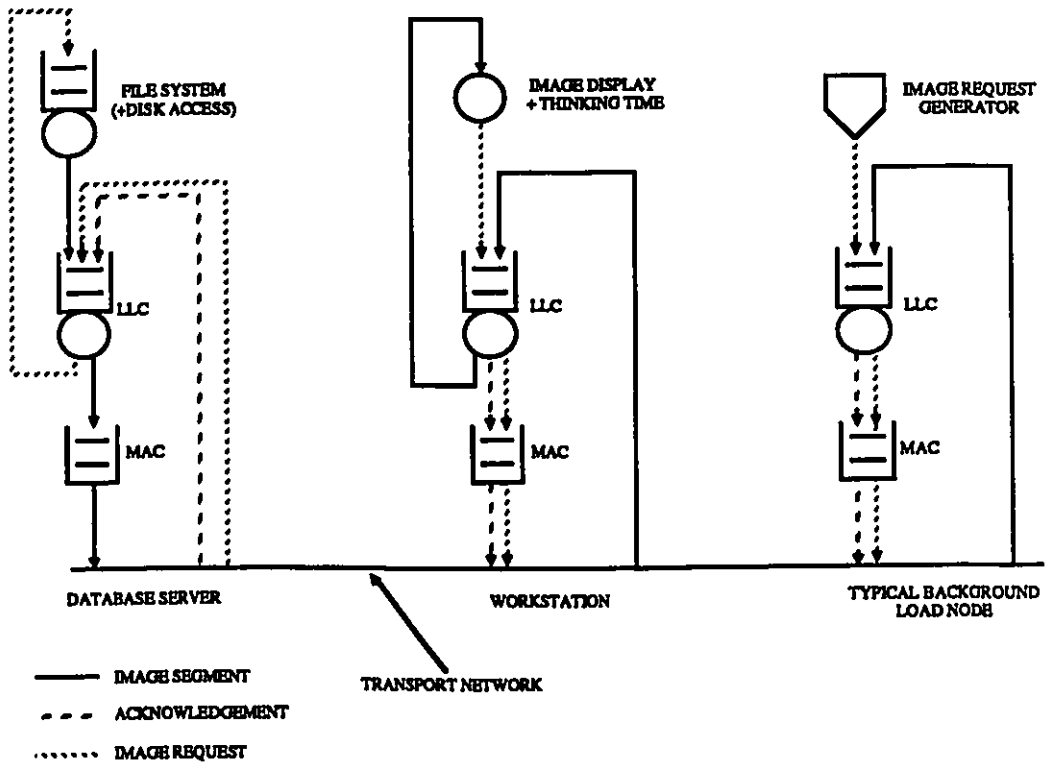


Figure 5.19: Models to simulate the database server, the workstation and the background load node.

Assumptions

The assumptions established in the first application are still valid. However, only a LLC window size of 8 and a bit rate of 16 Mbps for token rings are assumed. The presence of some other database servers and background load nodes which are symmetrically distributed in the system is assumed (see Fig. 5.18). In the database servers, the same system and assumptions defined in section 4.3.2 are considered.

The thinking time of the workstation is still assumed to be exponentially distributed with a mean of 2 seconds. This delay represents the time taken by the workstation user to observe the browsed image and to ask for the next image. In the case of browsing including progressive image transmission, the same thinking time is assumed, but if after 5 seconds the database does not receive any request from the workstation, it will send the next higher resolution of the image. At the background load nodes, the time interval between requests is also assumed exponentially distributed, and several values for each system configuration are compared. The total background load on each token ring is distributed among five nodes. Furthermore, 20% on the requests are sent to database servers situated on the remote token ring, since most of the image requests are done to the local database servers under realistic situations.

Analysis and Results

In this second application, the database server is now the major source of delay. As previously mentioned, the processing time of an image of 1000 x 1200 pixels x 8 bits, has been evaluated to be 1.5 seconds, which means that 40 images/minute can be sent by one database server. The database server is then able to process 781.3 1 bytes packets per second (see table 5.1). On the other hand, the minimum time between each image requested by a workstation is represented by Eq. (5.1), where T_{proc} and $T_{thinking}$ represent respectively the time for processing the image at the database and the time needed to display and read the image at the workstation. Hence, the workstation may request an average of 17.1 images/minute.

$$T_{between\ requests} = T_{proc} + T_{thinking} = 3.5\ sec \tag{5.1}$$

$$\lambda_{req} = 60/T_{between\ requests} = 17.1\ req/min \tag{5.2}$$

In the case where only one database server per token ring is used, this server will bring significant display delay under significant generation of image requests by the background load nodes, since the reading and segmentation of an image in the database are the most time-consuming processes in the system. However, adding one or more database servers on each token ring might eventually saturate the token rings.

The next paragraphs present the simulation results for the interwork system allowing image file transfers between database servers and medical workstations. Two scenarios are considered: 1) image browsing, in which fixed (same) size images are sequentially sent; 2) image browsing with the use of progressive image transmission (in which reduced-resolution images are browsed with the possibility, for the workstation user, to increase the resolution of an image that captures his attention). For each scenario, the average time for displaying the requested image at the workstation is measured for various number of image requests generated by the background load nodes, as well as the number of database servers on each token ring. The configuration, in which higher ring access priority and exhaustive service are allocated to the gateway MACs sending packets on the token rings, is also examined.

Figure 5.20 shows the results of the first scenario where non priority mode is used in token rings (the method considered for validating the results is the one described in appendix C). The mean image display at the measured workstation is plotted versus the image request rate generated by all the background load nodes on each token ring. In the case where one database server per token ring is used (DS = 1, in Fig. 5.20), this server is the direct bottleneck of the system. This explains why the mean image display delay increases sharply after 33 req/min. By increasing the number of database servers on each token ring (i.e., distributed database), the system performance is improved and it is now bound not only by the database service

time but also by the interwork system delay. In fact, the gateway MAC sending packets on the token ring is now slowing down the sending of frames by the database server through the LLC protocol. For example, the system cannot serve more than 75 req/min per token ring with 3 database servers (DS = 3) compared to 39 with one database server (DS = 1) per token ring. The results when higher priority mode for gateways is used in token rings are shown in Fig. 5.21. The results look almost the same as in the previous scenario, but they are in fact slightly improved since the queueing delay in the gateway has been reduced by giving it higher priority in sending packets onto the token rings. For instance, for 3 database servers (DS = 3) per token ring, a load of 81 req/min can now be satisfied instead of 75 req/min. On the other hand, for the workstation user performing image browsing, the mean image display delay must be as low as possible. Then, if a delay less than 5 seconds is acceptable, then, by considering globally both cases of priority modes, a background load up to 30 req/min can be accepted by the system comprising one database server per token ring, 50 with 2 database servers, and 56 with 3 database servers (60 under priority mode in token rings).

Figure 5.22 corresponds to browsing activities including progressive image transmission, where images compressed according to the reduced-pyramid data structure are used. The curves give the mean image display delay for each pyramid level as a function of the background load traffic generated by remote stations. The same scenario as the one defined in section 4.3.3 is considered. Images of 1024 x 1280 pixels are used, which means that the first and second reduced-resolution are 256 x 320 and 512 x 640 pixels. The workstation user, currently performing a browsing session, initially browses images of 256 x 320. If a particular image captures his attention, the user can request a higher resolution for this image. It is assumed that the user decides, with a probability of 20%, to bring up the resolution of the image to 512 x 640. Then, having read the improved image, the user decides with a probability of 80% to receive the data representing the full resolution image of 1024 x 1280. Background load requests are directly made for images of 1024 x 1280 (all pyramid levels being sent in sequence). In Fig. 5.22, results are shown for the case where one database server per token ring is used. The performance is quite improved compared to the previous scenario where full

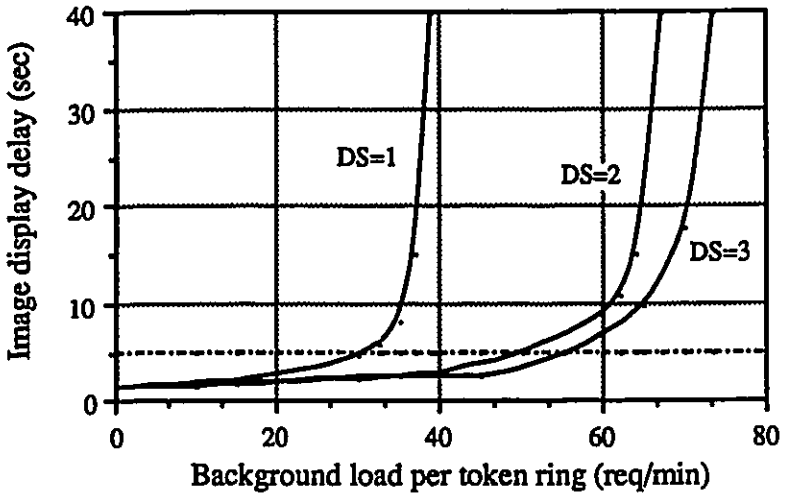


Figure 5.20: Average image display delay at the workstation as a function of background load (non priority mode for the gateways in 16 Mbps token rings).

resolution images were used, since mean image display delay is quite reduced for the same background load. In fact, the mean image display delay, for each pyramid level, is quite low compared to the first scenario, because fewer segments are needed to represent each level. Hence, this reduces the global service and waiting time at the database servers. For instance, in order to obtain delays shorter than 5 seconds between browsed images (or reduced-resolution images), background load up to 62 req/min can now be accepted by the system. Figure 5.23 shows the same scenario where 2 database servers are present on each token ring. The delay between browsed reduced-resolution images (or images) is acceptable until the load reaches 120 req/min on each token ring. For higher load, the mean image display delay increases sharply.

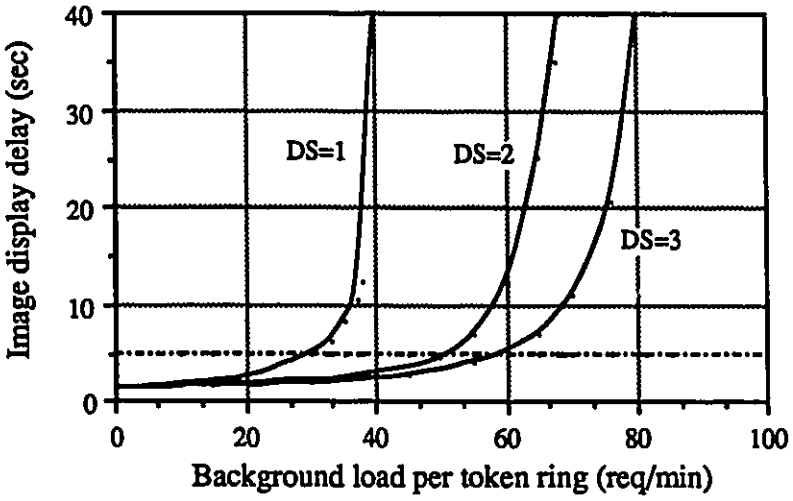


Figure 5.21: Average image display delay at the workstation as a function of background load (priority mode for the gateways in 16 Mbps token rings).

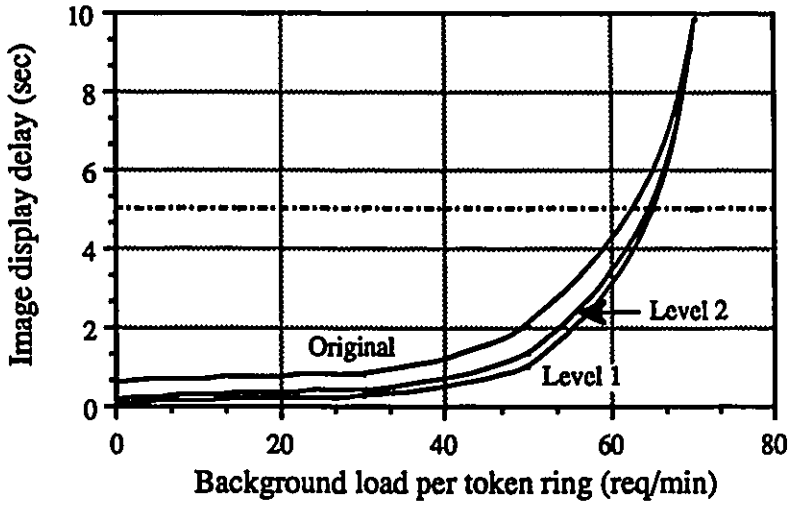


Figure 5.22: Average image display delay at the workstation as a function of background load (using progressive image transmission and priority mode in token rings, and having 1 database server per token ring).

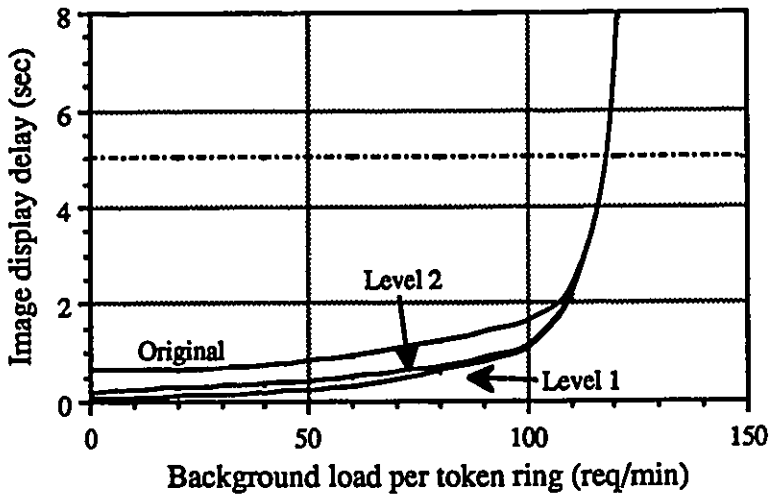


Figure 5.23: Average image display delay at the workstation as a function of background load (using progressive image transmission and priority mode in token rings, and having 2 database servers per token ring).

5.4 Concluding Remarks

This chapter has presented the performance of an interwork system using the proposed IEEE 802.6 DQDB MAN standard as a backbone network interconnecting two token rings. Such a system introduces the problem of interconnecting networks which do not use the same access protocol and maximum packet length. Gateways must be used to provide protocol translation between both types of networks and perform the segmentation and reassembly of packets. Besides, two modes of priority in token rings have been considered : lower priority for each token ring node and higher ring-access priority for the gateway MAC (with exhaustive service). Furthermore, the system has been evaluated for two different applications. The first application was the transfer of packets between homogeneous stations situated on different token rings. It has been found that under low offered load, the size of the LLC window does not affect the performance, but it does under higher offered load. In non priority mode,

larger window sizes worsen the performance. The use of priority mode for the gateways allows getting better performance, mainly because congestion problems are considerably reduced in the gateway. Increasing the token ring bit rate in priority mode improves considerably the performance, in particular for larger window sizes.

The second application was one of heavy traffic: performing radiological image browsing within a medical environment. The performance of a workstation doing a browsing session with a database server situated on a remote token ring was evaluated compared to given background load traffic generated by other stations. Original and compressed images have been employed. The reduced-difference pyramid technique has been chosen as a compression method. In addition to facilitating the manipulation of images, this technique allows performing progressive image transmission. Such transmission allows the workstation user to browse over a set of reduced-resolution images with the option of halting on images in order to increase their resolution. Results have showed that the database servers are obviously a source of significant delay, but with the presence of distributed database servers, system throughput is also bound by the congestion in the gateways. Moreover, the use of the progressive image transmission reduces global system load and provides lower image display delay.

The simulation results have shown that the proposed DQDB MAN performs well in interconnecting slower LANs. It appears, however, that congestion problems in the gateways can be the source of important delays. The major problem has been the queueing delay at the gateway MACs having to send packets on the token rings. Hence, for reducing congestion in this server, the allocation of higher ring access priority and exhaustive service for the gateway MACs has been proposed. Another observed problem has been the time for segmenting the token ring frames into derived DQDB MAC PDU, in particular when the speed of the interconnected low-speed LANs have a bit rate higher than 16 Mbps. The segmentation process then becomes the system bottleneck. This problem can become particularly significant if it is planned to interconnect DQDB with another high speed network like FDDI-II which has been shortly described in chapter 2.

Chapter 6

Conclusions and Suggestions for Further Research

This thesis has presented performance studies of multimedia communications systems supported by the IEEE 802.6 DQDB MAN. DQDB was selected because it was recently proposed by the IEEE 802.6 committee [IEEE87B] and is defined as a high speed network integrating circuit and packet switching. The use of DQDB has been particularly evaluated for transferring high resolution images and for interconnecting low-speeds LANs over packet switching mode.

In chapter 3, a model has been proposed for simulating the "distributed queue" defined by the IEEE 802.6 DQDB MAN standard for performing the QA slot access control. The simulation results have confirmed the main characteristics of DQDB observed in other performance studies [CONTI,DAVIDS,NEW86,NEW88A]. A first one is that the DQDB "distributed queue" can be approximated by the M/D/1 queueing system (plus some delays due the slotted structure and the presence of NA slots in DQDB). However, it has also been seen that the M/D/1 approximation neglects the fact that, under heavy load conditions, there is unfairness between the nodes located within DQDB. Indeed, nodes adjacent to the heads of bus are favored. Another important characteristic of DQDB is that its performance is not significantly affected by important variations of network size.

In chapter 4, the use DQDB has been evaluated for supporting radiological image brows-

ing applications between database servers and medical workstations distributed over different hospitals. Results obtained through simulation have showed that DQDB is a good candidate since it can provide high bandwidth. In fact, it has been observed that that delay problems are related to the methods used to store and transfer images. Hence, in addition to full resolution images, the use of compressed images has been considered. Particularly, the technique called reduced-difference pyramid [WANG], which also allows performing progressive image transmission, has been examined.

In chapter 5, the use of DQDB for interconnecting IEEE 802.5 token rings via gateways has been considered. The simulation results have showed that DQDB performs well as a backbone network interconnecting LANs. It appears, however, that congestion problems in the gateways can be the source of significant delays under heavy load conditions. In fact, two kinds of congestion have been observed. The first one occurs when the speeds of the interconnected LANs are smaller than the speed of DQDB, and when the service times of the processors performing the segmentation and/or reassembly of frames in the gateways is less than the time to transmit the frames onto the interconnected LANs. In this situation, congestion occurs in the part of the gateways performing the sending of frames on the low-speed LANs, since these LANs become saturated under heavy load conditions. For overcoming this congestion, it has been proposed to allocate a higher access priority and exhaustive service for the gateway MAC procedures sending frames on the interconnected LANs. By doing this, frame rejections in the gateways have been avoided. At the same time, the allocation of a higher priority to the gateways allows throttled the generation of frames by the end stations since, most of the time, the interconnected LANs transfer traffic coming from the gateways. Another observed congestion problem in the gateways has occurred when the speed of the interconnected LANs is increased. The higher the LAN speed is, the more the processor segmenting frames is overloaded. The need to segment the frames coming from the LANs into short slots in order to transmit information through DQDB can then be a drawback. In particularly, this can be an important limitation if it is planned to interconnect DQDB with another high speed network, such as FDDI, which uses a different maximum packet size.

As follow up studies, it can be proposed further evaluations of DQDB used as a backbone network for interconnecting LANs. For instance, further work can be done for evaluating the optimum buffer size in the gateways in order to avoid packet rejections there. Scenarios involving various kinds of traffic simultaneously applied on the interwork system can also be examined. On the other hand, it would be appropriate to evaluate the interconnection of DQDB with ISDN and B-ISDN, since DQDB is also defined for supporting the services offered by these networks.

Bibliography

[AHUJA] S. R. Ahuja, J. R. Ensor and D. N. Horn, "The Rapport Multimedia Conferencing System", in *Conf. on Office Information Systems*, Palo Alto, California, USA, Mar. 1988.

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

[ANSI85A] ANSI/IEEE Standard 802.2, "Logical Link Control", IEEE Project 802, IEEE, New York, 1985.

[ANSI85B] ANSI/IEEE Standard 802.3, "CSMA/CD Access Method and Physical Layer Specification", IEEE Project 802, Local Area Network Standards, IEEE, New York, 1985.

[ANSI85C] ANSI/IEEE Standard 802.4, "Token Bus Access Method and Physical Layer Specification", IEEE Project 802, Local Area Network Standards, IEEE, New York, 1985.

[ANSI85D] ANSI/IEEE Standard 802.5, "Token Ring Access Method and Physical Layer Specification", IEEE Project 802, Local Area Network Standards, IEEE, New York, 1985.

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

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

[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, pp. 1465-1473, Oct. 1981.

[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.

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

[CCITT88] CCITT Recommendation I.122, "Framework for Providing Additional Packet-Mode Bearer Services", July 1988.

[CCITT89] CCITT Expects of Study Group XVIII, "Report of the San Diego Meeting - Elaboration of a Series of Recommendations on B-ISDN", San Diego, USA, Jan. 1989.

[CCITT_E] CCITT Recommendation E.164, "Numbering Plan for the ISDN Era.".

[CONTI] 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.

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

[DOD83A] Department of Defense (USA), "Military Standard Internet Protocol", MIL-STD-1777, August 12, 1983.

[DOD83B] Department of Defense (USA), "Military Standard Transmission Control Protocol", MIL-STD-1778, August 12, 1983.

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

[FORS] H. Forsdick, "Explorations into Real-Time Multimedia Conferencing", in *Computer Message System-85*, R. Uhlig, Ed. Elseweiser Science, North-Holland, 1985, pp. 331-347.

[GEOR] N. D. Georganas, M. Goldberg, L. Orozco-Barbosa and J. Mastronardi, "A Multimedia Communications System for Medical Applications", *Proc. ICC'89*, Boston, USA, June 1989.

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

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

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

[HAUS] S. E. Hauser, M. I. Felsen, M. J. Gill and G. R. Thoma, "Networking AT-Class Computers for Image Distribution", *IEEE JSAC*, vol. 7, no. 2, pp. 268-275, Feb. 1989.

[HUANG] H. K. Huang, Elements of Digital Radiology - A Professional Handbook and Guide, *Prentice Hall Inc.*, New Jersey, 1987.

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

[IEEE85] Draft Standard IEEE P802.6 Metropolitan Area Network, "Media Access Control Specifications", IEEE 802.6 Document 802.6/85-D, Revision D, August 1985.

[IEEE87A] 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.

[IEEE87B] Minutes from the Nov 9-13, 1987 meeting of IEEE 802.6 Metropolitan Area Network Working Group.

[IEEE88A] Minutes from the March, July and Sept. 1988 meetings of IEEE 802.6 Metropolitan Area Network Working Group.

[IEEE88B] 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.

[IEEE89] Draft Standard IEEE P802.9, Integrated Voice/Data LAN Interface, "Media Access Control (MAC) and Physical Layer Specifications", Draft D3, Mar. 1989.

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

[JAIN] A. Jain, "Image Data Compression: A Review", *Proc. of the IEEE*, vol. 69, no. 3, pp. 349-391, March 1981.

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

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

[KARM] A. Karmouch and Nicolas D. Georganas, "Multimedia Document Architecture for Medical Applications", *Journal of Digital Imaging*, vol. 2, no. 2, May 1989.

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

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

[KOSIT] R. Kositpaiboon, P. Tsingotjidis, Luis Orozco-Barbosa and Nicolas D. Georganas, "Packetized Radiographic Image Transfers over Local Area Networks

for Diagnosis and Conferencing", *IEEE JSAC*, vol. 7, no. 5, pp. 842-856, June 1989.

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

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

[MAYD] U. M. Maydell, P. S. Gill and W. A. Davis, "Design Criteria for an Interactive Image Computer System", *IEEE ELECTRONICOM'85*, Toronto, Canada, Oct 1985, pp. 282-285.

[MCGUF] L. J. McGuffin, L. O. Reid and S. R. Sparks, "MAP/TOP in CIM Distributed Computing", *IEEE Network*, vol. 2, no. 3, pp. 23-31, May 1988.

[MOLL] J. F. Mollenauer, "Status of the IEEE 802.6 MAN Standards", *19th Conf. on Local Computer Networks*, Minneapolis, USA, 1988, pp. 166-169.

[MOUG] 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, pp. 20-28, April 1988.

[PANG88] 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.

[PANG89] J. Pang and F. A. Tobagi, "Generalized Access Control Strategies for Token Passing Systems", *Proc. INFOCOM'89*, Ottawa, Canada, April 1989, pp. 332-341.

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

[REIS75] M. Reiser and H. Kobayashi, "Queueing Networks with Multiple Closed Chains: Theory and Computational Algorithms", *IBM Jour. Research Development*, vol. 19, pp. 283-294, May 1975.

[REIS80] M. Reiser and S. S. Lavenberg, "Mean-value Analysis of Closed Multichain Queueing Networks", *Journal ACM*, vol. 27, no. 2, pp. 313-322, April 1980.

[ROSS] F. E. Ross, "FDDI - A Tutorial", *IEEE Communications Magazine*, vol 24, no. 5, pp. 10-17, May 1986.

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

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

[SARIN] S. Sarin and I. Geif, "Computer-Based Real-Time Conferencing Systems", *IEEE Computer*, vol. 18, no. 10, pp. 33-45, Oct. 1985.

[SAUER] C. H. Sauer and K. M. Chandy, Computer Systems Performance Modeling, *Prentice-Hall Inc.*, Englewood Cliffs, New Jewsey, 1981.

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

[SCOTT] C. J. Scott, P. G. Potter and A. M. Day, "Broadband Network Development in Australia", *GLOBECOM'88*, Hollywood, Florida, USA, Nov. 1988, pp. 374-379.

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

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

[TANI] S. L. Tanimoto, "Image Transmission with Gross Information First", *Computer Graphics and Image Processing*, vol. 4, pp. 104-119, 1975.

[THOM] A. Thomas, J. P. Coudreuse and M. Servel, "Asynchronous Time-Division Techniques: An Experimental Packet Network Integrating Videocommunication", *ISS'84*, Florence, Italy, May 1984, 32C.

[TURN] J. Turner, "Fast Packet Switching and Visual Communications", *MULTIMEDIA'89, 2nd IEEE ComSoc Int. Workshop on Multimedia Communications*, Montébello, Québec, Canada, April 1989.

[VAL89A] Richard Vallée, Luis Orozco-Barbosa and N. D. Georganas, "Modeling and Simulation of Multimedia Communications Networks", *SPIE'89, Medical Imaging III*, Newport Beach, California, USA, Jan. 1989.

[VAL89B] 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.

[VAL89C] Richard Vallée, Luis Orozco-Barbosa and N. D. Georganas, "Performance of Token Rings Interconnected by IEEE 802.6 DQDB MAN", submitted to the *IEEE Trans. On Communications.*

[WAIN] 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.

[WANG] L. Wang and M. Goldberg, "The Reduced-Difference Pyramid: A Data Structure for Progressive Image Transmission", *Optical Engineering Journal*, vol. 28, no. 1, pp. 708-716, July 1989.

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

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

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

[ZUK88] Moshe Zukerman, "Overload Control of the Isochronous Traffic in QPSX", *GLOBECOM'88*, Hollywood, Florida, USA, Nov. 1988, pp. 1241-1245.

Appendix A

Overview of the IEEE 802 MAC PDUs

This appendix summarily presents the MAC PDUs (media access control protocol data units) defined by the IEEE 802 standards specified at the MAC and physical layers of the OSI Reference Model. These standards have been presented in chapter 2, section 2.2.

A.1 IEEE 802.3 CSMA/CD

Figure A.1 shows the MAC frame format used by the CSMA/CD protocol [ANSI85B]. The preamble field is used to allow the PLS (Physical Signaling) circuitry, comprised in the physical layer, to reach its steady-state synchronization with the received frame timing. Following is the start frame delimiter (SFD) field which is the sequence 10101011 and which indicates the beginning of the CSMA/CD frame. Then, there are the address fields: destination and source addresses (SA and DA). Individual and group (multi-cast or broadcast) addresses can be used. Next is the the length field (LI) whose value indicates the number of bytes in the data field. If its value is less than the minimum (slot time) required for proper operation of the protocol, a pad field (a sequence of random octets) will be added at the end of the data field but prior to the FCS field (the equation calculating the pad size is shown in Fig. A.1). In fact, the minimum frame length (equivalent to the slot time mentioned in section 2.2.1) shall be long enough in order to get reasonable system performance [BUX81].

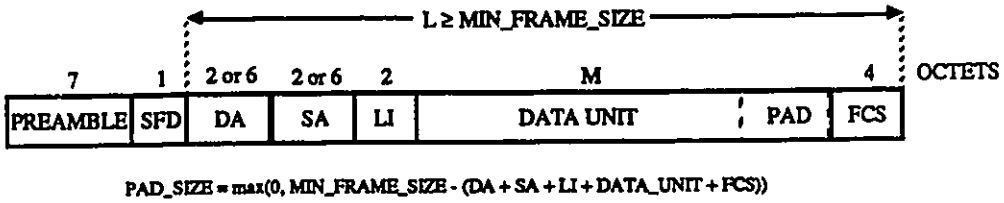


Figure A.1: CSMA/CD frame structure.

The data unit field contained in the CSMA/CD frame is filled by a PDU coming from the LLC sublayer (the LLC PDU is described in section 2.2.6). Finally, the last 4 octets are the FCS (frame check sequence). A cyclic redundancy check (CRC) field is used by the transmit and receive algorithms to generate a CRC value for the FCS field. This value is computed as a function of the contents of the DA, SA, LI, data unit, and pad fields. The encoding is defined by the generating polynomial represented by the following equation:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

A.2 IEEE 802.4 Token-Passing Bus

Figure A.2 shows the MAC frame format defined for the IEEE 802.4 token-passing bus standard [ANSI85C]. The preamble field is used not only for allowing signal level and phase lock synchronization (like for the CSMA/CD protocol) but also to guarantee a minimum ED to SD time period allowing stations to process the frame previously received. Thus the preamble length is a function of the bit rate used on the medium (longer length for higher bit rates). After the frame start delimiter (SD) field, there is the frame control (FC) field which contains information about frame classes (MAC control or data frames), frame types (for MAC control frames) and service class priority (for data frames). Then there are the destination and source address (DA and SA) fields which have the same structure than the one defined by the IEEE 802.3 standard. Such as data unit fields, there are the MAC management data frames which are used to exchange information between MAC management entities, and the LLC PDU.

Following, there is the FCS field, which is generated by the same method defined in the IEEE 802.3 standard, and whose CRC value is computed as a function of the contents of the FC, DA, SA, and data unit fields. Finally, the IEEE 802.4 frame is ended by the end delimiter (ED) field which contains bits signaling if more transmissions from the station will follow and if there is a FCS error within the frame.

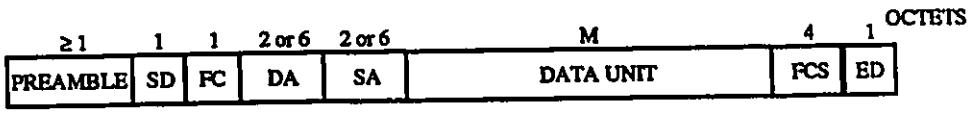


Figure A.2: Token-passing bus frame structure.

A.3 IEEE 802.5 Token-Passing Ring

Figure A.3(a) shows the structure of the frame used for transmitting data information onto the IEEE 802.5 token-passing ring [ANSI85D]. After the start delimiter (SD) field, there is the access control (AC) field which contains 1 bit indicating if the frame is the token, 3 others indicating the priority of the token and 3 more, called RRR bits, which are used by the stations having high priority frames to request (in frames or tokens as they are repeated) that the next token be issued at the requested priority (this is discussed in section 2.2.3). The frame control (FC), address (DA, SA), data unit, FCS and end delimiter (ED) fields are similarly defined as in the case of the IEEE 802.4 standard. Finally, at the end of the IEEE 802.5 frame, there is the frame status (FS) field used for indicating that the destination station has received and copied the frame. The token format, which only contains the SD, AC and ED fields, is presented in Fig. A.3(b) .

A.4 IEEE 802.6 DQDB MAN

The DQDB MAC PDU structure [IEEE88B] is presented in Fig. A.4. It comprises the address (DA,SA), the protocol identification (PI), the quality of service (QOS), the bringing (BR),

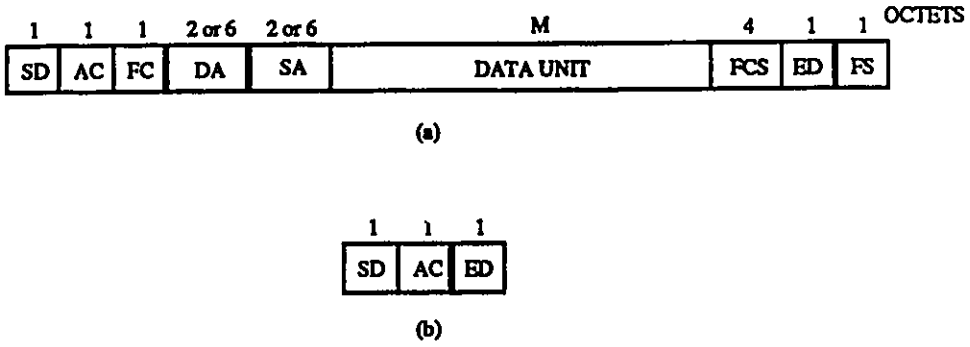


Figure A.3: a) Token-passing ring frame format; b) Token format.

the length indicator (LI), the MAC PDU header checksum (MHC), the information and the information check sequence (ICS) fields. The address fields can be used in the same way than in the other IEEE 802 MAC standards, but they have a longer length (8 octets instead of 6). In fact, by using 8 octets, 60 bits MSAP (MAC service access point) addresses can also be used to carry addresses coded according to CCITT recommendation E.164 [CCITT_E] (intended to be used within ISDN environments). The BR field is reserved for future use with MAC level bridging. The LI field indicates the length of the entire MAC PDU which shall not exceed 9216 octets. The ICS field is generated in the same way than for the FCS defined in the other IEEE 802 MAC and physical layer standards.

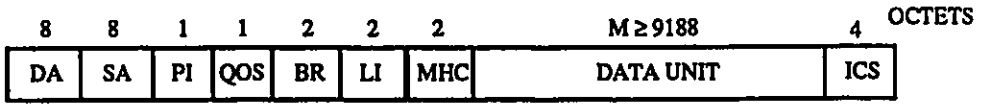


Figure A.4: DQDB MAC PDU format.

A.5 IEEE 802.9 IVD LAN Interface

The IEEE 802.9 committee is currently working to define a MAC layer supporting IEEE 802 LANs, but there is no concrete proposal at this time.

Appendix B

Overview of QNAP2

This appendix gives an overview of QNAP2 (Queue Network Analysis Package) [QNAP2] which can be defined as a system for describing, handling and solving queueing network models. QNAP2 is comprised of a specification language (based on Pascal) which is used for the description of the models under study and the control of their resolution, and of several resolution modules, or solvers, implementing the different algorithms currently available. In particular, QNAP2 incorporates known analytical solutions of queueing networks such as the MVA (mean value analysis) [REIS80] and convolution algorithms [BUZEN,REIS75]. Moreover, it incorporates a discrete-event simulator.

In QNAP2, a queueing network consists of a set of stations (each typical station comprising one server and one queue) through which customers circulate according to given rules. The processing done by each station may be described by a simple time duration or by a complex algorithm which may include synchronization operations. An example of a small network, evaluated by using QNAP2, is presented in Fig. B.1 and B.2. Figure B.1 shows the queueing model of this network, while Fig. B.2 gives the QNAP program used to evaluate the performance of the system. In this example, the QNAP2 discrete-event simulator is used to model the system. For each station, some parameters have to 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; and finally the "next station" where the customer in service is going to be moved (using the TRANSIT command).

In the service part, the effective service time of the server can be represented by commands such as EXP(t) (for exponential service time) and CST(t) (for constant service time). For instance, the station "X", represented as a source in Fig. B.1, will generate customers with a time exponentially distributed between each customer generation.

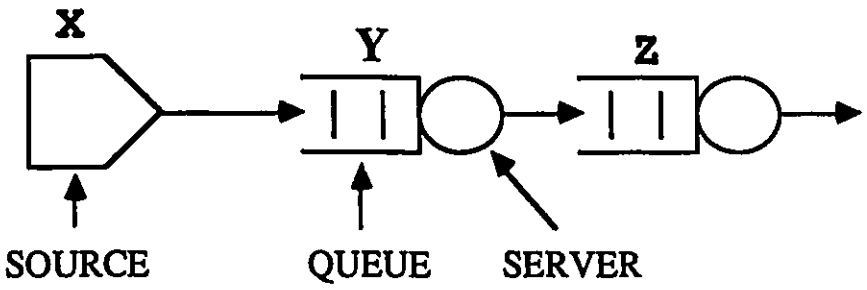


Figure B.1: Model of a small computer system.

```
$RUN QNAP02
\DECLARE\
REAL      arrival = 200,
          cpu_time = 5E-3;
QUEUE     x,y,z;

\STATION\ NAME = x;
TYPE = SOURCE;
SERVICE = BEGIN
              EXP(1/arrival);
              TRANSIT(y);
            END;

\STATION\ NAME = y;
TYPE = SERVER;
SCHED = FIFO;
SERVICE = BEGIN
              CST(cpu_time);
              TRANSIT(z);
            END;
.....

\CONTROL\ TMAX = 1;   & simulation duration in second
\EXEC\ SIMUL
\END\
```

Figure B.2: Simulation program of the computer system presented in the previous figure.

The results of the simulation are represented in a form of table (see Fig. B.3). For each station, the results comprise the mean service time, the utilization, the mean number of customers queued and in service, the mean response time and the number of customers served from the beginning of the simulation.

```
*** SIMULATION ***

... TIME =      1.00
*************************************************************************
*  NAME  * SERVICE * BUSY PCT *  CUST NB * RESPONSE *  SERV NB  *
*************************************************************************
*        *         *          *           *          *           *
*   X    *0.9345E-02 *  1.0000  *  1.0000   * 0.9345E-02 *   107     *
*        *         *          *           *          *           *
*   Y    *0.5000E-02 * 0.5343   *  0.7490   *0.7000E-02 *   107     *
*        *         *          *           *          *           *
*  .....  *         *          *           *          *           *
*        *         *          *           *          *           *
*        *         *          *           *          *           *
*************************************************************************
 ... END OF SIMULATION ...
```

Figure B.3: Simulation results of the example presented in the two previous figures.

Furthermore, in QNAP2, confidence intervals may be produced with a confidence level of 95% when simulation models are evaluated. The confidence intervals produced are themselves estimates because the computation of exact confidence intervals would require an a-priori knowledge of the estimated characteristics.

QNAP2 includes three methods for confidence interval estimation:

- The **replication method** which consists in generating several simple path of the model studied, so that these sample path are statistically independent and identical [SAUER]. This is achieved by reseting the original initial state of the model at the beginning of each replication, and by using a different random number stream for each replication.
- The **regeneration method** which consists to split the the simulation run into sev-

eral successive intervals, so that the model behavior in these intervals be statistically equivalent and independent.

- The **spectral method** which applies to correlated samples provided that the identical distribution property is satisfied.

Appendix C

Method used for Validating Simulation Results

In the previous appendix, it has been mentioned that the QNAP2 package can provide confidence interval level of 95%. However, due to undeterminated reasons, it has been not possible to use the tools offered by the QNAP2 package when an evaluation of the system comprising IEEE 802.6 DQDB MAN has been undertaken.

Hence, another method has been used in order to ensure the validity of the obtained simulation results. This method is to periodically measure the results which have been under interest (throughput, mean end-to-end delay, access delay). Most of the time, this measure has been the mean value calculated from the beginning of the simulation. Then, successive measures have been compared. If three successive measures were taken with only 1% of error between them, then the simulator was stopped. This is represented in Fig. C.1. At the beginning of the simulation, the results differ. However, after a given time, the results tend to converge. The time to get the interval of 1% of error will depend of the parameters used in the simulation. Moreover, the interval between each measure should be long enough in order to ensure that the results will be got for a system in equilibrium [SAUER].

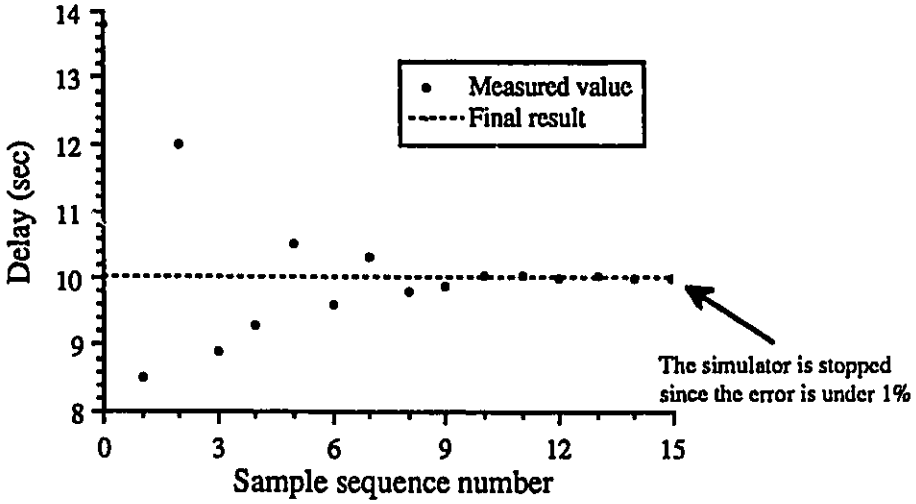


Figure C.1: Converge of a simulation result.

Appendix D

Program listings

D.1 Simulation Program of DQDB

```
$RUN QNAPBIG
$DECK
/CONTROL/ OPTION = NSOURCE;
          CLASS = ALL QUEUE;


&*************************************************************************:
&****IEEE 802.6 DQDB MAN*************************************************:
&*************************************************************************:


&*************************************************************************:
& THIS PROGRAM SIMULATES THE DQDB "DISTRIBUTED QUEUE" DEFINED FOR
& CONTROLLING THE ACCESS TO THE QA SLOTS
&*************************************************************************:

&*************************************************************************:
& Declaration of variables
&*************************************************************************:
/DECLARE/
INTEGER           au = 25,                    & number of access units
                  p = au+1,                   & number of slot places
                  q = au-1,                   & number of inter-stations
                  b = 2,                      & number of buses
                  nb_fr = 20,                 & number of frame used
&*************************************************************************:
                  nb_na = 1,                  & nb of NA slots/frame
                  nb_qa = 4,                  & nb of QA slots/frame
&*************************************************************************:
                  nb_sl = nb_na+nb_qa,        & number of slots/frame
                  i,j,                        & 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=552,                 & slot length (bits)
                  standby(b,au);              & standby mode
REAL              mdelay(ind),                & mean access delay
&*************************************************************************:
                  offer=0.8,                  & network load
&*************************************************************************:
                  t_cus,                      & total of packets sent
                  dummy,                      & dummy variable
                  tdelay,                     & total delay
&*************************************************************************:
                  bus_len=1,                  & 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
CLASS             slot,                      & QA slots
                  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,end_node,             & 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,nb_sl);           & QA slot in a frame


&*****************************************************************************
& 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;
          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:=1;      & 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;
          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;
              IF standby(icb,in)=1     & Is the AQ in standby mode ?
              THEN
                BEGIN
                 standby(icb,in):=0;   & AQ is moved to the countdown state
                 SET(send(icb,in));
                 RESET(send(icb,in));
                END;
             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 (or standby) state
                 THEN
                   BEGIN
                    IF cd_ct(ib,in)=0  & Is the CD counter equal to zero ?
                    THEN
                      BEGIN
                       busy:=1;        & The QA slot is set BUSY
                       cd(ib,in):=0;   & The AQ is moved to the CD state
                       SET(send(ib,in));
                       RESET(send(ib,in));
                      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
            ELSE
              BEGIN
               IF standby(ib,in)=1   & Is the AQ is standby mode ?
               THEN
                 BEGIN
                  standby(ib,in):=0; & The AQ is moved to the CD state
                  SET(send(ib,in));
                  RESET(send(ib,in));
                 END;
              END;
          END;
         IF inext > 0
         THEN
           BEGIN
            IF inext = p
            THEN TRANSIT(end_node)   & 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;
          $switch (a_switch)

&*************************************************************************
& Stations sending on bus B.
&*************************************************************************
/STATION/ NAME = d_switch;
          $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.
&*************************************************************************
/STATION/ NAME = end_node;
          SERVICE = BEGIN
                     IF type=0
                     THEN busy:=0;
                     req:=0;
                     bus:=2;
                     next:=au;
                     CST(latency);
                     TRANSIT(d_switch(au));
                    END;


&*************************************************************************
& 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
                 IF req_ct(ib,in)=0          & Is the REQ counter equal to zero ?
                 THEN
                   BEGIN
                    standby(ib,in):=1;       & AQ moved to standby mode
                    cd(ib,in):=1;
                    cd_ct(ib,in):=0;         & Countdown counter set to zero
                    WAIT(send(ib,in));       & Wait the next QA slot
                    IF standby(ib,in)=1      & Is the next QA slot empty ?
                    THEN
                      BEGIN
                       CST(sl_time);         & Segment transmission
                       TRANSIT(OUT);
                      END;
                   END;
                 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;
$END

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

&*************************************************************************
& Access queue for sending on bus B.
&*************************************************************************
/STATION/ NAME = d_node;
```

```
          $node (d_que)


&*************************************************************************
& Source station
&*************************************************************************
& This station generates asynchronous segments with a time exponentially
& distributed between each segment generation.
&*************************************************************************
/STATION/ NAME = station;
          TYPE = SOURCE;
          SERVICE = EXP(arrival);
          TRANSIT = a_node(in),a_que,au-in,d_node(in),d_que,in-1;


$MACRO print_out
 BEGIN
  PRINT("TIME = ",TIME);
  PRINT("  ");
  PRINT("Access unit service (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 response delay (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:=(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;
  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("   ");
  END;
 $END

 /CONTROL/ TMAX = 15.0;
           PERIOD = 1.0;      & Sample period
           TEST = BEGIN       & Stop the simulation if the results converge
                   OUTPUT;
                   $print_out
                   IF ((ABS(mdelay(ind)-mdelay(ind-1))/mdelay(ind) < 1E-2) AND
                       (ABS(mdelay(ind)-mdelay(ind-2))/mdelay(ind) < 1E-2) AND
                       (ABS(mdelay(ind)-mdelay(ind-3))/mdelay(ind) < 1E-2))
                   THEN STOP;
                  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("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
             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;
          SIMUL;
         END;
 /END/
```

D.2 Simulation Program of DQDB Supporting Image Browsing Activities

```
$RUN QNAPBIG
$DECK
/CONTROL/ OPTION = NSOURCE;
          NMAX = 50;
          CLASS = ALL QUEUE;


&*************************************************************************
& DQDB SUPPORTING RADIOLOGICAL IMAGE BROWSING APPLICATIONS
&*************************************************************************


&*************************************************************************
& THIS PROGRAM SIMULATES DQDB USED AS A TRANSPORT NETWORK FOR TRANSFERRING
& RADIOLOGICAL IMAGES BROWSED BY MEDICAL WORKSTATIONS LOGICALLY LINKED TO
& REMOTE DATABASE SERVERS.
&*************************************************************************


&*************************************************************************
& THE CASE OF THE BROWSING INCLUDING THE USE OF THE COMPRESSION TECHNIQUE
& CALLED THE REDUCED-DIFFERENCE PYRAMID IS PRESENTED IN THIS PROGRAM. THUS
& WORKSTATION USERS CAN BROWSE REDUCED-RESOLUTION IMAGES WITH THE POSSIBILITY
& TO INCREASE THE IMAGE RESOLUTION.
&*************************************************************************


&*************************************************************************
& DECLARATION OF VARIABLES
&*************************************************************************
/DECLARE/
INTEGER          nb_ds = 8,                  & number of database server (AU)
                 au=nb_ds,                   & number of access units on DQDB
                 b = 2,                      & number of buses
                 p = au+1,                   & number of slot places
                 q = au-1,                   & number of inter-stations
                 nb_fr = 20,                 & number of frame used
&*************************************************************************
                 nb_qa = 6,                  & nb of QA slots/frame
&*************************************************************************
                 nb_na = 1,                  & nb of NA slots/frame
                 nb_sl = nb_na+nb_qa,        & number of slots/frame
                 i,j,                        & counter variables
                 ind = 4,                    & Nb of values within 1 % of error
                 ty_ima = 3,                 & number of pyramid levels
                 modulo = 16,                & send sequence number modulo
                 cd(b,au),                   & initial of counters (REQ,CD)
                 cd_ct(b,au),                & countdown counter
                 req_ct(b,au),               & REQ counter
                 rqm(b,au),                  & to send REQ bit on contra bus
                 sl_len=552,                 & slot length (bits)
                 standby(b,au),              & standby mode
                 nod_buf = 8,                & buffer size in AU
                 image(ty_ima),              & number of segments/level
                 ima_seg=19469;              & DQDB slots/image
REAL             mdelay(ty_ima,ind),         & mean image display delay
```

```
                    num(ty_ima,ind),              & total of images sent
                    nb_ima(ty_ima,nb_ds),         & nb of images sent by each DS
                    dummy(ty_ima),                & dummy variable
                    tdelay(ty_ima,nb_ds),         & total delay
                    prob(ty_ima),                 & prob next pyramid level request
                    t_begin = 200E-3,             & initial thinking time
                    think = 2.0,                  & thinking time
                    t_seg = 80E-6,                & segmentation time
                    bus_len=25,                   & bus length in km
                    fr_time=125E-6,               & frame period time in second
                    latency=180E-9,               & station latency in seconds
                    sl_gen=fr_time/nb_sl,         & slot generation period
                    p_delay=5E-6,                 & propagation delay per km
                    fr_len=nb_sl*sl_len,          & frame length (bits)
                    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
FLAG                last(nb_ds),                  & last image segments has been sent
                    grant(nb_ds),                 & node releases one partition
                    send(b,au);                   & to control sending of REQ bit
CLASS INTEGER       icl;                          & class number
CLASS               norm,                         & ordinary image segments
                    lastp,                        & last image segment
                    slot,                         & QA slots
                    segm;                         & packets sent by DS
QUEUE REAL          delay;                        & delay for image
QUEUE INTEGER       ib,                           & bus identifibation
                    icb,                          & contra bus
                    first,                        & first image sending
                    in,                           & station number
                    inext,                        & next station on the bus
                    step,                         & pyramid level processed
                    x,y;                          & counter variable
QUEUE               head,end_node,                & head and end point
                    a_switch(au),b_switch(au),    & ascending and descending switches
                    link,                         & link between stations
                    a_node(au),b_node(au),        & node which queues arrival packets
                    station(au);                  & station generating asyn. segments
CUSTOMER INTEGER    bus,                          & destination bus
                    busy,                         & busy bit
                    next,                         & next station on the bus
                    req,                          & req bit
                    type,                         & type of slot
                    dest;                         & destination station
REF CUSTOMER        SEG(nb_ds,modulo),            & image segment
                    SL(nb_fr,nb_sl);              & QA slot in a frame


&*****************************************************************************
& The stations simulating the "head", "a_switch", "b_switch", "end_node", and
& "link" stations are the same than the one defined in appendix D.I.
&*****************************************************************************


&*****************************************************************************
& QA slot access queue
&*****************************************************************************
```

```
& This station, is similarly defined than the one in appendix D.1, but it is
& slightly modified in order to limit the number of image segments which can
& be queued to it.
&*************************************************************************
$MACRO node
  BEGIN
   IF req_ct(ib,in)=0
   THEN
     BEGIN
      standby(ib,in):=1;
      cd(ib,in):=1;
      cd_ct(ib,in):=0;
      WAIT(send(ib,in));
      IF standby(ib,in)=1
      THEN CST(sl_time)
      ELSE
        BEGIN
         rqm(ib,in):=rqm(ib,in)+1;
         cd(ib,in):=1;
         cd_ct(ib,in):=req_ct(ib,in);
         req_ct(ib,in):=0;
         WAIT(send(ib,in));
         CST(sl_time);
        END;
     END
   ELSE
     BEGIN
      rqm(ib,in):=rqm(ib,in)+1;
      cd(ib,in):=1;
      cd_ct(ib,in):=req_ct(ib,in);
      req_ct(ib,in):=0;
      WAIT(send(ib,in));
      CST(sl_time);
     END;
  END;
$END

&*************************************************************************
& Access queue for sending on bus A.
&*************************************************************************
/STATION/ NAME = a_node(1 STEP 1 UNTIL nb_ds/2);
SERVICE(norm) = BEGIN
                 $node
                 SET(grant(in)); & Release one space in its buffer
                END;
SERVICE(lastp)= BEGIN
                 $node
                 SET(last(in));  & Indicate that the last segment has been sent
                END;
TRANSIT=OUT;

&*************************************************************************
& Access queue for sending on bus B.
&*************************************************************************
/STATION/ NAME = b_node((nb_ds/2+1) STEP 1 UNTIL nb_ds);
SERVICE(norm) = BEGIN
                 $node
                 SET(grant(in));
                END;
SERVICE(lastp)= BEGIN
```

```
                $node
                SET(last(in));
              END;
TRANSIT=OUT;



&*****************************************************************************
& Database server
&*****************************************************************************
& This station simulates the database which contains images browsed by medical
& workstations.
&*****************************************************************************
$MACRO dataserver(xnode)
SERVICE = BEGIN
           IF first=1  Is it the first time an image is generated ?
           THEN
             BEGIN
              EXP(t_begin);
              first:=0;
             END;
           delay:=TIME;  & Begin to measure the mean image display delay
           FOR x:=1 STEP 1 UNTIL image(step)-1 DO    & Image segmentation
             BEGIN
              CST(t_seg);                            & Creation of one segment
              y:=y+1-INTREAL(y/modulo)*modulo;
              SEG(in,y):=NEW(CUSTOMER);
              IF xnode(in).NB > nod_buf              & Is the node buffer full ?
              THEN
                BEGIN
                 WAIT(grant(in));     & Wait for the release of a partition
                 RESET(grant(in));
                END;
              TRANSIT(SEG(in,y),xnode(in),norm);     & Segment sent to the AQ
             END;
           CST(t_seg);                               & Creation of the last segm
           y:=y+1-INTREAL(y/modulo)*modulo;
           SEG(in,y):=NEW(CUSTOMER);
           TRANSIT(SEG(in,y),xnode(in),lastp);       & Sending of the last segm
           WAIT(last(in));                  & Wait the last image segment sending
           RESET(last(in));
           tdelay(step,in):=tdelay(step,in)+TIME-delay;
                    & The delay for the image display is calculated
           nb_ima(step,in):=nb_ima(step,in)+1;
           IF step < ty_ima      & Did the DS just send a reduce-resolution image
           THEN
             BEGIN
              IF DRAW(prob(step)) & Does the DS increase the image resolution ?
              THEN step:=step+1
              ELSE step:=1;
             END
           ELSE step:=1;
           EXP(think);                & Simulates the workstation user thinking time
           TRANSIT(ds(in));
          END;
INIT(segm) = 1;
$END


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

```
& Database server sending on bus A
&*************************************************************************
/STATION/ NAME = ds(1 STEP 1 UNTIL nb_ds/2);
          $ fileserver(a_node)

&*************************************************************************
& Database server sending on bus B
&*************************************************************************
/STATION/ NAME = ds((nb_ds/2+1) STEP 1 UNTIL nb_ds);
          $ fileserver(b_node)



&*************************************************************************
& Measurement of the mean image display for each image resolution observed
& by the workstation users.
&*************************************************************************
$MACRO print_out
 BEGIN
  PRINT("   ");
  PRINT("   ");
  PRINT("TIME =",TIME);
  PRINT("Number of QA slots/frame :",nb_qa);
  FOR j:=1 STEP 1 UNTIL ty_ima DO          & Same procedures for each image
    BEGIN                                  & resolution requested
     FOR i:=2 STEP 1 UNTIL ind DO
     num(j,i-1):=num(j,i);
     dummy(j):=0;
     num(j,ind):=0;
     FOR i:=1 STEP 1 UNTIL nb_ds DO
       BEGIN
        num(j,ind):=num(j,ind)+nb_ima(j,i);
        dummy(j):=dummy(j)+tdelay(j,i);
       END;
     IF num(j,ind)>num(j,ind-1)
     THEN
       BEGIN
        FOR i:=2 STEP 1 UNTIL ind DO
        mdelay(j,i-1):=mdelay(j,i);
        mdelay(j,ind):=dummy(j)/num(j,ind); & Mean image display delay obtained
       END;
     PRINT("Delay level",j,"  (sec)    :  ",mdelay(j,ind));
    END;
 END;
$END


/CONTROL/
TMAX = 500.0;   & Maximum time of the simulation
PERIOD = 5;     & PERIOD forces the measurement of the mean image display delay
                & for each pyramid level requested (at each 5 seconds)
TEST=BEGIN
      IF TIME > 60
      THEN
        BEGIN         & Verify if results converge
         $print_out
         IF ((ABS(mdelay(1,ind)-mdelay(1,ind-1))/mdelay(1,ind) < 1E-2) AND
             (ABS(mdelay(1,ind)-mdelay(1,ind-2))/mdelay(1,ind) < 1E-2) AND
             (ABS(mdelay(1,ind)-mdelay(1,ind-3))/mdelay(1,ind) < 1E-2) AND
             (ABS(mdelay(2,ind)-mdelay(2,ind-1))/mdelay(2,ind) < 1E-2) AND
```

```
            (ABS(mdelay(2,ind)-mdelay(2,ind-2))/mdelay(2,ind) < 1E-2) AND
            (ABS(mdelay(2,ind)-mdelay(2,ind-3))/mdelay(2,ind) < 1E-2) AND
            (ABS(mdelay(3,ind)-mdelay(3,ind-1))/mdelay(3,ind) < 1E-2) AND
            (ABS(mdelay(3,ind)-mdelay(3,ind-2))/mdelay(3,ind) < 1E-2) AND
            (ABS(mdelay(3,ind)-mdelay(3,ind-3))/mdelay(3,ind) < 1E-2))
         THEN STOP;   & Stop the simulation if results converge
        END;
    END;
ENTRY = BEGIN
         PRINT("IMAGE BROWSING ONTO DQDB");
         PRINT("----------------------");
         PRINT(" ");
         PRINT(" ");
         PRINT("USING PROGRESSIVE IMAGE TRANSMISSION");
         PRINT("-----------------------------------");
         PRINT(" ");
         PRINT(" ");
         PRINT("INPUT PARAMETER ----> X-RAY IMAGES");
         PRINT("2048 x 2560 pixels image");
         PRINT(" ");
         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 access units    : ',F3.0)",au);
         PRINTF("(1H,'Ring length (km)          :  ',F4.1)",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);
        END;
EXIT = $print_out

/EXEC/ BEGIN
        FOR i := 1 STEP 1 UNTIL ty_ima DO
          BEGIN
           FOR j := 1 STEP 1 UNTIL ind DO
           mdelay(i,j):=1E-21;
          END;
        FOR i := 1 STEP 1 UNTIL au DO
          BEGIN
           a_switch(i).in:=i;
           b_switch(i).in:=i;
           a_switch(i).ib:=1;
           b_switch(i).ib:=2;
           a_switch(i).icb:=2;
           b_switch(i).icb:=1;
           a_switch(i).inext:=i+1;
           b_switch(i).inext:=i-1;
           a_node(i).in:=i;
           b_node(i).in:=i;
           a_node(i).ib:=1;
           b_node(i).ib:=2;
           ds(i).in:=i;
           ds(i).first:=1;
           ds(i).step:=1;
          END;

        prob(1):=0.2;     & Prob. to ask for data detailing the 2nd R-R version
```

```
        prob(2):=0.8;     & Prob. to ask for data detailing the F-R image
        image(1):=2643;   & nb of DM PDUs representing the 1st reduced-resolution
        image(2):=7928;   & nb of DM PDUs representing the 2nd reduced-resolution
        image(3):=31711; & nb of DM PDUs representing the original

        SIMUL;
       END;
/END/
```

D.3 Simulation Program of DQDB used as a Backbone Network Interconnecting Two Token Rings

D.3.1 Data Packet Transfer Application

```
$RUN QNAPBIG
$DECK
/CONTROL/ OPTION = NSOURCE;


&*************************************************************************
& DQDB INTERCONNECTING LOW-SPEED LANS
&*************************************************************************



&*************************************************************************
& DATA PACKET TRANSFER APPLICATION
&*************************************************************************



&*************************************************************************
& In this program we simulates communications between nodes situated on two
& 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.
&*************************************************************************

&*************************************************************************
& The use of higher priority mode allocated to the gateways in token rings is
& considered.
&*************************************************************************

/CONTROL/ NMAX = 60;
          CLASS = ALL QUEUE;


&*************************************************************************
& Declaration of variables
&*************************************************************************
/DECLARE/
INTEGER          r = 2,                    & number of token ring
                 st = 6,                   & number of stations/ring
                 nod = st+1,               & number of nodes/ring
                 b = 2,                    & number of buses
                 au = 2,                   & number of access units on DQDB
                 pl = au+1,                & number of slot places
                 nb_qa = 9,                & number of non iso slots per frame
                 nb_na = 1,                & number of iso slots/frame
                 nb_sl = nb_na+nb_qa,      & number of slots/frame
                 mod = 128,                & modulus size
                 index = mod*2,            & index range
&*************************************************************************
                 window = 8,               & 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= 62,                    & payload segment length
                sl_len=69,                      & slot length (bits)
                nb_seg = 17,                    & 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
                standby(b,au);                  & standby mode
REAL            buffer = 8*i_frame,             & buffer size
&*****************************************************************************
                ar_rate = 100,                  & 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
&*****************************************************************************
                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,                & T.R 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 bit rate (bits/s)
                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
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   ic,icl,icr,                 & indicate classes
                iclop,                      & indicate remote class
                length,                     & packet length in bits
                size;                       & packet length in bytes
CLASS           token(r),                   & token
                dm(st),dm2(st),        & DM PDUs
                i_f1(st),i_f2(st),          & I-frames
                s_f1(st),s_f2(st),          & 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;                        & counter variable
QUEUE REAL      load,tdelay;                & total delay
QUEUE           stat1(st),stat2(st),        & stations
                hlay1(st),hlay2(st),        & high layers
                wind1(st),wind2(st),        & windows
                buff1(st),buff2(st),        & buffers
                llc1(st),llc2(st),          & LLC
                sink1(st),sink2(st),        & sinks
                tim1(st),tim2(st),          & timer
                tok_dq(r),dq_tok(r),        & gateway
                mac1(nod),mac2(nod),        & MAC
                ring(r),                    & token rings
                head,end_node,              & 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
                busy,                       & busy bit
                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,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
&*************************************************************************
$MACRO 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;
$END

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

&*************************************************************************
& Application layer generating data unit eventually sent on token ring # 2
&*************************************************************************
/STATION/ NAME = stat2;
          $stat (hlay2,i_f2)


&*************************************************************************
& High layer
&*************************************************************************
& This station is throttling the generation of data unit generated by the
& corresponding application layer entitylinggenerating data units.
&*************************************************************************
$MACRO 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;
$END

&*************************************************************************
& High layer serving data units eventually sent on token ring # 1
&*************************************************************************
/STATION/ NAME = hlay1;
          $highlayer (wind1,llc1,i_f1)

&*************************************************************************
& High layer serving data units eventually sent on token ring # 2
&*************************************************************************
/STATION/ NAME = hlay2;
          $highlayer (wind2,llc2,i_f2)


&*************************************************************************
& 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.
&*************************************************************************
$MACRO buffer (wind,tank,llc,i_f)
  SERVICE = BEGIN
            WAIT(dump(ir,is));
            IF resent(ir,is)=0    & Has a N(R) been received ?
            THEN                  & When N(R) simply acknowledges I-frame
              BEGIN
               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)
               IF pos(ir,is,sn) < pos(ir,is,ack_fr(ir,is))
               THEN               & Remove I-frames from the window
                 BEGIN
                  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
                  IF sn=resent(ir,is)
                  THEN RESET(dump(ir,is));
                  TRANSIT(llc(is));  & I-frames are sent to the LLC process
                 END;
              END;
          END;
$END

&*************************************************************************
& Buffer serving I-frames eventually sent on token ring #1
&*************************************************************************
```

```
/STATION/ NAME = buff1;
          $buffer (wind1,tank1,llc1,i_f1)

&*************************************************************************
& Buffer serving I-frames eventually sent on token ring #2
&*************************************************************************
/STATION/ NAME = buff2;
          $buffer (wind2,tank2,llc2,i_f2)


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

&*************************************************************************
& LLC window serving I-frames eventually sent on token ring #2
&*************************************************************************
/STATION/ NAME = wind2;
          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.
&*************************************************************************
$MACRO llc (i_loc,i_rem,s_loc,s_rem,llcc,macc,tim,buff,sink)

&*************************************************************************
& Service of the I-frames sent by the station
&*************************************************************************
SERVICE(i_loc)= BEGIN
                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
                 BEGIN
                  CST(t_first);
                  first(ir,is,sn):=1;
                 END
               ELSE CST(t_llc);
               TRANSIT(I(ir,is,ind),macc(is),i_loc(icl));
               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)=BEGIN
               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);
                  TRANSIT(S(ir,is,ind),macc(is),s_loc(icl)); & 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)
                  TRANSIT(S(ir,is,ind),macc(is),s_loc(icl)); & 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)= BEGIN
                 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);
                     TRANSIT(S(ir,is,ind),macc(is),s_loc(icl));
                    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;
$END

&*************************************************************************
& LLC serving I-frames eventually sent on token ring #1
&*************************************************************************
/STATION/ NAME = llc1;
          $llc (i_f1,i_f2,s_f1,s_f2,llc1,mac1,tim1,buff1,sink1)

&*************************************************************************
& LLC serving I-frames eventually sent on token ring #2
&*************************************************************************
/STATION/ NAME = llc2;
          $llc (i_f2,i_f1,s_f2,s_f1,llc2,mac2,tim2,buff2,sink2)


&*************************************************************************
& 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
&*************************************************************************
  SERVICE = BEGIN
             tdelay:=tdelay+TIME-start;
             TRANSIT(OUT);
            END;
$END

&*************************************************************************
& I-frame sink in stations situated on token ring #1
```

```
&*************************************************************************
/STATION/ NAME = sink1;
          $sink

&*************************************************************************
& I-frame sink in stations situated on token ring #2
&*************************************************************************
/STATION/ NAME = sink2;
          $sink


&*************************************************************************
& LLC TIMERS
&*************************************************************************
& This station simulates two timers: the acknowledgement timer, and the P-bit
& timer
&*************************************************************************
$MACRO 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;
$END

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

&*************************************************************************
& LLC timers used by stations situated on token ring #2
&*************************************************************************
/STATION/ NAME = tim2;
          $timer (mac2,tim2,s_f2)


&*************************************************************************
& 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)
SERVICE=BEGIN
         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 TRANSIT(tok_dq(ir))  & If yes, ---> frame transmission
          ELSE TRANSIT(OUT);        & If not, the frame is discarded
         END;
$END

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

&*************************************************************************
& MAC layer serving end stations on token ring #2
&*************************************************************************
/STATION/ NAME = mac2(1 STEP 1 UNTIL st);
          $maclayer(i_f2,s_f2,d_node)


&*************************************************************************
& 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
&*************************************************************************
$MACRO 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(icl));
            END;
$END

&*************************************************************************
& 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)



&*****************************************************************************
& 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) = IF ir=icr THEN 1
                                  ELSE 0;
```

```
$END

&*************************************************************************
& Token ring monitor on ring #1
&*************************************************************************
/STATION/ NAME = ring(1);
          $tok_ring(mac1)

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



&*************************************************************************
& 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) = 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))
                         ELSE TRANSIT(DM(ir,icl,unic,x),d_node(2),dm2(icl));
                        END;
                      CST(t_sgm);
                      IF ir=1
                      THEN TRANSIT(a_node(1),i_f1(icl))
                      ELSE TRANSIT(d_node(2),i_f2(icl));
                     END;
SERVICE(s_f1,s_f2) = BEGIN
                      &************************
                      & S-frame processing
                      &************************
                      CST(t_conv+t_sgm);
                      IF ir=1
                      THEN TRANSIT(a_node(1),s_f1(icl))
                      ELSE TRANSIT(d_node(2),s_f2(icl));
                     END;



&*************************************************************************
& Processor performing the reassembly of token ring frames in the gateways
&*************************************************************************
/STATION/ NAME = dq_tok(1 STEP 1 UNTIL r);
```

```
SERVICE(dm1,dm2) = BEGIN            & BOM, and COM DM PDUs reception
                     CST(t_rec);
                     IF sg=exp_sg(iso,icl,unic) & Is received segme expected?
                     THEN
                       BEGIN
                        counter(ir):=exp_sg(iso,icl,unic); & For further use
                        exp_sg(iso,icl,unic):=exp_sg(iso,icl,unic)+1;
                        IF exp_sg(iso,icl,unic)=nb_seg-1 & all BOM AND COM
                        THEN all(iso,icl,unic):=1;       & have been received
                       END
                     ELSE counter(ir):=0;       & Rejection of DM PDU
                     TRANSIT(OUT);
                    END;
SERVICE(i_f1,i_f2)=BEGIN              & EOM DM PDU reception
                    CST(t_rec);
                    IF all(iso,icl,unic)=1 & Are all previous segments received
                    THEN
                      BEGIN
                       CST(t_conv);    & The token ring is reassemblied
                       counter(ir):=0;
                       IF ir=1
                       THEN TRANSIT(mac1(nod))
                       ELSE TRANSIT(mac2(nod));
                      END
                    ELSE TRANSIT(OUT);
                   END;
SERVICE(s_f1,s_f2)=BEGIN             & Reception of DM PDU containing a S-frame
                    CST(t_sgm+t_conv);
                    IF ir=1
                    THEN TRANSIT(mac1(nod))
                    ELSE TRANSIT(mac2(nod));
                   END;



&*************************************************************************
& The stations simulating the "head", "a_switch", "b_switch", "end_node", and
& "link" stations are the same defined in appendix D.1.
&*************************************************************************



&*************************************************************************
& 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.
&*************************************************************************
$MACRO trans (macc,i_f,s_f)
  full(iclop,icl,sn+sg):=payload*nbmini(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));
  IF full(iclop,icl,sn+sg)<=buffer-sl_len
  THEN TRANSIT(dq_tok(iclop))
  ELSE TRANSIT(OUT);
$END


/STATION/ NAME = link;
TYPE=INFINITE;
```

```
SERVICE(i_f1,s_f1,mini1)=BEGIN
                          CST(t_link);
                          $trans(mac2,i_f1,s_f1)
                         END;
SERVICE(i_f2,s_f2,mini2)=BEGIN
                          CST(t_link);
                          $trans(mac1,i_f2,s_f2)
                         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 (class,server)
SERVICE(class)= BEGIN
                 IF req_ct(ib,in)=0          & Is the REQ counter equal to zero ?
                 THEN
                   BEGIN
                    standby(ib,in):=1;       & AQ moved to standby mode
                    cd(ib,in):=1;
                    cd_ct(ib,in):=0;         & Countdown counter set to zero
                    WAIT(send(ib,in));       & Wait the next QA slot
                    IF standby(ib,in)=1      & Is the next QA slot empty ?
                    THEN
                      BEGIN
                       CST(sl_time);         & Segment transmission
                       TRANSIT(link);
                      END;
                   END;
                 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(link);
                END;
$END

&*************************************************************************
& Access queue for sending on bus a.
&*************************************************************************
/STATION/ NAME = a_node;
          $node

&*************************************************************************
& Access queue for sending on bus B.
&*************************************************************************
/STATION/ NAME = d_node;
          $node


$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(" ");
END;
$END


/CONTROL/ TMAX = 100.0;
          PERIOD=2.0;
          TEST=BEGIN     & Use to stop the simulation if the results converged
                IF TIME > 20
                THEN
                  BEGIN
                   FOR i:=2 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;
                     del(count):=del(count)+sink1(i).tdelay/sink1(i).NBOUT+
                                 sink2(i).tdelay/sink2(i).NBOUT;
                    END;
                  thru(count):=byte*info*thru(count)/TIME;
                  del(count):=del(count)/2/st;
                  $print_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("PRIORITY MODE IN TOKEN RING");
                   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");
                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;
            tok_dq(i).iso:=r-i+1;
            dq_tok(i).iso:=r-i+1;
            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;
            stat1(i).is:=i;
```

```
    stat2(i).is:=i;
    stat1(i).num:=1;
    stat2(i).num:=1;
    hlay1(i).is:=i;
    hlay2(i).is:=i;
    hlay1(i).ir:=1;
    hlay2(i).ir:=2;
    hlay1(i).init:=1;
    hlay2(i).init:=1;
    buff1(i).is:=i;
    buff2(i).is:=i;
    buff1(i).ir:=1;
    buff2(i).ir:=2;
    llc1(i).is:=i;
    llc2(i).is:=i;
    llc1(i).ir:=1;
    llc2(i).ir:=2;
    sink1(i).is:=i;
    sink2(i).is:=i;
    sink1(i).ir:=1;
    sink2(i).ir:=2;
    sink1(i).iso:=2;
    sink2(i).iso:=1;
    tim1(i).is:=i;
    tim2(i).is:=i;
    tim1(i).ir:=1;
    tim2(i).ir:=2;
    i_f1(i).icl:=i;
    i_f2(i).icl:=i;
    i_f1(i).iclop:=2;
    i_f2(i).iclop:=1;
    i_f1(i).size:=i_frame;
    i_f2(i).size:=i_frame;
    i_f1(i).length:=i_frame*byte;
    i_f2(i).length:=i_frame*byte;
    s_f1(i).icl:=i;
    s_f2(i).icl:=i;
    s_f1(i).iclop:=2;
    s_f2(i).iclop:=1;
    s_f1(i).size:=s_frame;
    s_f2(i).size:=s_frame;
    s_f1(i).length:=s_frame*byte;
    s_f2(i).length:=s_frame*byte;
    mini1(i).icl:=i;
    mini2(i).icl:=i;
    mini1(i).iclop:=2;
    mini2(i).iclop:=1;
  END;
FOR i:=1 STEP 1 UNTIL nod DO
  BEGIN
    mac1(i).is:=i;
    mac2(i).is:=i;
    mac1(i).ir:=1;
    mac2(i).ir:=2;
  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/
```

D.3.2 Image Browsing Application

```
$RUN QNAPBIG
$DECK
/CONTROL/ OPTION = NSOURCE;



&*************************************************************************
& DQDB INTERCONNECTING LOW-SPEED LANS
&*************************************************************************



&*************************************************************************
& IMAGE BROWSING APPLICATION
&*************************************************************************


&*************************************************************************
& This program simulates browsing activities between workstations and
& database servers situated in the same or different hospitals. A token ring
& DQDB MAN is used to interconnet each token ring.  Gateways are used to
& interconnect DQDB and token rings, since both types of network do not use
& the same kind of packets (packet length, overhead). The IEEE 802.2 type-2
& logical-link-control(LLC) is employed as an end-to-end protocol between the
& communicating stations.
&*************************************************************************


&*************************************************************************
& In this program, we are mainly interested to evaluate the performance
& for image browsing by a medical workstation in a hospital, linked
& with a database server situated on a remote token ring serving another
& hospital. At the same time, to simulate requests to the database server from
& other devices, we consider background load from other nodes which generate
& image requests to any databse in the interwork system. The emphasis is put
& on the elapsed time measured from the time an image browsing request is
& issued at the measure workstation until the image is displayed on its monitor.
&*************************************************************************



&*************************************************************************
& In this program, reduced-resolution images are browsed. Moreover, the users
& have the choice to halt more time on image whoses resolution will be
& increased as long as the image is observed. A progressive image transmission
& technique based on the use of pyramid data structure is employed.
&*************************************************************************

/CONTROL/ NMAX   = 60;
          CLASS = ALL QUEUE;


&*************************************************************************
& Declaration of variables
&*************************************************************************
/DECLARE/
INTEGER             r = 2,                    & number of token rings
                    bg = 5,                   & background mode
```

```
                ws = bg+1,                  & number of workstations/ring
                ds = 3,                     & number of database servers/ring
                nod = ws+ds+1,              & number of nodes/ring
                stat = ws+ds,               & number of stations/ring
                au = r,                     & number of access units
                pl = au+1,                  & number of slot places
                nb_fr = 16,                 & number of used frames
                nb_niso = 9,                & number of non iso slots per frame
                nb_iso = 1,                 & number of iso slots/frame
                nb_sl = nb_iso+nb_niso,     & number of slots/frame
                byte = 8,                   & 1 byte --> 8 bits
                nb_level = 3,               & nb of pyramid level sent
                window = 8,                 & window size used by by LLC
                modulo = 16,                & image segment identification
                num = modulo*2,             & iden number for background request
                ind = 3,                    & dummy variable
                i,j,                        & looped variables
                c_sg = 17,                  & nb of cells for MAX DQDB MAC PDU
                e_sg = 15,                  & nb of cells for last DQDB MAC PDU
                llc_ctr = 4,                & LLC PDU control bits
                l_info = 896,               & length for last DQDB MAC PDU info
                m_info = 1024,              & length for MAX DQDB MAC PDU info
                l_llc = l_info+llc_ctr,     & length for last DQDB MAC PDU info
                m_llc = m_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
                tthru(ind),                 & for storing WS thruput
                s_frame = tk_hdr+tk_tr+llc_ctr,  & s-frame length in TR
                ss_frame = dq_hdr+llc_ctr+dq_tr, & s-frame in DQDB MAC PDU
                seh_hdr = 4,                  & segment header
                payload= 62,                  & payload segment length
                l_pck = tk_hdr+l_llc+tk_tr,   & length of last TR frame
                m_pck = tk_hdr+m_llc+tk_tr,   & length for MAX TR frame
                l_dq_p = dq_hdr+l_llc+dq_tr,  & length of last DQ frame
                m_dq_p = dq_hdr+m_llc+dq_tr,  & length for MAX DQ frame
                counter,                    & nb of images sent by the database
                max_pck = 640,              & nb of images pck for back request
                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
                nb_seg(3),                  & nb of image pck for pyr levels
                queue,                      & timer request status
                standby(b,au);              & standby mode
                ask(r,nod);                 & MAC having packet queues
REAL            tok_spd = 16E6,             & token ring bit rate
&*************************************************************************
                bg_arr1 = 160,              & local background traffic rate
                bg_arr2 = bg_arr1,          & remote background traffic rate
&*************************************************************************
                choice(nb_level),           & probability for wishing next level
                dusty = 1E-6,               & dusty variable
                initial = 0.2,              & initial thinking time
                inter = 0.20,               & percentage of local traffic
                think = 2.0,                & thinking time
                t_first = 250E-6,           & proc. time to send I-frame
                t_rr = 300E-6,              & receive I-frame and send RR-frame
                t_slide = 100E-6,           & time to slide the window
```

```
                t_req = 100E-6,                 & time to process image request
                t_conv = 150E-6,                & proc. time to convert packets
                t_ds= 1.25E-6,                  & processing time/byte in DB server
                t_timer = 5.0,                  & database timer period
                t_p_last = t_ds*l_info,         & processing of max segm
                t_p_max = t_ds*m_info,          & processing of last segm
                t_rec = 20E-6,                  & processing to receive DQDB slots
                t_sgm = 31E-6,                  & time to format DQDB slots
                thru(nb_level),                 & nb of received image levels
                tdelay(nb_level),               & total delay
                mdelay(ind),    & mean delay for displaying the original image
                t_thru,                         & total throughput
                bus_len=1,                      & bus length in KM
                fr_time=125E-6,                 & frame period time in second
                latency=600E-9,                 & access unit latency
                p_delay=5.085E-6,               & propagation delay per km
                sl_len=552,                     & slot length (bits)
                sl_siz=69,                      & slot length (bytes)
                st_to_st=bus_len/(au+1),        & distance between two stations
                n_to_n=p_delay*st_to_st,        & node to node propagation delay
                fr_len=nb_sl*sl_len,            & frame length (bits)
                dq_spd=fr_len/fr_time,          & DQDB bit rate (bits/s)
                sl_time=sl_len/dq_spd,          & slot length in seconds
                t_link=sl_time/2+sl_time+n_to_n+latency, &time in link
                dq_ser=fr_time/nb_niso;         & perfect scheduler approx
FLAG            fl(r,nod),                      & token possession
                poll(r),                        & passing of the token
                windfl(r,ds);                   & access to the WS MAC
CLASS INTEGER   icl,                            & classes
                icr,                            & ring classes
                length,                         & packet length
                nbseg;                          & number of slots/TR frame
CLASS           dm1(ds),dm2(ds),                & DQDB cells
                pm1(ds),pm2(ds),                & to transfer image segments
                pl1(ds),pl2(ds),                & to transfer image segments
                ack1(ws),ack2(ws),              & acknowledgement of image segments
                r1(ws),r2(ws),                  & to transfer WS requests
                rt,                             & special request from timer
                slot,                           & DQDB slots
                token(r),                       & token
                sh,que;                         & empty use
QUEUE INTEGER   active,                         & server status
                ir,is,iso,                      & ring,station number
                ind,                            & index
                init,                           & initialization
                mac,                              position of token
                first,                          & first request
                num,                            & send sequence number
                ib,                             & bus identifibation
                icb,                            & contra bus
                in,                             & station number
                inext,                          & next station on the bus
                mac,                            & position of the token
                inext,                          & next station number on the path
                x,y;                            & looped variables
QUEUE REAL      bg_arr;                         & backload arrival rate
QUEUE           ws1(ws),ws2(ws),                & workstations
                ds1(ds),ds2(ds),                & database servers
                timer,                          & database timer
                llc1(stat),llc2(stat),          & LLC in database servers
```

```
                tok_dq(r),dq_tok(r),        & segmentation/reassembly processors
                mac1(nod),mac2(nod),        & MAC
                ring(r),                    & token rings
                head,end_node,              & head and end point on DQDB
                a_switch(au),d_switch(au),  & ascending and descending switches
                link,                       & link transferring DQDB slots
                a_node(au),b_node(au),      & DQDB nodes
                shut;                       & printing facilities
CUSTOMER REAL   start;                      & measure the image display delay
CUSTOMER INTEGER level,                     & pyramid level
                d_ring,                     & destination ring
                d_llc,                      & destination LLC
                next,                       & next station on DQDB
                req,                        & req bit
                bus,                        & destination bus
                busy,                       & busy bit
                type;                       & type of slot
                sn;                         & sequence number
REF CUSTOMER    TOK(r,nod),                 & token
                REQ,                        & workstation request
                REQQ(r,bg,num),             & backgroud node request
                SEG(r,ds,modulo,c_sg),      & 1 image --> x segmentc_sg)
                TIM,                        & Request generated by the TIMER
                SL(nb_fr,nb_sl);            & QA slot in a frame

&*****************************************************************************
& Workstation whose use is evaluated (situated on token ring #1)
&*****************************************************************************
/STATION/ NAME = ws1(ws);
SERVICE = BEGIN
          active:=1;
          IF first=1
          THEN          & This is the first an image request is generated
            BEGIN
             first:=0;
             REQ:=NEW(CUSTOMER);  & Generation of the image request
             REQ.d_ring:=2;
             REQ.d_llc:=ws+1;
             REQ.level:=1;
             EXP(initial);
             REQ.start:=TIME;   & Begin to measure the mean image display delay
             TRANSIT(REQ,llc1(is),r1(is));
             TRANSIT(OUT);
            END
          ELSE
            BEGIN
             tdelay(level):=tdelay(level)+TIME-start; & End to measure the ...
             thru(level):=thru(level)+1; & Measure number of images received
             IF DRAW(choice(level))
             THEN               & Decide to wait for the next higher resolution
               active:=0;
               TRANSIT(OUT);
              END
             ELSE               & Request the next image
               BEGIN
                level:=1;       & Will receive the first reduce resolution
                EXP(think);
                active:=0;
                start:=TIME;    & Begin to measure the time for the calcul
                                & of the mean image display delay.
```

```
                TRANSIT(llc1(is),r1(is));
               END;
            END;
         END;
INIT(r1) = IF is=icl THEN 1
                     ELSE 0;

&*************************************************************************
& Workstations generating background load requests
&*************************************************************************
$MACRO ws(llc,r)
$ws(llc,r)
TYPE=SOURCE;
SERVICE = BEGIN
           EXP(1/bg_arr);
           x:=x+1-INTREAL(x/num)*num;
           REQQ(ir,is,x):=NEW(CUSTOMER);  & Generation of the image request
           IF DRAW(inter)                 & Image request sent on any token ring
           THEN REQQ(ir,is,x).d_ring:=y
           ELSE REQQ(ir,is,x).d_ring:=ir;
           REQQ(ir,is,x).d_llc:=ws+RINT(1,ds);& Image request--> database server
           TRANSIT(REQQ(ir,is,x),llc(is),r(is));
           TRANSIT(OUT);
          END;
$END

&*************************************************************************
& Workstations generating background load requests, situated on token ring #1
&*************************************************************************
$MACRO ws(llc,r)
/STATION/ NAME = ws1(1 STEP 1 UNTIL bg);
                 $ws(llc1,r1)

&*************************************************************************
& Workstations generating background load requests, situated on token ring #2
&*************************************************************************
/STATION/ NAME = ws2(1 STEP 1 UNTIL bg);
                 $ws(llc2,r2)


&*************************************************************************
& Database server
&*************************************************************************
& This server, upon request arrival, is assumed to retrieve the requested images
& and to segment it in order to transmit it on the system. When the request is
& coming from the measured workstation, and a reduce resolution is transferred,
& the database server has to start a timer at end of the service.
&*************************************************************************
$MACRO ds(llc,pm,pl)
SERVICE = BEGIN
           SET(windfl(ir,is));
           IF icl=ws
           THEN
             BEGIN
              counter:=nb_seg(level);
              IF sn=m_info
              THEN
                BEGIN
                 IF ws1(ws).active=1
                 THEN
```

```
                BEGIN
                 ws1(ws).active:=0;
                 MOVE(ws1(ws),OUT);
                END;
              END;
            END
           ELSE counter:=max_pck;
           FOR x:=1 STEP 1 UNTIL counter-1 DO  & counter = nb of segments
            BEGIN                              & representing an image (or R.R)
             CST(t_p_max);                     & Time for formatting I-frame
             y:=y+1-INTREAL(y/modulo)*modulo;
             SEG(ir,is,y,c_sg):=NEW(CUSTOMER); & generation of an image segment
             SEG(ir,is,y,c_sg).sn:=y;
             SEG(ir,is,y,c_sg).d_ring:=icr;    & icr,and icl represent the
             SEG(ir,is,y,c_sg).d_llc:=icl;     & location of the request sender
             WAIT(windfl(ir,is));              & Wait if the LLC is saturated
             TRANSIT(SEG(ir,is,y,c_sg),llc(inext),pm(is));
            END;
           CST(t_p_last);
           y:=y+1-INTREAL(y/modulo)*modulo;
           SEG(ir,is,y,c_sg):=NEW(CUSTOMER);    & Gen. of the last image segment
           SEG(ir,is,y,c_sg).sn:=y;
           SEG(ir,is,y,c_sg).d_ring:=icr;
           SEG(ir,is,y,c_sg).d_llc:=icl;
           SEG(ir,is,y,c_sg).level:=level;      & pyramid level sent
           SEG(ir,is,y,c_sg).start:=start;
           WAIT(windfl(ir,is));                 & Wait if the LLC is saturated
           TRANSIT(SEG(ir,is,y,c_sg),llc(inext),pl(is));
           IF ((icl=ws) AND (level < nb_level))
           THEN     & The image has been sent to the  measured workstation
            BEGIN   & (This has also been a reduced-resolution
             TIM:=NEW(CUSTOMER);
             TIM.d_ring:=2;
             TIM.d_llc:=ws+1;
             TIM.level:=level+1;
             TIM.sn:=m_info;
             TRANSIT(TIM,timer,r2(ws)); & The database timer will be started
            END;
           IF sn=m_info THEN queue:=0;& Indicate that a request for sending an
           TRANSIT(OUT);              & image to the measured WS has been served
          END;
$END

&*************************************************************************
& Database server on token ring #1
&*************************************************************************
/STATION/ NAME = ds1;
                $ds(llc1,pm1,pl1)

&*************************************************************************
& Database server on token ring #2
&*************************************************************************
/STATION/ NAME = ds2;
                $ds(llc2,pm2,pl2)



&*************************************************************************
& Database timer
&*************************************************************************
```

```
& When the database has completed the transmission of the reduced-resolution,
& it initializes the timer. If there is no request receipt before the timer
& expires, the timer generates a request to the database for sending to the
& measured workstation the next higher resolution of the image.
&*****************************************************************************
/STATION/ NAME = timer;
SERVICE = BEGIN
            active:=1;
            CST(t_timer);
            start:=TIME;
            & At this point, the timer is considered having expired
            active:=0;
            queue:=1;      & Indicate that the timer has queued an image request
            TRANSIT(ds2(1));  & Send a request to the local database
          END;

&*****************************************************************************
& LLC comprised in the medical workstation
&*****************************************************************************
& The Go back N protocol is used with window size of eight. This LLC is mainly
& used for performing acknowledgement of I-frames sent by the database server.
&*****************************************************************************
$MACRO llc_ws(macc,wss,ack,r)
SERVICE(r)=BEGIN
             CST(t_first);
             TRANSIT(macc(is));
           END;
SERVICE(pm1,pm2)= BEGIN   & reception of I-frame (having maximum allowed length)
                   CST(t_rr);
                   d_ring:=icr;
                   d_llc:=ws+icl;
                   TRANSIT(macc(is),ack(is));  & Acknowledge the I-frame
                  END;
SERVICE(pl1,pl2)=BEGIN   & reception of the last I-frame comprising the image
                  CST(t_rr);
                  IF ((ir = 1) AND (is = ws)) & Addressed to the measured WS
                  THEN
                    BEGIN
                     REQ:=NEW(CUSTOMER);
                     REQ.d_ring:=2;
                     REQ.d_llc:=ws+1;
                     REQ.level:=level;
                     REQ.start:=start;              & Received image will
                     TRANSIT(REQ,wss(is),r(is));  & be displayed at the WS
                    END;
                  d_ring:=icr;
                  d_llc:=ws+icl;
                  TRANSIT(macc(is),ack(is));  & Acknowledge the I-frame
                 END;
$END

&*****************************************************************************
& LLC comprised in the medical workstation situated on token ring #1
&*****************************************************************************
/STATION/ NAME = llc1(1 STEP 1 UNTIL ws);
                 $llc_ws(mac1,ws1,ack1,r1)

&*****************************************************************************
& LLC comprised in the medical workstation situated on token ring #2
&*****************************************************************************
```

```
/STATION/ NAME = llc2(1 STEP 1 UNTIL ws);
                 $llc_ws(mac2,ws2,ack2,r2)


&*************************************************************************
& LLC comprised in the medical database server
&*************************************************************************
& The Go back N protocol is used with window size of eight. This LLC is mainly
& used for performing sending of I-frames to medical workstation
&*************************************************************************
$MACRO llc_ds(macc,dbserver)
SERVICE=BEGIN
          CST(t_first);
          x:=x+1;
          IF x >= window                    & Window flow control.
          THEN RESET(windfl(ir,inext));     & "Wind" used to block the generation
          TRANSIT(macc(is));                &  of image segments by the database
        END;
SERVICE(ack1,ack2)=BEGIN
                    CST(t_slide);
                    x:=x-1;                 & The window is slided
                    IF x=window-1
                    THEN SET(windfl(ir,inext)); & One window room is released
                    TRANSIT(OUT);
                   END;
SERVICE(r1,r2)=BEGIN
                CST(t_req);
                IF icI=ws     & Apply to requests generated by the measured WS
                THEN
                  BEGIN
                   IF timer.active=1
                   THEN           & The timer is stopped if it is running
                     BEGIN        &
                      timer.active:=0;
                      MOVE(timer,OUT); & Timer stopped
                     END
                   ELSE
                     BEGIN
                      IF queue=1          & If there is already a request queued
                      THEN TRANSIT(OUT); & for the measured WS (following timer
                     END;                 & expiration)
                  END;
                TRANSIT(dbserver(inext));
               END;
$END

&*************************************************************************
& LLC comprised in the medical database server on token ring #1
&*************************************************************************
/STATION/ NAME = llc1(ws+1 STEP 1 UNTIL stat);
                 $llc_ds(mac1,ds1)

&*************************************************************************
& LLC comprised in the medical database server on token ring #2
&*************************************************************************
/STATION/ NAME = llc2(ws+1 STEP 1 UNTIL stat);
                 $llc_ds(mac2,ds2)
```

```
&*************************************************************************
& The other stations are basically the same than the one defined in
& in appendix D.3.1, except than higher priority services are allocated to
& the class representing the image requests. Moreover, there is no packet
& rejection in the gateways
&*************************************************************************


$MACRO macc(server)
SCHED=PRIOR;
PRIOR(pm1,pl1,pm2,pl2,ack1,ack2)=1;PRIOR(r1,r2)=2;
SERVICE=BEGIN
         ask(ir,is):=1;
         IF ring(ir).active=0
         THEN
           BEGIN
            TOK(ir,is):=NEW(CUSTOMER);
            TRANSIT(TOK(ir,is),ring(ir));
           END;
         WAIT(fl(ir,is));
         RESET(fl(ir,is));
         CST(length/tok_spd);
         ask(ir,is):=0;
         SET(poll(ir));
         IF d_ring=ir
         THEN TRANSIT(server(d_llc))
         ELSE TRANSIT(tok_dq(ir));
        END;
$END

/STATION/ NAME = mac1(1 STEP 1 UNTIL stat);
          $macc(llc1)

/STATION/ NAME = mac2(1 STEP 1 UNTIL stat);
          $macc(llc2)

$MACRO gate(llc)
SCHED=PRIOR;
PRIOR(pm1,pl1,pm2,pl2,ack1,ack2)=1;PRIOR(r1,r2)=2;
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));
         inext:=d_llc;
         TRANSIT(llc(inext));
        END;
$END

/STATION/ NAME = mac1(nod);
$gate(llc1)
```

```
/STATION/ NAME = mac2(nod);
$gate(llc2)

$MACRO token (macc)
  SERVICE = BEGIN
              active:=1;
              y:=RINT(1,nod);
              WHILE mac >= 0 DO
                BEGIN
                 x:=x+1;
                 mac:=mac+1-INTREAL(mac/nod)*nod;
                 IF ask(ir,mac)=1
                 THEN
                   BEGIN
                    x:=0;
                    SET(fl(ir,mac));
                    WAIT(poll(ir));
                    RESET(poll(ir));
                    CST(dusty);
                   END;
                 WHILE ask(ir,nod)=1 DO
                   BEGIN
                    x:=0;
                    SET(fl(ir,nod));
                    WAIT(poll(ir));
                    RESET(poll(ir));
                    CST(dusty);
                   END;
                 IF x > nod+y
                 THEN
                   BEGIN
                    x:=0;
                    active:=0;
                    TRANSIT(OUT);
                   END;
                END;
            END;
          TRANSIT = ring(ir);
          INIT(token) = IF ir=icr THEN 1
                                  ELSE 0;
$END

/STATION/ NAME = ring(1);
          $token (mac1)

/STATION/ NAME = ring(2);
          $token (mac2)



$MACRO in_gat_p(dm,node)
   BEGIN
    CST(t_conv);
    FOR x:=2 STEP 1 UNTIL nbseg DO    & one I-frame --->"nbseg" segment
      BEGIN
       CST(t_sgm);
       SEG(icr,icl,sn,x):=NEW(CUSTOMER);
       SEG(icr,icl,sn,x).d_ring:=d_ring;
       TRANSIT(SEG(icr,icl,sn,x),node(ir),dm(icl));
```

```
      END;
     CST(t_sgm);
     TRANSIT(node(ir));
    END;
$END

$MACRO in_gat_r(node)
    BEGIN
     CST(t_conv+t_sgm);
     TRANSIT(node(ir));
    END;
$END


&*************************************************************************
& Processor performing the conversion and segmentation of token ring frames
& in the gateways
&*************************************************************************
/STATION/ NAME = tok_dq(1 STEP 1 UNTIL r);
SCHED=PRIOR;
PRIOR(pm1,pl1,pm2,pl2,ack1,ack2)=1;PRIOR(r1,r2)=2;
SERVICE(pm1,pl1)=$in_gat_p(dm1,a_node)
SERVICE(pm2,pl2)=$in_gat_p(dm2,b_node)
SERVICE(ack1,r1)=$in_gat_r(a_node)
SERVICE(ack2,r2)=$in_gat_r(b_node)


&*************************************************************************
& Processor performing the reassembly of token ring frames in the gateways
&*************************************************************************
/STATION/ NAME = dq_tok(1 STEP 1 UNTIL r);
SCHED=PRIOR;
PRIOR(dm1,dm2,pm1,pl1,pm2,pl2,ack1,ack2)=1;PRIOR(r1,r2)=2;
SERVICE(dm1,dm2) = BEGIN
                    x:=x+1;
                    CST(t_rec);
                    TRANSIT(OUT);
                   END;
SERVICE(pm1,pl1,pm2,pl2) = BEGIN          & I-frames
                            x:=0;
                            CST(t_rec+t_conv);
                            IF ir=1
                            THEN TRANSIT(mac1(nod))
                            ELSE TRANSIT(mac2(nod));
                           END;
SERVICE(ack1,ack2,r1,r2) = BEGIN          & S-frames
                            CST(t_rec+t_conv);
                            IF ir=1
                            THEN TRANSIT(mac1(nod))
                            ELSE TRANSIT(mac2(nod));
                           END;


&*************************************************************************
& The stations simulating the "head", "a_switch", "b_switch", "end_node",
& "link", "a_node" and "b_node" stations are the same defined
& in appendix D.1 and D.3.1.
&*************************************************************************
```

```
&*************************************************************************
& Measurement of the mean delay for displaying each one of the three last
& levels of the pyramid data structure representing an image
&*************************************************************************
$MACRO print_out
  PRINT(" ");
  PRINT("TIME=",TIME);
  PRINTF("(1H,'Local traffic (req/min)    :    ',F5.1)",bg_arr1);
  PRINTF("(1H,'Remote traffic (req/min)   :    ',F5.1)",bg_arr2);
  PRINTF("(1H,'Inter-traffic              :    ',F4.1,' %')",100*inter);
  PRINT(" ");
  PRINT("RESULTS");
  FOR i:=1 STEP 1 UNTIL nb_level DO
    PRINTF("(1H,'Mean display delay for image level ',F2.0,'  :  ',F8.3)",
           i,tdelay(i)/thru(i)); & mean image display of the level "level"
  PRINT(" ");
  PRINT(" ");
$END

/STATION/ NAME = shut;
          SERVICE = BEGIN
                      FOR x:=1 STEP 1 UNTIL 9 DO
                        BEGIN
                         CST(300);
                         OUTPUT;
                         $print_out
                        END;
                      TRANSIT(OUT);
                     END;
          INIT(sh)=1;

/CONTROL/ TMAX = 3000;
          PERIOD = 20.0;
          TEST=BEGIN     & The simulation is stopped if the results converge
                IF (thru(nb_level)>0) AND (TIME>120)
                THEN
                  BEGIN
                   tthru(ind-1):=tthru(ind);
                   tthru(ind):=thru(nb_level);
                   IF tthru(ind) > tthru(ind-1)
                   THEN
                    BEGIN
                     mdelay(ind-2):=mdelay(ind-1);
                     mdelay(ind-1):=mdelay(ind);
&*************************************************************************
& Calcul of the mean image display encountered by the measured workstation
& wanting to read the full resolution of the image
&*************************************************************************
                    mdelay(ind):=tdelay(nb_level)/thru(nb_level);
                    $print_out
                    IF (ABS(mdelay(ind)-mdelay(ind-2))/mdelay(ind) < 0.01) AND
                       (ABS(mdelay(ind)-mdelay(ind-1))/mdelay(ind) < 0.01)
                    THEN STOP; & If results converge, the simulation is stopped
                   END;
                  END;
                END;
    ENTRY=BEGIN
          PRINT("BROWSING INCLUDING PROGRESSIVE IMAGE TRANSMISSION");
          PRINT("-----------------------------------------------");
          PRINT(" ");
```

```
          PRINT(" ");
          PRINT("PRIORITY MODE IN TOKEN RING");
          PRINT("--------------------------");
          PRINT(" ");
          PRINT("INPUT PARAMETER --> IMAGE");
          PRINT("Image size                 :   1024 X 1280 pixels");
          PRINTF("(1H,'Max segment length       :   ',F5.0,' bytes')",m_info);
          PRINT("The last three levels of the pyramid are sent.");
          PRINT(" ");
          PRINT("INPUT PARAMETER --> TOKEN RINGS");
          PRINT("Token ring bit rate       : ",tok_spd);
          PRINTF("(1H,'Nb of rings                :  ',F3.0)",r);
          PRINTF("(1H,'Nb of DBase servers/ring   :  ',F3.0)",ds);
          PRINTF("(1H,'Local traffic (req/min)    :   ',F5.1)",bg_arr1);
          PRINTF("(1H,'Remote traffic (req/min)   :   ',F5.1)",bg_arr2);
          PRINTF("(1H,'Inter-traffic              :   ',F4.1,' %')",100*inter);
          PRINT("  ");
          PRINT("INPUT PARAMETER ----> DQDB MAN");
          PRINTF("(1H,'Nb of slots/frame          :   ',F3.0)",nb_sl);
          PRINTF("(1H,'Nb of non iso slots/frame  :  ',F3.0)",nb_niso);
          PRINTF("(1H,'Nb of access units         :  ',F3.0)",au);
          PRINTF("(1H,'Looped circumference (KM)  :  ',F4.1)",bus_len);
          PRINT("Frame period (sec)        : ",fr_time);
          PRINT("Bit rate (bits/s)         : ",dq_spd);
          PRINT("Non-iso bit rate (bits/s) : ",nb_niso*sl_len/fr_time);
          PRINT("Time between stations     : ",n_to_n);
        END;
      EXIT= BEGIN
          $print_out
        END;

/EXEC/ BEGIN
        FOR i := 1 STEP 1 UNTIL ind DO
        mdelay(i):=1E-9;

        FOR i := 1 STEP 1 UNTIL r DO
          BEGIN
           ring(i).ir:=i;
           tok_dq(i).ir:=i;
           dq_tok(i).ir:=i;
           token(i).icr:=i;
          END;
        FOR i := 1 STEP 1 UNTIL ds DO
          BEGIN
           ds1(i).ir:=1;
           ds2(i).ir:=2;
           ds1(i).is:=i;
           ds2(i).is:=i;
           ds1(i).first:=1;
           ds2(i).first:=1;
           ds1(i).inext:=i+ws;
           ds2(i).inext:=i+ws;
           llc1(i+ws).inext:=i;
           llc2(i+ws).inext:=i;
           pm1(i).icr:=1;
           pl1(i).icr:=1;
           pm2(i).icr:=2;
           pl2(i).icr:=2;
           pm1(i).icl:=i;
           pl1(i).icl:=i;
```

```
    pm2(i).icl:=i;
    pl2(i).icl:=i;
    pm1(i).nbseg:=c_sg;
    pm2(i).nbseg:=c_sg;
    pl1(i).nbseg:=c_sg;
    pl2(i).nbseg:=c_sg;
    pm1(i).length:=m_pck*byte;
    pm2(i).length:=m_pck*byte;
    pl1(i).length:=m_pck*byte;
    pl2(i).length:=m_pck*byte;
    dm1(i).icl:=i;
    dm2(i).icl:=i;
   END;
FOR i:=1 STEP 1 UNTIL nod DO
   BEGIN
    mac1(i).ir:=1;
    mac2(i).ir:=2;
    mac1(i).is:=i;
    mac2(i).is:=i;
    mac1(i).inext:=1;
    mac2(i).inext:=1;
   END;
FOR i:=1 STEP 1 UNTIL stat DO
   BEGIN
    llc1(i).ir:=1;
    llc2(i).ir:=2;
    llc1(i).is:=i;
    llc2(i).is:=i;
   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;
    b_node(i).in:=i;
    a_node(i).ib:=1;
    b_node(i).ib:=2;
   END;

FOR i := 1 STEP 1 UNTIL ws DO
   BEGIN
    ws1(i).ir:=1;
    ws2(i).ir:=2;
    ws1(i).is:=i;
    ws2(i).is:=i;
    ws1(i).first:=1;
    ws2(i).first:=1;
    ws1(i).bg_arr:=bg_arr1/60.0/bg;
    ws2(i).bg_arr:=bg_arr2/60.0/bg;
    ws1(i).y:=2;
    ws2(i).y:=1;
    r1(i).icr:=1;
    r2(i).icr:=2;
    r1(i).icl:=i;
```

```
            r2(i).icl:=i;
            r1(i).length:=s_frame*byte;
            r2(i).length:=s_frame*byte;
            ack1(i).icr:=1;
            ack2(i).icr:=2;
            ack1(i).icl:=i;
            ack2(i).icl:=i;
            ack1(i).length:=s_frame*byte;
            ack2(i).length:=s_frame*byte;
          END;

        nb_seg(1):=40;     & number if I-frames representing the 1st resolution
        nb_seg(2):=120;    & number if I-frames representing the 2nd resolution
        nb_seg(3):=480;    & number if I-frames representing the full resolution
        r2(ws).icr:=1;
        choice(1):=0.2;    & Probility to receive the 2nd resolution level
        choice(2):=0.8;    & Probility to receive the full resolution image
        choice(3):=1E-15;
        SIMUL;
      END;
/END/
```