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.

ISBN 0-315-56458-X

Performance Evaluation of a Multimedia Communications System

by

Periklis Tsingotjidis

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

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

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

I hereby declare that I am the sole author of this thesis. I authorize The University of Ottawa to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Periklis Tsingotjidis

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

Periklis Tsingotjidis

στους γονεὶς μου

Ιολὰνδα και Στὲλιο

Abstract

The introduction of multimedia communications within a hospital is recognized as being an important step towards the improvement and facilitation of patient care. Pictures generated by diverse digital image instrumentation within the radiology department, can be combined with voice annotation and text to form multimedia patients' folders.

In this thesis, performance measures for a proposed multimedia communications system between the radiology and emergency departments in a hospital are determined by the use of a simulated model. The system connects a number of image display workstations and image storage devices on a network.

The impact of different medium-access protocols in image response time is investigated. CSMA/CD and the recently proposed standard for Metropolitan Area Networks, FDDI-II, are examined. For the latter one an analytical model is built using stochastic Petri-nets. Simulation results are compared with analytical results for this integrated services protocol.

The workload is specified as a mix of two user activities such as image display and magnification display requests. Inputs to the simulation model are system parameters such as transmission medium capacity and number of workstation connected to the system.

Acknowledgements

First I would like to express my gratitude to my thesis supervisor Dr. N. Georganas for his patience, encouragement and precise guidance.

Also the financial assistance of the Onassis' Foundation, the University of Ottawa and the Hellenic Canadian Association is deeply recognized. Without their support the completion of this thesis would have been impossible.

Moreover I would like to thank Professor C. M. Woodside for his permission to use the GreatSPN software package. Also the friendship and help of all my colleagues and that of Dr. Luis Orozco Barbosa and Dr. Rungroj Kositpaiboon is highly appreciated.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Multimedia Documents and Systems

Apart from natural language, the document plays a central role in the exchange of information. Increasingly, software tools for personal workstations facilitate the handling of electronic documents. Until recently these documents were composed of alphanumeric data only. The addition of other kinds of information, for example image and voice, in these documents presents a great interest. Multimedia electronic documents can be enormously useful tools to teaching, research and learning, while also providing great help in military, industrial and domestic applications.

Working, however, with multimedia documents necessitates a fundamental, common understanding of the structure of these documents. For this reason, international standard committees (CCITT, ISO and ECMA) are currently making great efforts to draw up standards that will enable the interchange of documents among open systems [HORA85].

Besides these efforts for standardization, several trials have been made in the field of communicating these multimedia documents between different processes running on different computer systems.

At the University of Southern California an experimental multimedia mail system was developed and implemented under the sponsorship of the Defense Advanced Research Projects Agency (DARPA). With this system the user can create messages containing

text, image and voice data and send such messages to other users in the ARPA Internet [REYN85].

The CCWS (Control and Command Workstation) system is another computer-based multimedia information system, that is aimed at providing computer-based tools for the exchange of multimedia information in both store-and-forward and real-time modes, and for the management and processing of such information in a human-oriented manner [POGG85].

In Japan, a distributed interoffice mail system was experimented [SAKA85]. Diamond, another multimedia message system was developed in the BBN laboratories [THOM85].

Some work has been done in multimedia conference systems [FORS85,SARI85,LANT86, STEF87]. A series of standards known as X-400 for Message Handling Systems [CCIT84] has been published by CCITT. This series of standards defines architectures and protocols mainly for electronic mailing systems. Most of the currently work in electronic mailing systems is based on this standard series.

1.2 The MUSIC Project

1.2.1 Multimedia Communications in Medicine

As it was stated previously multimedia documents have different application areas. Here their application in the medical environment will be discussed.

Hospital operations generate, for each patient, a large number of data sets of different types and formats, including handwritten requests, typed forms, computer based data, voice messages and reports, and diagnostic information such as electrocardiograms and images. We note that the patient files are really multimedia documents, and, therefore, require the use of multimedia communications and storage technologies. These multimedia documents containing voice, image and text segments need to be shared during consultation sessions between an attending physician and a radiologist. We refer to teleconsulta-

tion as the process of carrying out a consultation at a distance through telecommunication facilities.

Teleconsultation has a vital role to play in both intra- and inter-hospital consultation. An example of the first is a consultation between a physician in the critical care unit and a specialist located in his office or laboratory. An example of the second is a consultation between a physician in a remote clinic and a specialist in a tertiary care hospital. The rapid communication of diagnostic information together with the patient files will lead to improved patient care and can reduce the length of time spent in the hospital by the patient. Both factors are important incentives for the introduction of a multimedia communications system.

At the University of Ottawa, within the MUltimedia System for Integrated Communications (MUSIC) project, a prototype multimedia radiology communication system has been implemented and is intended to be installed for clinical in-hospital evaluation. This system will link the Departments of Radiology and Emergency in a large hospital (Ottawa Civic Hospital), and provide the following features :

- *Remote Access:* The referring physician will have remote electronic access to the radiographs and dictated reports, thus greatly reducing the waiting times.

- *Multimedia reports:* The voice reports will be linked directly to the radiographs in an animated form. Thus when the report speaks of some findings on an image, the corresponding image will be shown and the area of interest highlighted.

- *Shared visual Workspace:* The physician-radiologist consultation process will be facilitated through the creation of a shared visual workspace on two remotely located workstations.

1.2.2 The Radiology Communication Process

Figure 1.1 illustrates the flow of radiological information in the traditional setting. From

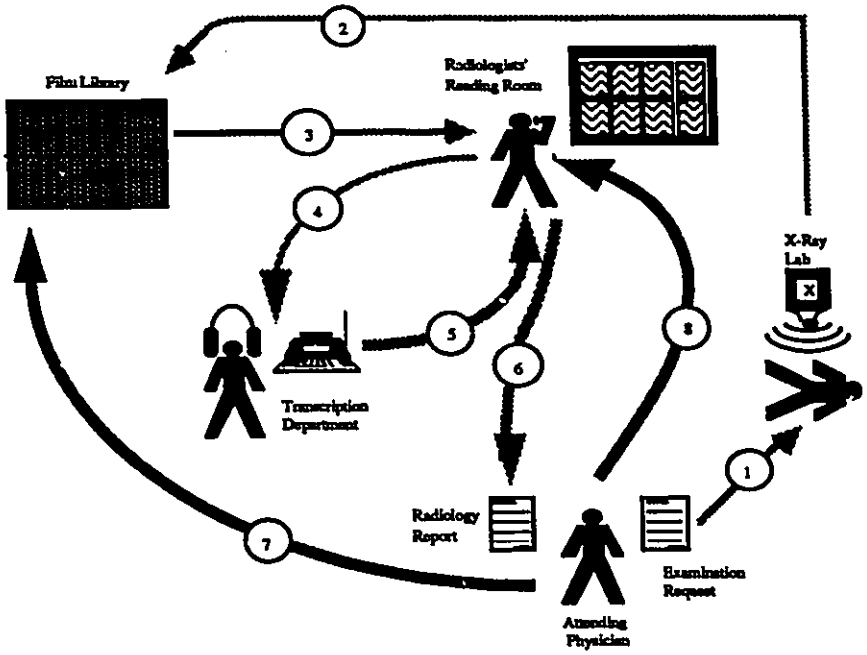

Figure 1.1: Path of the radiology examination/consultation in a conventional radiology department.

this diagram it is obvious that this information flow involves considerable movement of people, paper and X-ray film from place to place with all the attendant opportunities of delay and loss. Following the path of the typical radiological in-patient examination we find the ensuing (numbers in parenthesis correspond to the numbers assigned to the arrows in the figure):

- *Ordering:* On his morning rounds the physician orders a radiological examination for a patient. This is done by filling out a form which has spaces for the type of examination and whatever clinical information the physician decides to include. This form is sent by intra-hospital mail to the radiology department where the examination is scheduled (1).

- *Acquisition:* The patient is taken to the radiology department and the examination is performed. Once the films are developed, they are sent to the film library where they are logged and collated with the patient's existing radiology file (2). The patient's entire radiology file is then added to the batch of files that will be sent to a particular radiologist for interpretation and reporting (3).

- *Reading:* The radiologist reads the cases and reports them by dictating his or her opinion into a dictating machine. At some point in the day the dictation tapes are picked up and delivered to the transcribing pool (4). Here the reports are transcribed and then returned to the radiologist, in batches, for verification and signature (5). The signed reports are returned to the transcription pool for distribution to the film library, the primary care physicians and other locations. On average it takes from three to five days for the radiologist's report to reach the patient's chart from the time a primary care physician orders a radiological examination (6).

- *Consultation:* If the primary care physician needs or wants the results of the radiographic examination before the report reaches the patient's chart, he can find and interpret the images himself, find the images and a radiologist and have the radiologist interpret the images, or telephone the radiology department and, if the radiologist's report has been transcribed have the clerk read it to him/her (8).

The previously described procedure is expected to be modified after the introduction of a system capable of handling and communicating multimedia documents inside the

hospital as follows:

- *Ordering:* The physician orders a radiological examination via the multimedia communications system. He/she can leave voice messages for the radiology staff concerning the patient and consult with either the technicians or a radiologist to determine the optimum type of examination etc.

- *Acquisition:* The patient is taken to the radiology department and the examination is performed. Once the films are developed they are digitized and automatically collated with the patient's chart. At this point the images are available to the physician via his/her workstation. The system assigns the case to a radiologist and puts it into his/her message queue, with a priority code depending on the patient's condition.

- *Reading:* The radiologist, as before, reads cases as time permits. However, he/she now has the patient's detailed medical history at hand and possibly additional information in the form of a voice message from the clinician. The radiologist can prepare multimedia reports so that features of interest in the images are enhanced in concert with the radiologist's voice report. Since the radiologist dictates his report into the system it is available to both the transcription pool and the primary care physician as soon as the radiologist has finished the report.

- *Consultation:* This takes place through the system, either asynchronously by leaving voice messages or synchronously using the shared visual workspace. In either case they will make use of image, text and voice data simultaneously [GEOR88]

1.2.3 Prototype System's Configuration

Each workstation consists of a control screen, an image screen and a regular telephone set (Fig. 1.3). The telephone set is connected directly to a PBX (Fig. 1.2) and is used

for recording or playing back of voice messages, or for real-time voice communications during conference sessions. The image screen is a 1000 line high bandwidth colour monitor. There the digitized radiographs are displayed and different image enhancements appear. These enhancements permit highlighting of different areas of a radiographic image through a magnification facility of the workstation. Magnification, reverse video, and angle and distance mensuration comprise the possible available image enhancements at the workstation. The image screen is driven by an IMAGRAPH image/graphics process-

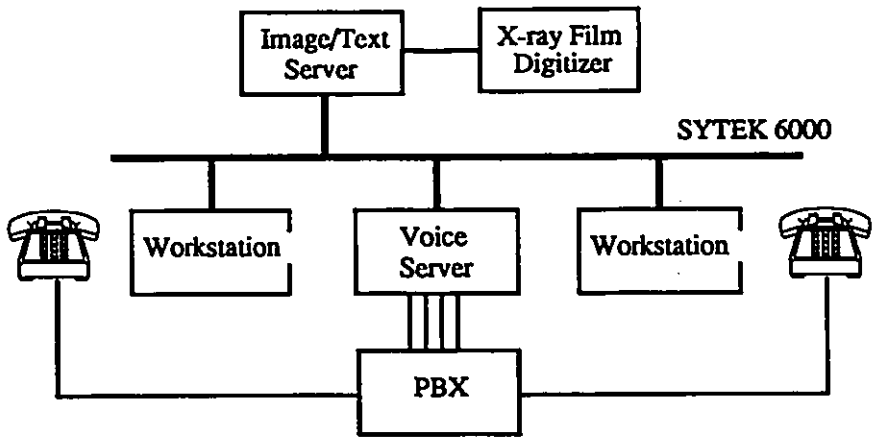

Figure 1.2: The experimental system's overall configuration.

ing system which is connected to the PC. Another monitor connected to the PC is used as a control screen. Here, a pointing device (mouse) is used to select different actions, or different views and enhancements of the already displayed image at the image screen, as explained previously (Fig. 1.3).

The image/text server is also PC-based. Here, all the copies of the digitized radiographs are kept. In fact, there are two versions of the same image stored at the image/text server. The 1024 $\times$ 1024 pixels one, which is the one that is actually transmitted to the workstations and displayed at their image screen, and a higher resolution copy of the same image of 2048 $\times$ 2048 pixels. Both of them use 256 grey levels or otherwise 8 bits/pixel,

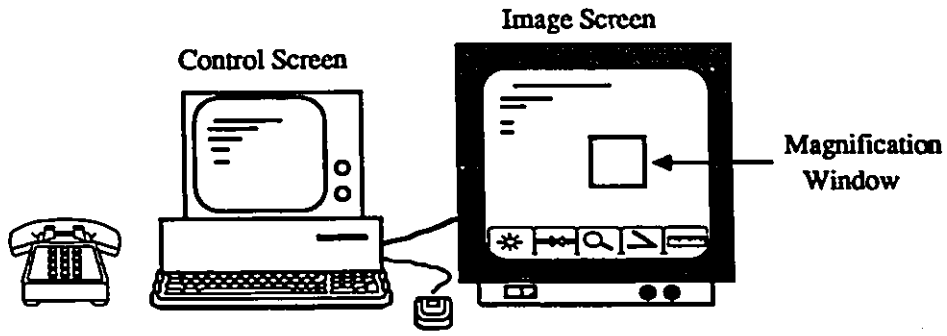

Figure 1.3: A multimedia workstation

resulting in sizes of 1 Mbyte and 4 Mbytes correspondingly. The higher resolution version of a radiograph is useful during the magnification displays, where zoom on specific areas is requested by the users. The loading of the images at the memory of the image/text server is accomplished through a X-ray film digitizer via a special PC port.

In addition, all the PCs host the necessary firmware for interfacing the interconnecting network. For our prototype this network is a SYTEK 6000 local area network. This was chosen for conformance with the network that is already at use at the Ottawa Civic Hospital, where the whole prototype system would be installed, and go under thorough evaluation from the medical community. This LAN uses broadband transmission with a transmission rate of 2 Mbits/s. The protocol used is the CSMA/CD, and it is explained in detail in a later chapter. Studies concerning other kinds of networks will be presented in a following chapter.

The previously described system can be used by the doctors in the hospital for two different purposes:

- *Disease diagnosis.* In this case the radiologist or a physician sitting at his/her workstation may like to see some radiographic images of a particular patient for diagnostic purposes, such as making or consulting a diagnostic report. Any existing

voice annotations are played back during the viewing of the images through the telephone-set.

- *Conferencing.* This mode is invoked whenever a physician would like to discuss a patient case with a radiologist for possible clarifications in real-time. Then, first a conferencing session is established in between them and after multimedia documents from the patient's folder are retrieved and displayed simultaneously in both their workstations.

1.3 Description of the Thesis

This chapter gives an overview of already held trials in the multimedia communication systems field. Moreover the effort begun at the University of Ottawa for applying the multimedia technology into the medical environment within the MUSIC project is explained. Specifically a prototype multimedia communications system for the connection of the Radiology and Emergency departments within a hospital is described.

Chapter 2 gives a broad overview of the conventional protocols used for local area networks (LANs) mainly at the medium access layer level. A simulation model of the well-known CSMA/CD protocol is presented. Finally the emerging standards for the metropolitan area networks (MANs) are discussed. These standards try to integrate the transmission of different media (data/voice/images) within a fairly wide geographical area (larger than the LANs).

Chapter 3 gives one of our original contributions, namely the simulation and analytical models of FDDI-II, a proposed protocol standard for MANs. Petri-nets are used as a tool for the analytical model's construction. Performance measures concerning the throughput and delay of an FDDI-II network are derived using these models. Finally the results from both models are compared, evaluating the validity of our analytical model.

In chapter 4 the simulation models of FDDI-II and CSMA/CD that were developed in chapters 3 and 2 are used to obtain performance measures concerning the prototype multimedia communications system that was described in chapter 1. Several possible configurations and transmission rates are examined. This is another contribution of this thesis.

Finally in the last chapter conclusions are drawn concerning the previously examined protocols and system configurations, and the most appropriate configuration for our application is proposed.

Chapter 2

Local and Metropolitan Area Networks

2.1 Local Area Networks

Trying to define a Local Area Network (LAN), we could say that a LAN is a resource-sharing data communications network which is limited in geographic scope to the range of 0.1–10 km, provides high bandwidth communication (above 1 Mbit/sec) over inexpensive transmission media and is usually privately owned [TSAO84].

LANs can be classified by topology, media access method and transmission medium.

LAN topologies. Local Area Networks physically could have a star, bus, ring or tree configuration. These configurations are the most frequently encountered in the literature. The distinctive feature of a *star* is the existence of a central switching point, known as a hub, which is shared among all the stations. Stations are connected to the hub by means of point-to-point transmission links. The major drawback of this set-up is the existence of a common central point susceptible to failures. A *tree* consists of several links which direct the transmissions from every station towards a common end point being the root of the tree, called the headend. Then data is broadcasted from the headend back through the same tree towards the interconnecting devices. A *ring* is composed of several unidirectional point-to-point links connected together by active repeaters in a closed loop.

Each station communicates with the ring through one of the repeaters with a short line. A *bus* is a long broadcast medium. Stations are attached to the medium through taps better known as bus interface units (BIUs).

Logically however, there are only two network topologies: the broadcast bus and the ring with active repeaters. The significant difference between them is that, in the first configuration, what is being transmitted by one station on the bus can be heard by all the other stations, while on the ring, transmissions have to be relayed by the repeaters one by one in order to propagate around the ring. This difference has an influence on the choice of the media access protocols that can be applied to each configuration. Moreover, looking the configurations from the reliability point of view, the ring topology is much more vulnerable to failures and provision has to be taken to address this weakness.

The most popular *medium access methods* are :

1. CSMA/CD.

2. token bus.

3. token ring.

A fairly extensive description of each of the medium access strategies is given in a subsequent section.

The most used *transmission media* in Local Area Networks' implementations are the wire (twisted pair), the coaxial cable (baseband and broadband) and recently the optical fiber. *Coaxial cable* is very common in broadcast bus configurations because of its quite good bandwidth-distance characteristics and the tapping easiness. *Optical fibers* on the other hand can increase our ability to transmit at higher data rates for longer distances without using any repeaters, but they are very difficult to tap. Therefore their usage is limited mostly to rings and more generally to point-to-point connections. Finally *twisted*

pair is used mostly in ring applications, where repeaters are needed in short distances to recover the conveying signal, because of the very poor bandwidth-distance characteristics of the wire. However, there are cases where bus configurations are implemented in twisted pair (twisted pair ETHERNET, twisted pair STARLAN etc). For a more detailed study of the existing LAN topologies, medium access schemes and technological trends, the reader is encouraged to consult reference [STAL84]. Star configurations were not given the same attention as the ring and bus architectures, at least in the recent literature, although there exist several commercial applications of networks with star configuration.

As it was stated previously the advent of the fiber optics technology with the very large bandwidth $\times$ distance product opened new design issues at the local area networks field. The up-to-then favoured bus topology had to be modified to guarantee same signal level across receivers.

Also the advent of fiber optics had an impact on another related field, namely the integration of voice/data/image in the local area environment. Voice/data integration at the long haul transmission level was investigated first, and interesting solutions can be found in the literature. In that environment transmission bandwidth is an expensive resource and sophisticated protocols were employed to efficiently use the available bandwidth. On the other hand, in the local area environment, bandwidth is not a very precious resource so less complex protocols can be used allowing less expensive implementations [MAXE82].

2.1.1 CSMA/CD, Token Ring, Token Bus.

As aforesaid the most promising medium access methods, which have already been standardized by the IEEE, are the CSMA/CD, token ring and token bus medium access strategies. Here a brief description of the official standards, approved by the IEEE 802 committee, is given.

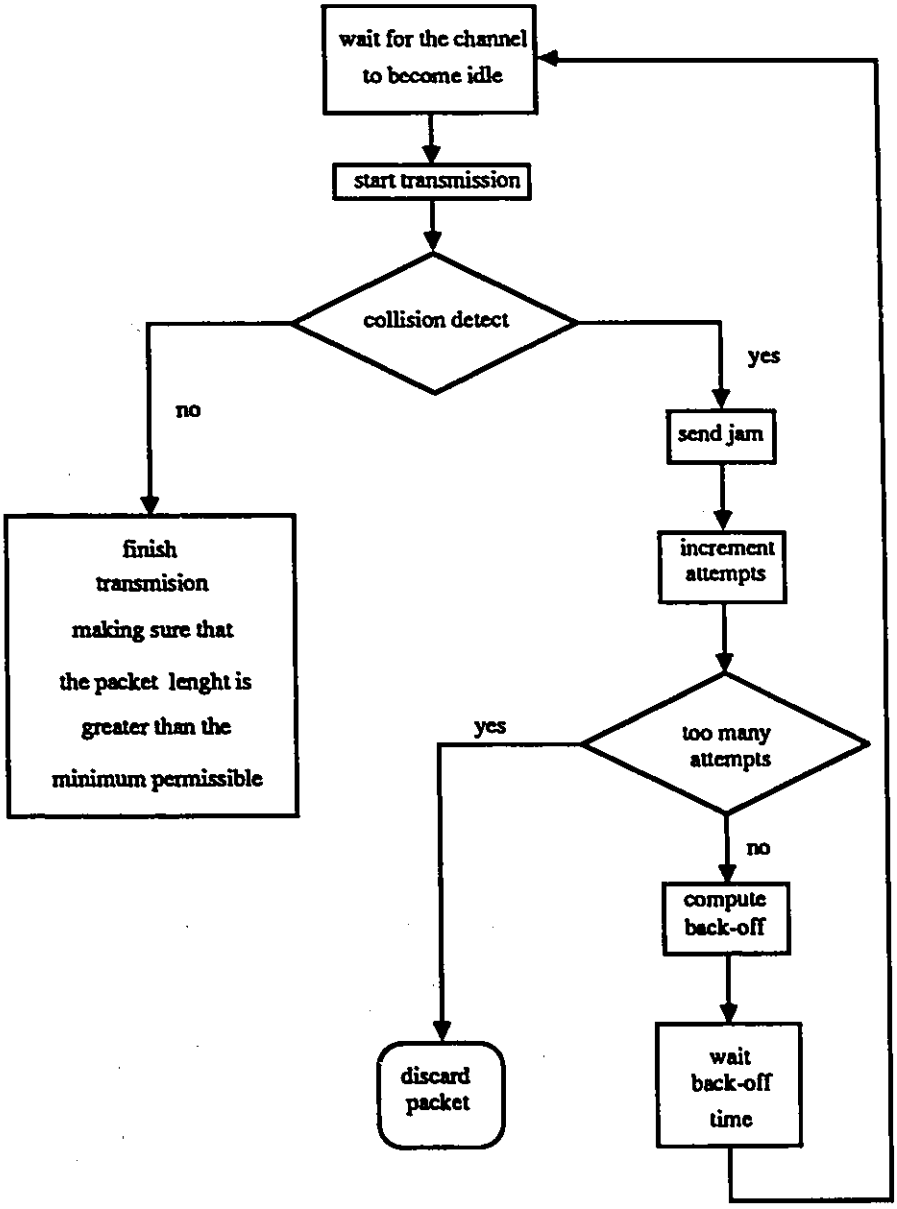

Figure 2.1: The flow control chart of the IEEE 802.3 standard

The IEEE 802.3 Standard (CSMA/CD)

This is a random access method of assigning the transmission bandwidth of a broadcast medium to the stations attached to the medium. Stations that have a newly arrived packet to transmit would first sense the channel. If it is idle they will transmit right away but they continue monitoring the channel. As soon as a collision is detected they will cease transmission and transmit a jamming signal to assure that all stations know that there has been a collision. Then the stations that have been encountered a collision refrain from seizing the channel for a random interval of time. In Ethernet and the IEEE 802.3 standard a binary exponential back-off technique is used. According to this strategy the average random delay that a station waits until its next trial to transmit is doubled before every next attempt. This strategy is used up to the first ten transmission attempts. After the tenth unsuccessful trial the size of the average back-off interval remains the same. Moreover transmission trials are repeated up to the sixteenth trial in a row and after that the packet is discarded and the Logical Link Control (LLC) layer is informed accordingly. Figure 2.1 gives the control flow chart of the process already described [IEEE84].

The IEEE 802.5 Standard (token ring)

This medium access method is not random, as the CSMA/CD method described previously. The medium access is controlled by the circulation of a distinctive bit sequence on the ring, called the token. Within the token, there are groups of bits related with the operation of the ring with priorities. These bits comprise the priority and reservation fields of the token and their use in the operation of the protocol is explained in the sequel.

During normal operation, without priorities, a station waits for a free token to pass by. Whenever it has a packet to transmit and it detects a free token passing by, it transforms the free token into a busy one by simply setting a special bit in the token, called the

token bit, to one. Then it begins transmitting until either its queue becomes empty or a special timer, called the token holding timer (THT) expires. After completing its transmission, the station waits for the busy token to return and then it issues another free token. Such an operation, guaranteeing the existence of only one token on the ring, is called single-token operation.

The IEEE 802.5 standard allows the existence of various priority levels during the ring operation. In each token there are three bits in the access control field that indicate the priority of the token. Only stations that have packets of higher priority than the one indicated by the token can make use of the free token for transmission. All the other stations that have a packet to transmit, can only reserve a token at their priority level using the reservation bits of the token passing by. When a station gets the free token it transforms it to a busy one leaving the priority field of the token unchanged and setting the reservation field to 0. Now the stations downstream can reserve a token at their priority level using the reservation bits of the token. But if a reservation of higher priority has already been made from an upstream station, they let the token pass unchanged and they hope for a later chance. When the busy token returns back to the station that issued it and also the station had finished transmission, it issues a new token with the priority bits set to the maximum of the P_r, R_r and P_m where P_r, R_r, and P_m are defined as the received priority of the token, the received reservation and the priority of the message queued at the station waiting for transmission. The reservation bits are set to the maximum of R_r and P_m.

When a station detects the free token and the priority level of its queued packet is higher than the priority of the token, the station uses the reservation bits to reserve the next token for itself, if there was no upstream station that had reserved at a higher priority.

To prevent perpetual circulation of a token of priority in the ring, caused by the

previous described method of increasing the priority level of the token, provision has been taken and has given authorization to the station which upgrades the priority of the token to downgrade it to its former setting when all higher priority stations are finished. When the station sees a token at the higher priority, it can assume that there is no more higher priority traffic waiting, and it downgrades the token before passing it on [IEEE85,STAL84].

The IEEE 802.4 (Token Bus)

A broadcast medium is used to connect all the stations. However, the access is not random as in CSMA/CD, but it is controlled by the circulation of a token. There is no physical ring. The stations maintain a logical ring between them. By that we mean that each station knows its predecessor and successor and there is a special protocol for inviting other stations to the bus or for a station to detach itself from the bus.

The IEEE 802.4 standard breaks up the protocol into six portions, namely :

- addition of a node
- deletion of a node
- Fault management by the token holder
- Ring initialization
- Classes of service

Here only the last portion is analyzed in more detail. The standard defines three classes of services (the numbers in parentheses are these by which the class will be referred in the sequel) :

- synchronous (6)

- asynchronous urgent (4)
- asynchronous normal (2)
- asynchronous time available (0)

Also, there is one variable associated with each class of service as follows :

- THT : Token Holding Time. The maximum time that a station can hold the token transmitting synchronous (class 6) data.
- TRT4 : Token Rotation Time for class 4 data. The maximum time for a token to circulate around the ring and still permitting class 4 data to be transmitted.
- TRT2 : The same as above but for class 2 data.
- TRT0 : The same as above for class 0 data.

Stations can transmit synchronous data as soon as they receive the token for THT time. In this way the maximum amount of time that can be used for synchronous data transmission per cycle is limited to n $\times$ THT (where n is the number of stations on the ring).

After receiving the token and transmitting possibly its synchronous data, a station can continue with the other classes of data. However, the possibility of transmitting the next class of data depends on how much time is left until the predefined TRT4 time value. If there is some time left, the station can proceed with the transmission of its asynchronous urgent data. Otherwise the transmission of this class of data is postponed for a next cycle. Then the same reasoning is applied for the next class of data and finally the token is passed to the next station.

In this way a distributed priority system is created for the bus access. A guaranteed bandwidth is given to the synchronous data. The access to the bus for all the other

classes of data is not guaranteed. Preference is given to asynchronous urgent data that have the second highest priority, after the synchronous data and so on with the other classes. Lower priority data tend to be transmitted at cycles where there is not heavy transmission of synchronous data, a fact that makes the circulation of the token faster. As the last observation, the time of the next reception of the token by a station is upper limited and therefore it is guaranteed that the token will return to a station after a more or less fixed interval. That characteristic of the protocol is particularly interesting for synchronous data transmission [IEEE84,STAL84].

2.1.2 Model of Ethernet

There are several analytical studies for the performance evaluation of the CSMA/CD medium access strategy such as [LAM80] and [TOBA80]. Here a simulation approach will be presented.

In Appendix A.1 the list of the simulation program developed in our labs is given. A special software package for the analysis of queueing networks has been used, called QNAP2 (Queueing Networks Analysis Package 2).

QNAP is a software tool for describing and solving queueing network models. QNAP consists of:

- a collection of resolution algorithms, including discrete event simulation, exact and approximate mathematical methods,
- a common user interface for model description, analysis control, and result presentation [QNAP84].

Three global variables are key in the model, namely `br`, `COLLI`, and `discard`. The first one measures the number of stations that simultaneously try to seize the channel during a contention interval. If it is greater than one then two or more stations have tried to

transmit and hence we had a collision. The transmission of the same packets will be tried again later, after a binary-exponential back-of interval. The other two variables count the number of discarded packets and the number of collisions. However in our simulation model we have inactivated the packet discarding mechanism, after 16 attempts, and we continue trying to transmit a packet that has already attempted to be transmitted for 16 times. This mechanism, implemented by higher level protocols, is provided in the Ethernet standard.

Two other variables are associated with each customer. The transmission attempt's counter, which registers the number of transmission attempts a given customer has made and the back-off variable, useful at the calculation of the back-off interval. The program follows pretty close the flow control chart shown in Figure 2.1 and performance evaluation results can be derived for CSMA/CD using the appropriate form of input traffic.

2.2 Metropolitan Area Networks

Metropolitan Area Networks (MANs) constitute a new concept in networking. The main differences of a MAN from a LAN are summarized in the following list.

- Distance : city and suburbs versus a few Kilometers.
- Backbone : use for interconnecting LANs and large computers.
- Voice and video : more demanding access requirements.
- Central operation : provides maintenance and billing.
- Public operation : raises privacy and security issues [MOLL88].

Since 1981, the IEEE 802 committee tries to define a standard for this kind of networks. There is a great interest of standardization from the industry and the carriers since it is

expected that usually MANs will be operated by common carriers and in addition they would be a significant step towards a full broadband ISDN.

A most promising proposal that also has a great support from the Telcos is the dual bus QPSX MAN. Another equally promising proposal is a protocol for fiber optic implementation, called FDDI-II, supported by the ANSI. Both of them are described in the sequel.

2.2.1 QPSX or DQDB

QPSX was proposed by Telecom Australia as a candidate for the IEEE 802.6 standard [IEEE88]. It is also known as distributed queue, double bus (DQDB) MAN.

The proposed architecture is that of two contra-directional buses. Stations are attached to both the buses through access units (AUs). There are two connections for each AU to each bus. One for reading information from the bus and the other for writing onto

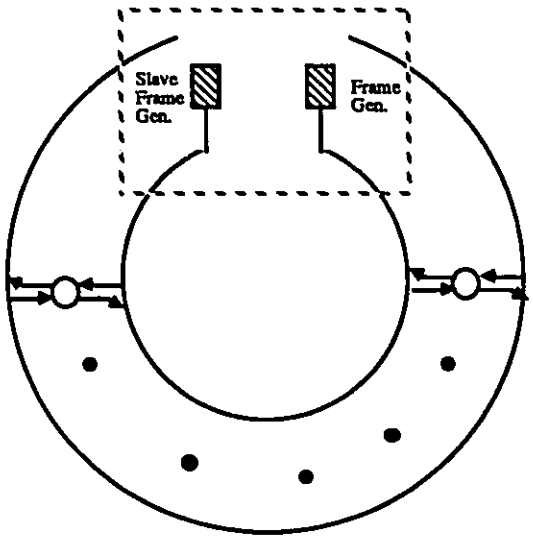

Figure 2.2: QPSX looped bus architecture

the bus. The two far end stations on both sides of the bus have a more specialized func-

tion. The one at the head of the forward bus is used to generate the frame pattern and the other at the head of the reverse bus keeps the same rate on the reverse bus (the naming of the buses has only relative importance). For improved reliability the bus is configured as a looped bus (Figure 2.2). In this way the end points of the two buses are co-located. This can give flexibility of reconfiguration after the detection of a bus fault. The system can close the data buses through the head point and become again fully operational.

Communications on the buses can be accomplished using a frame format of 125 μs. The frame is further divided into a number of slots (see Figure 2.3). A slot can be allocated to either isochronous or non-isochronous communications. Slots can be allocated to isochronous communications keeping up with the current demand of isochronous communications. The remaining slots are free for packet switched use. Within an isochronous slot each octet corresponds to a 64Kbits/s channel. By assigning more than one octet per frame higher rates can be achieved. For low rate transfer the use of a multi-frame count at the frame header is necessary.

The medium access policy used for accessing the non-isochronous slots is fundamentally different from the already existing MAC protocols. Its name is Distributed Queueing algorithm and controls the access of fixed length data segments to the slots on the QPSX bus. The difference of this protocol is that every station keeps track of the state of the network continuously. Therefore, whenever access to the network is needed, the delay is the minimum possible. On the contrary, the other MAC protocols try to schedule the transmissions on the network by information derived from the network only when they have something to transmit. The monitoring of the network's state is accomplished by the employment of a counter residing in each AU and the use of two bits in the header of each slot. Those bits are the busy/empty control bit and the request (REQ) bit. When there is a segment waiting for transmission in one of the AUs the REQ bit of one of the passing slots in the reverse bus is set. Now all the upstream stations (upstream with

respect to the forward bus) are informed that there is a segment waiting for transmission at a downstream station and their request counter (RQ) is incremented by one. In the same way when there is an empty slot passing by at the forward bus, the RQ counter is decreased by one because it is known that that slot has been reserved from a downstream station and it will be used from it. Besides this counter there is another one, the count-down (CD) counter, which is loaded with the current value of the RQ counter when a segment is ready for transmission. The value of this counter gives the actual position of the segment into the global distributed queue. This counter can only be decreased when an empty slot passes by, and transmission is attempted at the first empty slot after the CD counter reaches zero.

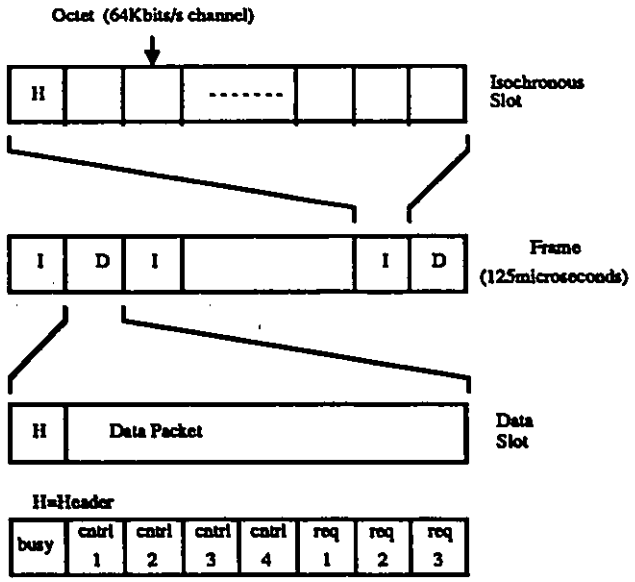

Figure 2.3: The QPSX frame structure

The previously described basic distributed queueing algorithm can be enhanced to handle priorities as well. Then more than one reservation bits have to be used in the slot header and more than one request and countdown counters at each AU.

2.2.2 Fiber Distributed Data Interface (FDDI and FDDI-II)

FDDI is a proposed American National Standard for high speed metropolitan area networks (100 Mbits/s) using optical fiber as the transmission medium. It is based on the IEEE 802.5 token ring configuration, but several modifications had to be introduced to comply with the very high data rate [ROSS86].

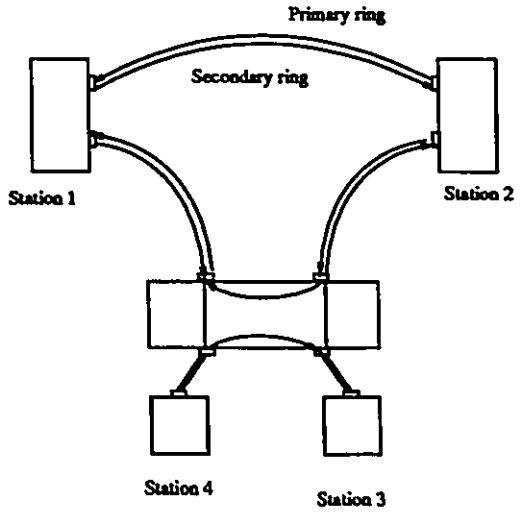

Figure 2.4: The FDDI ring topology

The FDDI physically is configured as a ring(Fig. 2.4). Each station is connected to the ring through a repeater. These repeaters regenerate the incoming signal coming from the upstream repeater and they transmit it downstream. There is another redundant ring connecting all the repeaters, which can be useful in line failures.

In Fig. 2.5 the frame and token structure is depicted. The different fields inside the frame consist of symbols, unique groupings of 4 bits, being defined for encoding purposes. The Preamble (PA) has not a fixed length and it is used for clock synchronization. The Starting (SD) and ending Delimiters (ED) establish the packet boundaries. The Frame Control (FC) distinguishes between different kinds of frames as synchronous or

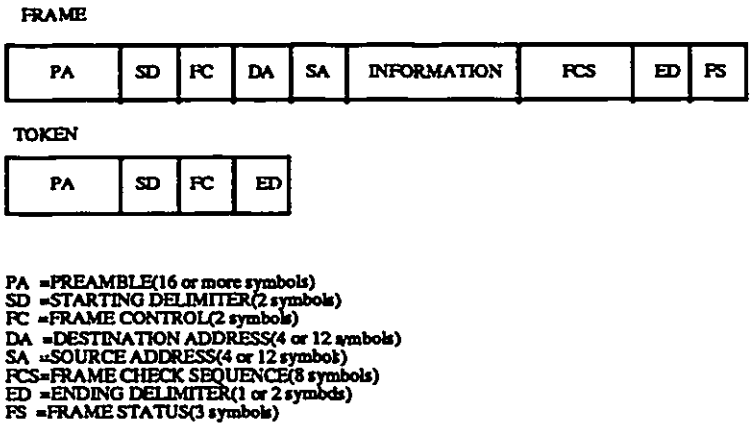

Figure 2.5: The FDDI frame and token structure

asynchronous, and between information or control frames. There are also the Destination (DA) and Source Addresses (SA) fields and the Frame Check Sequence (FCS). Finally, there is the Frame Status (FS) field for piggybacking an acknowledgement from the destination station to the source one. However, the token has only four of the aforementioned fields, namely the Preamble, the Starting and Ending Delimiters, and the Frame Control.

The IEEE 802.5 is a single-token protocol. If FDDI had been the same, the utilization of the medium would have been very low. Therefore the multiple token approach was taken. According to this approach, each station issues a new token immediately after completion of its own transmission. All the frames being transmitted by a station must be removed from the ring by the sender after they travel around the ring once. At that time notification is given to the higher protocol layers for a possible error or for the correct frame reception. Also the priority mechanism used at the IEEE 802.5 had to be changed because of the different token release policy at a station. Instead of this, a timed token protocol has been proposed.

As in the IEEE token bus standard, two timers are built in each station, the Token Rotation Timer (TRT) and the Token Holding Timer (THT). Also two classes of services

are defined :

- synchronous services, where fixed units of data are to be delivered at regular time intervals [ROSS86]
- asynchronous services, where there is no time constraint for the delivery of data units.

Synchronous data are always transmitted after a token reception from a station. On the contrary transmission of asynchronous data is not guaranteed and is conditioned on the "earliness" of the token. For that reason, directly after the token reception the station remembers the elapsed time since the previous reception into the TRT timer. When this elapsed time is short compared to a Target Token Rotation Time (TTRT) permission is granted to the station to transmit its asynchronous data packets for the time interval that equals the difference between the TTRT and the TRT. This difference is kept into the second timer, the THT. After the expiration of the THT the token is passed to the next station, downstream.

In the previous paragraph, a time value, by the name of TTRT, was mentioned. Here we would like to elaborate a little more on this. First of all, this value is known to all the stations after the ring initialization process is over, where it is determined for the first time. All the stations, that would like to communicate through the FDDI ring, knowing their synchronous data requirements, bid for the TTRT. Finally, the lowest bid becomes the operational TTRT value. By that process, it is guaranteed to all stations, even to those with very stringent requirements, that the token will return in a predetermined time. It has been proven that the average returning time for a FDDI token is no greater than the TTRT value, while it is guaranteed that the maximum return time for the token is upper bounded by the $2 \times$ TTRT [JOHN85,SEVC86].

From the previous, the operational resemblance of this protocol with the token bus

protocol is obvious. However, there is a difference in the measuring of the token return time, which furthermore has an influence in the station's decision to transmit or not asynchronous data. In FDDI, the THT timer is loaded directly after the reception of the token, whereas in the token bus protocol after the transmission of the synchronous data. In this way the time for the synchronous data transmission counts for both the protocols in different token rotation cycles. In FDDI, it counts for the next cycle, while in the IEEE 802.4 standard it counts for the recent one. The FDDI protocol has more similarities with the protocol proposed in [GROW82] and examined, at least preliminarily by Ulm in [ULM82].

Getting into more details, two kinds of tokens are supported by the protocol: the restricted and nonrestricted tokens. Restricted tokens are used only when two stations would like to communicate between each other. In this way, each of the stations issues a restricted token after finishing its transmission, permitting only its interlocutor to make use of the token. On the other hand a nonrestricted token can be used by all the stations on the network. Moreover, several priority levels can be introduced in the use of nonrestricted tokens, discriminating between different classes of input traffic.

In order to meet the requirements for MANs, FDDI had to be enhanced to support isochronous data services as well. Isochronous traffic is the traffic where fixed units of data are to be delivered at fixed time intervals relative to a single time reference. For that reason the inclusion of a circuit switching capability has been proposed. That new version of FDDI was given the name FDDI-II. This circuit switching capability has been built in such a way that inter-operability with the old protocol version is guaranteed. Actually, FDDI-II is initialized identically to the basic FDDI. The presence of any non FDDI-II capable stations on the ring prevents the activation of the FDDI-II operation mode.

To accommodate the isochronous communications, data are transmitted on the ring within cycles. Each cycle is a group of bytes, that lasts for exactly 125 microseconds.

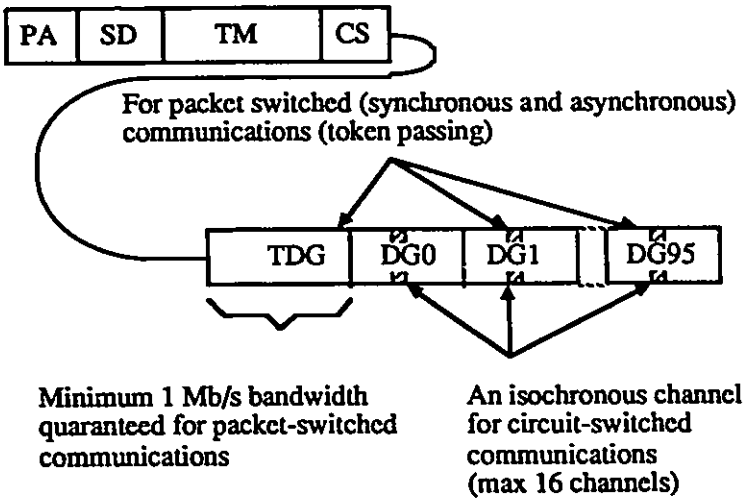

Figure 2.6: The FDDI-II cycle structure.

Provision is taken to permit the circulation of only an integral number of cycles synchronously on the ring. Each cycle is subsequently divided into fields as it is depicted in Figure 2.6. As it can be seen there are 96 separate data groups, each of which consists of 16 timeslots (8-bit bytes). Individual bytes within each data group are assigned to either circuit switching or packet switching use. However, the bandwidth apportionment algorithm ensures that byte "n" in every data group is identically assigned. In other words, if byte "0" in DG0 is assigned to packet switching, then bytes "0" in all others DGs are similarly assigned. In effect, this means that the granularity of apportionment for circuit and packet switching is 96 bytes, equivalent to 6.144 Mbit/sec (called a Wideband Channel-WBC). There are thus 16 WBCs individually assigned to either packet or circuit switching modes of operation. The WBCs are byte interleaved across the 96 data groups, WBC "n" consisting of the n^{th} byte in each data group.

However,there is always a data group, that is dedicated exclusively to token data (TDG). That gives a guaranteed minimum of 1.024 Mbits/s for packet-switched data. A frame or a token within this channel has the same format as in the non FDDI-II case except

that there is no preamble and as starting delimiters different symbols are used. Besides this bandwidth, any bytes within the 96 data groups, which were not assigned to any isochronous communications, can be used for packet-switched data transmission.

Chapter 3

Modelling of FDDI-II

3.1 Simulation Model

There are several studies, approximate and exact, analyzing the performance of token rings. A concise summary of those efforts could be found in [SEVC86]. Although FDDI belongs to this category of token rings, the peculiarities of the timed token protocol pose significant difficulties in its performance evaluation. In fact, the "timed" characteristic of the protocol, creates some dependencies among transmissions of various stations and these dependencies complicate the analysis of the protocol performance.

Ulm [ULM82] tried to analyze a similar "timed" protocol and to give some performance figures. He also investigated the dependence of some performance measures on various design parameters. Dykeman [DYKE88] continued that analysis in more detail and extended it including a study on the priority mechanism of FDDI. His results were verified by simulation tests. Before Dykeman, several other researchers had published simulation studies on FDDI as well [ATKI87,GREE87,SCIL87].

Here a simulation study of FDDI-II is presented. The simulation program was built using the QNAP2 facility of our lab. A list of the program is depicted in Appendix A.1.

As it is aforementioned in section 2.2.2, FDDI-II is an enhanced version of FDDI and their only difference is that FDDI-II can handle isochronous traffic in addition to the non-isochronous traffic of FDDI simple. The isochronous bandwidth is allocated to the

stations in increments of 6.144 Mbits/sec channels. The way this can be accomplished is not specified in the FDDI standard and is left to the upper protocol layers. The rest of the bandwidth is used for packet-switched communications (synchronous and asynchronous) in accordance with the FDDI protocol specifications. Therefore, assuming that the isochronous bandwidth allocation is kept fixed, the only difference between FDDI and FDDI-II, as far as the transmission of the packet-switched data is concerned, is the reduction of the actual transmission rate.

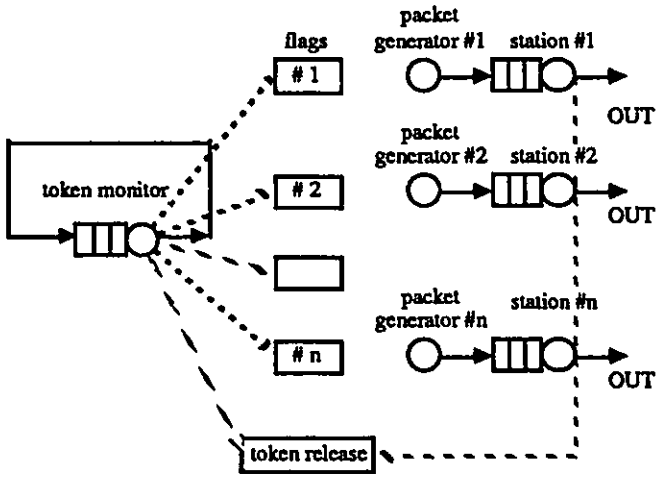

Figure 3.1: The queueing model of the FDDI-II protocol.

Our simulator program consists of three parts (Figure 3.1):

- The packet generators, which serve as an input to our model
- The stations, which actually perform the FDDI protocol
- The token monitor station, which controls the circulation of the token

Each station has its own packet generator, which generates packets of data and sends them to its attached station. Packets have variable lengths, drawn randomly from an exponential distribution. Also the interarrival times at the station, between consecutive

packets, are exponentially distributed. Packet generators create only asynchronous data packets. How synchronous data traffic is handled in our model, will be explained later.

The token monitor station is a fictitious queue that serves as the medium access control. In FDDI, each station is given the permission for transmission, in a round robin fashion. That, in our model, is achieved by the setting of flags. Each flag is associated with one of the stations. When one station's flag is set, permission is granted to that station to transmit. After finishing its transmission, the station informs the token monitor queue by setting a global flag called "release token". Then the token monitor proceeds by polling the next station. At this point the time that is required for the transmission of the token and the propagation time between two adjacent stations is taken into consideration. In our model the token is not passed to the stations that have no packets to transmit. However the token transmission and propagation time is considered in all cases. Finally, there is another function that has been incorporated into our token monitor service routine, having been adopted from Schill [SCIL87]. Whenever the ring is sensed idle (there are no asynchronous data queued for service in the stations) for more than a complete token circulation, the token is destroyed. This is done to limit the actual simulation time. Of course, the token must be regenerated when a new customer is generated in one of the stations, for the proper operation of the ring. For this reason, the first newly generated customer will also generate a token, and then the token monitor station will begin its pollings starting from a station selected at random. In Figure 3.2 the logic diagram of the actions at the token monitor station is presented.

The stations perform the FDDI algorithm. In Figure 3.3 this algorithm is given in a logic diagram form. As it is depicted in the figure, the time when the station receives the token is marked and is used to compute the token holding time (THT). In addition to that time value, only the time of the previous token reception is needed. The computed THT would determine the amount of asynchronous data that the station is allowed to

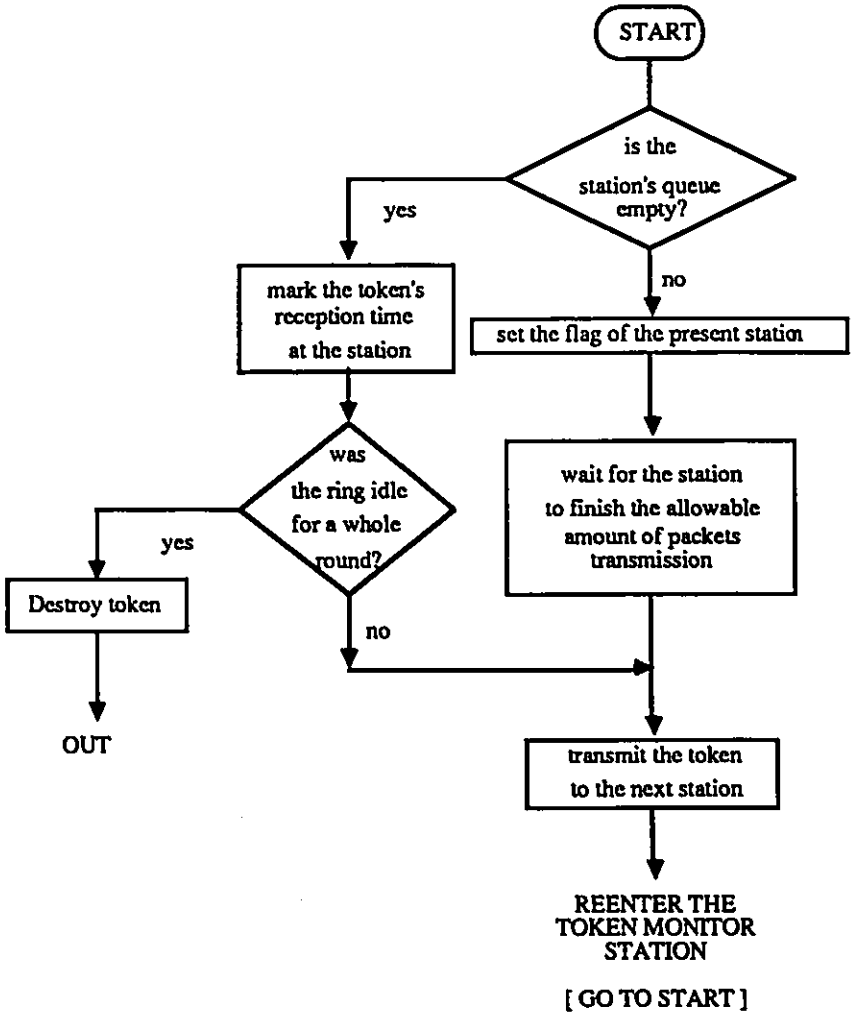

Figure 3.2: The logic diagram of the token monitor station's actions.

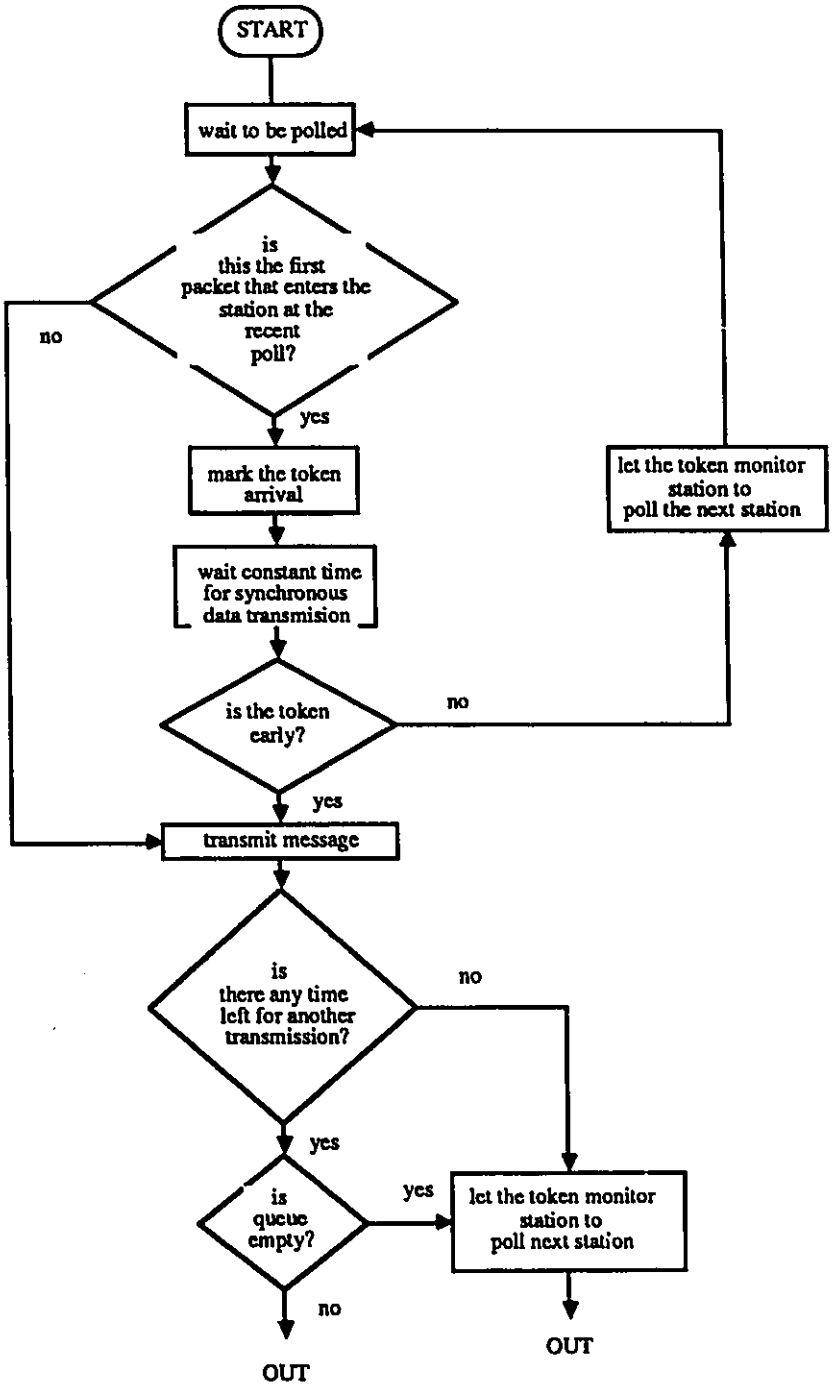

Figure 3.3: The logic diagram of the protocol procedure at the stations.

transmit. However, synchronous data is always permitted to be transmitted. Although our generators do not create any synchronous data traffic, we assume a delay for synchronous packet transmission in each station after the token reception. This delay counts for the transmission of some hypothetical synchronous data, which has been accumulated at the station during the previous token rotation around the ring. Furthermore, we assume that this synchronous data transmission delay is constant, as though a variable flow of synchronous data arrives at each station in such a way, that the transmission of the accumulated data in each token cycle is constant. Of course, provision must be taken, so as the sum of all the synchronous data transmission intervals at all the stations is less than the target token rotation time (TTRT).

After the synchronous data transmission, asynchronous data can be transmitted, if the previous calculated THT is positive. Packets enter the MAC layer for transmission one by one and the THT value is checked after each successful packet's transmission. This time value is decreased synchronously with the transmission of each packet by the amount of time that it is spent for this packet's transmission. If, during one of these checkings, it is found that the THT has a negative value, further packets transmissions are disabled at the station and the token monitor is informed accordingly.

TTRT	10ms
cable length	10km
propagation time	5.085μsec/km
station latency	600ns
synchronous data transmission time	0
cycle overhead	84 bits
frame overhead	164 bits
token length	24 bits
packet length	4096 bits

Table 3.1: Parameter values for the simulation model

In table 3.2.3 the values of the parameters that we use, are shown.

3.2 Analytical Model

3.2.1 Introduction

As it was stated in the previous section, it is difficult to build an analytical model of FDDI, because the "timed" feature of the protocol poses strong dependencies among the transmissions of the stations. Usually to analyze the token protocols the queue independence assumption is made, which permits us to easily analyze only one queue and conclude about the overall ring performance. In our case, such an assumption would be a very crude approximation giving inaccurate performance measures.

Here an attempt is made to present an analytical model of the protocol, based on a recently developed tool for systems performance evaluation, a specialized kind of networks, called Petri-nets. After developing the analytical model and deriving the performance measures, a comparison is given between these results and the results derived by simulation.

3.2.2 Petri-nets

At first a brief introduction on Petri-nets will be given. Petri-nets were developed from Petri's dissertation, who introduced them for the first time as a system's modeling tool.

Petri introduced the notion of places and transitions. Its nets consist of places and transitions connected together with arcs. Places are represented as circles and transitions as line sections. Places comprise the conditions, which should hold for an event to occur, while transitions are the actual events. Places that represent conditions for the same event have arcs, beginning from them and ending at the transition that actually represents this event. These places comprise the input places of this transition. The holding of a condition is depicted by the existence of a token in the place that represents this event. When all

the input places of a given transition have the sufficient number of tokens, the transition is enabled and by firing it removes the tokens from its input places and places them in places that comprise the consequent conditions of an event occurrence (output places). In this way the states of the system can be determined by the number and the distribution of the tokens among the places. Each state consists of a specific setting of the tokens in the net, which is called a marking. For a more detailed and formal presentation on this subject the reader is encouraged to refer to [PETE81].

Later, Ramchandani introduced time into the transitions and another class of Petri-nets, called timed nets (TPN), has emerged. When the time assigned to each transition is a random variable, we talk about stochastic Petri-nets (SPNs).

There are several subcategories in that class of nets. The categories that are of interest to us for the further development of our model are : the GSPNs (general stochastic Petri-nets) and the DSPNs (deterministic stochastic Petri-nets). In the first category, the time associated with each transition, defined as the interval from the time that the transition is enabled till it eventually fires, could be a random variable exponentially distributed (exponential transitions) or could be fixed (deterministic transitions). The state space of the GSPN and the reachability tree (a state space tree showing also the transitions between the states), could be built and the probability distribution of the tokens in the places could be found. After that, several performance measures, concerning the actual system, can be calculated. Although this method can always work in the case of GSPNs, there is one condition that must be satisfied for a DSPN, to have an analytical solution. As stated in [MARSA87] if we restrict our markings to enable at most one concurrent deterministic transition, then we can obtain our model's solution fairly easily. This restriction has a significant effect in our model's development.

3.2.3 The FDDI-II Petri-net Model

DSPN Model

To keep the state space as small as possible, so as to be able to solve our model by computer, only the case of two stations communicating with the FDDI-II protocol has been modelled. Figure 3.4 depicts this model.

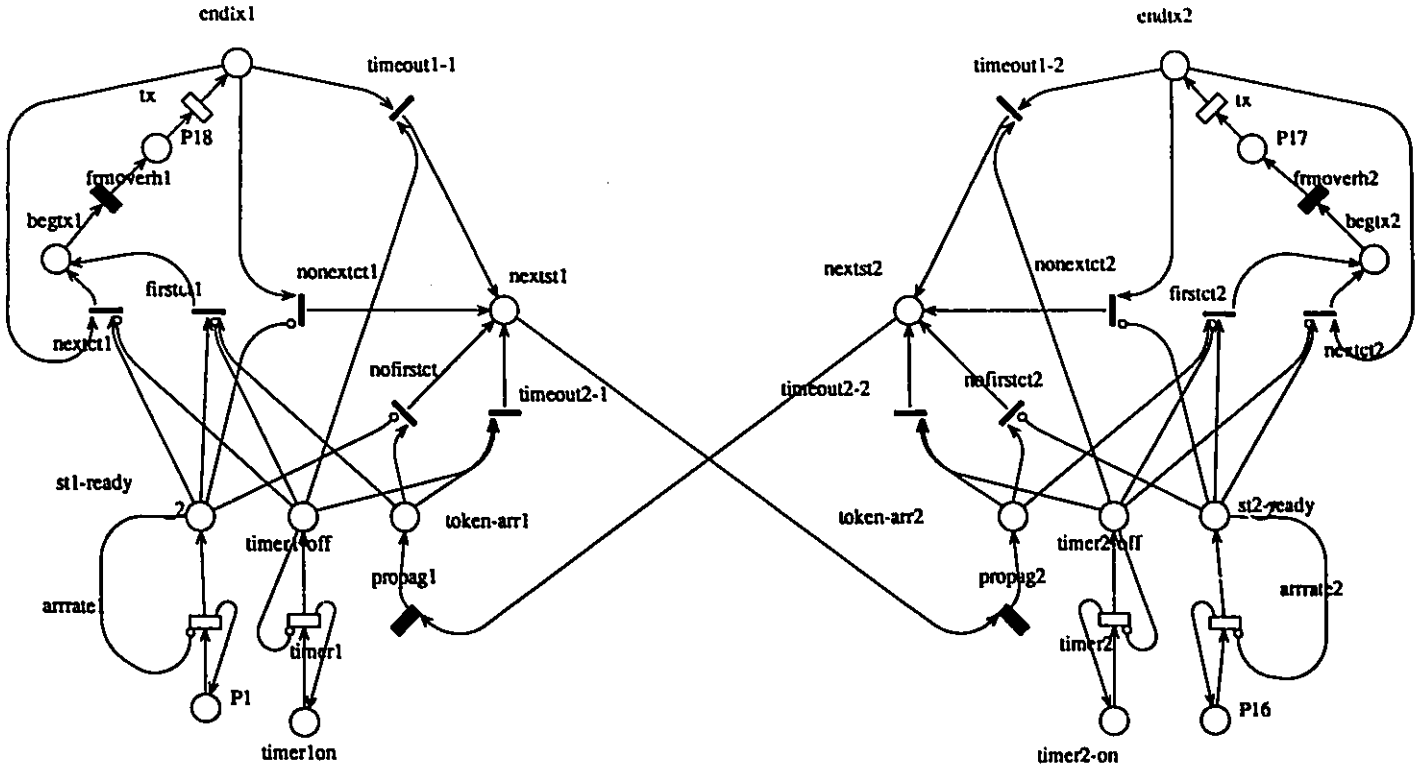

Figure 3.4: The DSPN model of the FDDI-II protocol in a ring of two stations.

Presenting the model of one station, we see that it consists of 4 distinct parts. The part that generates the costumers which require transmission through the network (places *P1* and *st1-ready* in the figure), the timer section (places *timer1-on* and *timer1-off*), the

TRANSITIONS	
arrate1	arrival rate of packets into MAC
timer1	timer (TTRT) firing rate
firstct1	permission to transmit the first packet in the queue
nextct1	permission to transmit other than the first packet in the queue
tx	the packets' transmission rate
frmoverh1	time for the transmission of the FDDI frame overhead
propag1	time for the FDDI token transmission and propagation between the sations
nonextct1	grant of service to the next station because all the packets had been served
nofirstct	grant of service to the next station because the queue is empty
timeout1-1	grant of service to the next station because the timer (TTRT) has expired during transmission
timeout2-1	grant of service to the next station because of late arrival of the FDDI token

Table 3.2: Transition's table for the DSPN Petri-net model.

transmission section (places *begtx1*, *P18*, and *endtx1*) and finally the part that represents the logic of the protocol (all the immediate transitions and the place *token-arr1*).

In Fig. 3.5 the model of the station is given. It consists of two places and a marking dependent exponential transition. Tokens represent packets that are generated at the station and need transmission through the network. There is always one token in the place P1. Whenever there are less than k tokens in the place P2, the transition t_1 will fire with a rate λ, adding a new token in place P2. Tokens in P2 represent the actual packets coming from upper protocol layers at the FDDI-II Medium Access Control layer, which require transmission. Using an inhibitor arc we limit the permissible queue length at each station to a maximum value of k packets. In our model this value is kept as small as possible for state space reduction purposes.

The timer part has the same topology as the previously explained packet generator part. The difference is that now the value of k is fixed to 2 and the exponential transition

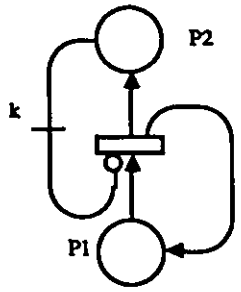

Figure 3.5: The station and timer Petri-net model.

has an average firing time of TTRT. This is actually an approximation. In reality the system's timer is always loaded with the TTRT value and expires in a constant time interval. The use of an exponential transition, instead of a deterministic one, was imposed by the conditions stated previously, which should hold for the DSPN model to have an analytical solution. In fact, since a state that allows both the stations' timers to run concurrently (a marking where both the timer transitions will be enabled) will always exist, it is impossible to solve the problem analytically (with current Petri-net theory), without introducing such approximations.

The number of tokens in the place which is called timer-off, counts the number of times that the TTRT has expired between two consecutive captures of the FDDI (protocol) token by the station. Here, by approximation, we limit the number of possible TTRT expirations to 2. Moreover, in the real system, each time that an FDDI token enters the station and finds the TTRT timer still on, before any other action, it resets it to each initial value of TTRT. Assuming though exponential timers, we abstain from such an action, relying of course on the memoryless property of the exponential distribution.

The transmission part consists of three places and two transitions, one exponential and the other deterministic. The exponential one (tx) counts for the actual transmission of the packet, which is assumed to have an exponentially distributed length. The de-

terministic transition (*frmoverh1*), represents the fixed time interval for the transmission of the frame overhead bits. The overhead introduced by the circulation of the cycles (see section 2.2.2) is considered by simply reducing the actual transmission rate of the medium. Moreover, the assignment of any channels to isochronous communications, has a similar effect. Actually, the bit rate is varied according to how much bandwidth is left for packet-switched data, after the allocation of the isochronous channels.

Tokens in place *st1-ready* represent packets, as it is mentioned previously, that have arrived from upper protocol layers to the FDDI Medium Access Control layer ready for transmission. To enter the transmission part of our model and to receive the requested service, packets have to wait for either transition *firstct1* or *nextct1* to be enabled. In other words, an FDDI token should arrive in the place *token -arr1* and moreover the timer should not have expired (no token in place *timer-off*). Whenever the latter two conditions are fulfilled, packets (tokens) from place *st1-ready*, enter the place *begtx1* and proceed with their service. After the completion of a packet's service (token in place *endtx1*), several other conditions are checked in order to determine if the next packet in place *st1-ready* will be served or the service will be granted to the second station. If there is no other packet in place *st1-ready*, transition *nonextct1* will fire forward the token to place *nextst1*. The same result is accomplished by the expiring of the timer (token in place *timer1-off*. Finally, if none of the previous conditions holds, the next packet starts service by enabling the transition *nextct1*. Tokens coming to the place *nextst1* represent the FDDI token that is ready to be transferred to the other station. Transition *propag2* counts for the token's transmission time and the propagation delay between the two stations. The stations are considered as being diametrically opposite each other.

Considering the previously given parameter values used in our simulation model, we can derive very easily the firing rates of each of the exponential transitions and the fixed firing times for each of the deterministic transitions used in our Petri-net model. Here

transmission rate	24228 cust/s
timer firing rate	100 cust/s
frame overhead transition's firing time	1.65μsec
propagation transition's firing time	26.625μsec

Table 3.3: Parameter values for the DSPN Petri-net model

they are presented in a table form in the table 3.3.

GSPN Model

Trying to obtain some performance measures from the previous model, we encountered some difficulties. The model was built on a SUN workstation, using the graphical editor of GreatSPN (GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets), a software package for the validation and solution of Petri-net models [CHIO87]. Actually the maximum permissible number of states was reached. Thus the transformation of the model to a GSPN model was considered. For this reason the deterministic transition for the propagation delay between the stations was replaced by an exponential one having the same mean value. Also the deterministic transition counting for the frame overhead transmission was omitted, and the frame overhead transmission time was included in the packet's transmission time. Because of this the transition rate of the exponential transition had to be slightly decreased. In Fig. 3.6 the GSPN model of the same protocol is given. Table 3.2.3 lists the transitions' firing rates used in this model.

transmission rate	23316 cust/s
timer firing rate	100 cust/s
propagation transition's firing rate	37558.7 cust/s

Table 3.4: Parameter values for the GSPN Petri-net model

The mean packet length is kept fixed to 4096 bits per packet. However, the arrival

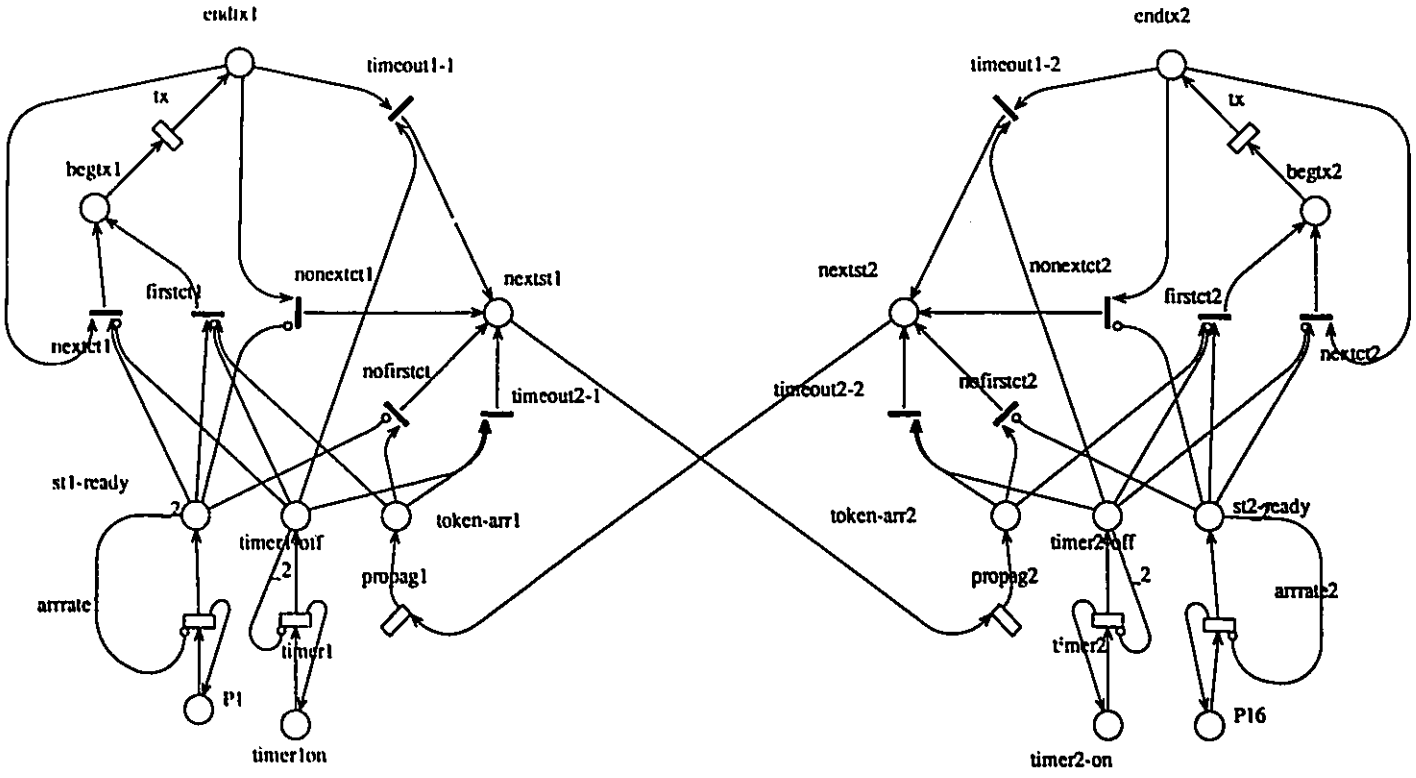

Figure 3.6: The GSPN model of the FDDI-II protocol.

rate of the packets has been varied. The value of k (the buffer space in each station), is kept as small as possible to minimize the state space of our model, but also big enough so as not to block the inflow of customers into the system.

Two performance measures are of interest to us: the throughput of the system and the total packet delay through the MAC and physical layers of the protocol. In this delay, the queueing delay at the station, the access time, the packet's transmission and propagation time are taken into consideration.

Denoting by λ the arrival rate of the packets at each station, and by N the mean number of customers in the places named st-ready, we can deduct the queueing delay, W_q, for each packet as follows.

$$W_q = \frac{N}{\lambda}.$$

To find the total delay (D) we must add to this time the transmission time t_{tr} (which can be easily computed knowing the average packet length and the transmission rate) and the propagation time t_p

$$D = W_q + t_{tr} + t_p.$$

Since we would like to compare the results from the GSFN Petri-nets model with the simulation results, the results of both the models are shown in Fig. 3.7.

As it can be seen, there is a discrepancy between the results of the two models. The magnitude of this difference remains steady as the throughput of the system increases. More specifically, the approximate GSPN Petri-net model overestimates the overall delay by approximately 17μsec. However, the relative error decreases, as expected, but not drastically, from 14% to 9% as the throughput increases.

Further research is needed to obtain the solution of the DSPN model and compare it with the results that have been already plotted in Fig. 3.7.

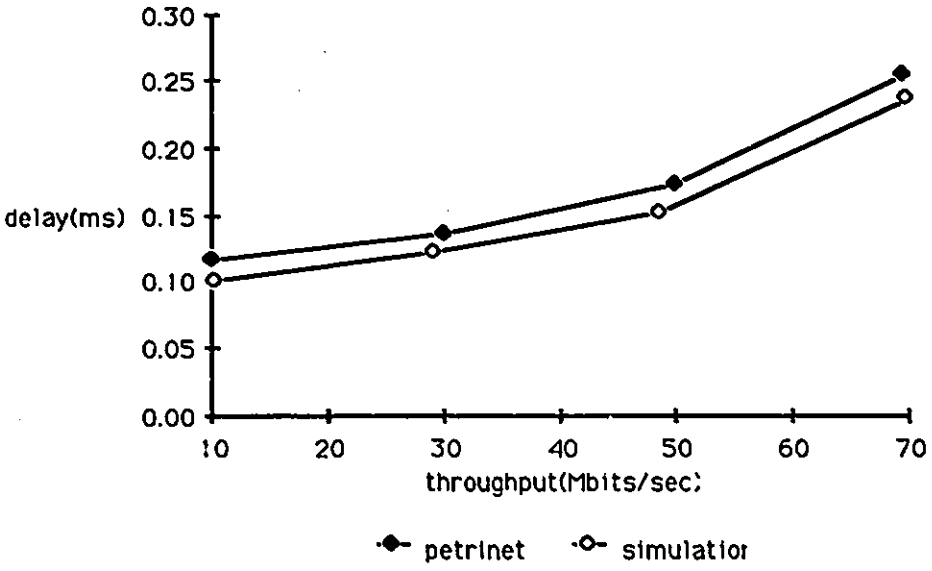

Figure 3.7: Comparison between the results of the simulation and the approximate GSPN Petri-net model results.

Chapter 4

Performance Evaluation of Image Filing Systems

In this chapter the performance evaluation of the prototype system proposed in chapter 1 is presented. The models of the protocols that have been explained in the previous chapters are employed here to derive performance measures.

However, a fundamental simplification is made. In the following we will deal only with the viewing of the radiographic images, simplifying the patient's folder to simple collections of image files. No voice or text are assumed coexisting with the image in any of the patients' folders. Therefore the image/text server becomes a simple file server with the radiographic images being the actual files.

The delay for an image display, together with the magnification display delay, constitute the main performance measures of our system. These delays can be defined as the time intervals between the issue of a display request by a user at a peripheral workstation, till the time that the requested item has completely appeared on his/her image screen of the workstation. Being the critical performance measures of our system, these delays should be kept as small as possible. Several factors affect them. The number of file servers and the way the images are distributed among them, the network's transmission rate, the existence or not of local storage at the workstations and the protocol overhead comprise the most important of them.

In the rest of the chapter, different configurations of the prototype system are compared and two different protocols are used. Computer simulation is the tool that has been used for obtaining the performance measures. More specifically, the simulation models of the two protocols presented in previous chapters (CSMA/CD and FDDI-II) are employed, to derive performance figures.

4.1 Image Filing System Using CSMA/CD

4.1.1 System's Configurations

Four configuration of the whole system are considered:

- *Centralized file server without local storage at the workstations.* This is the case where diskless workstations communicate with a single file server through a network. Image and magnification display requests, being short-length packets, travel through the network to the file server, and there they are queued for service on a first-come-first-served (FCFS) basis. The service time of these requests at the file server consists of the processing time for the request at the file server and the actual file access time. After the retrieval of the image or magnification data at the file server, it is being transmitted to the workstation through the network on a per packet basis.

- *Distributed file server without local storage at the workstations.* In this case instead of having just one file server, we have several. The sequence of events is exactly the same, except that the requests for image or magnification, can be served by different file servers. This configuration can alleviate a possible bottleneck at the file server, caused by the excessive service time at the server, for instance.

- *Centralized file server with local storage at the workstations.* Here we assume that the workstations are no more diskless, but they are capable of storing some data locally. Images are preloaded to the workstations' local storage, in a predefined

order, already known to the system, such that users at the workstations at a later time can retrieve them locally rather than from the remote file server. The image downloading process continues concurrently with the image viewing process at the workstation. In this way, images are retrieved from the local storage and viewed at the workstation, while image files concerning other patient cases, are stored at the workstation for later viewing. In this way, transmission delay is eliminated. This can be extremely advantageous, mainly in cases of networks of slow data-rate.

- *Distributed file server with local storage at the workstation.* Here again instead of one file server, several are used. The philosophy behind this change is the same as the one explained in the case with no local storage and several workstations. Otherwise there is no difference between this configuration and the previous one.

4.1.2 Modeling of Workstations and Servers

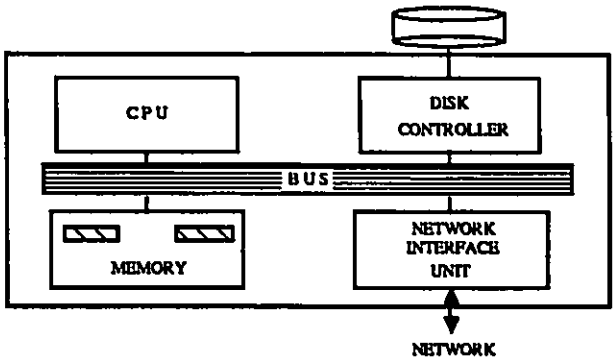

Figure 4.1: The PC-based file server's internal architecture

In Fig. 4.1 the PC-based file server is shown. It consists of a cpu, a memory, a network interface unit (NIU) and a disk access unit. The network interface unit performs all the protocol layers' functions. Whenever a packet is received, its data content is put into the memory. During transmission, data are taken from one of the two buffers, in the memory,

with DMA (Direct Memory Access) and segmented into smaller portions to form packets. Then, these packets are sent one by one to the network. The disk access unit is responsible for locating the requested image file on the disk and transferring it to the memory. The image is read to the memory block by block, and the available buffers in the memory are filled. If there is no empty buffer, the block transfer is disabled until the network interface unit empties one.

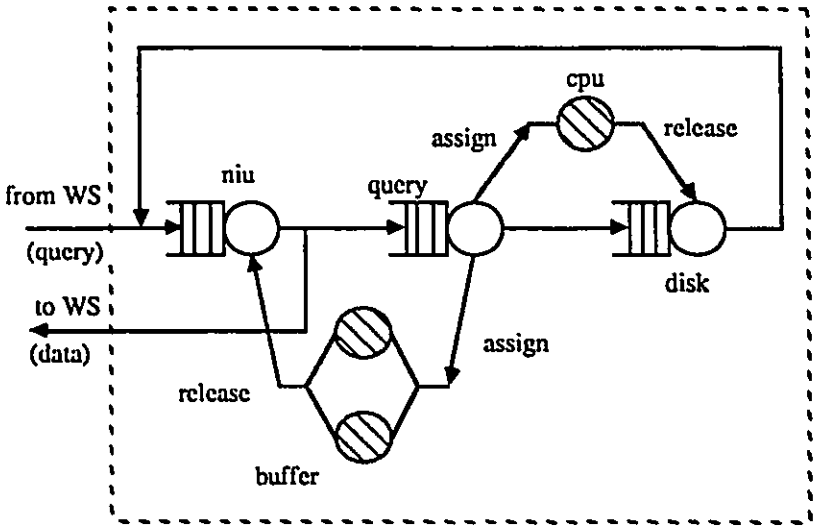

Figure 4.2: The file server's queueing model

The file server was modeled using a queueing network as it is shown in Fig. 4.2. Image and magnification requests enter the file server through the NIU station. There the stripping off of the various protocol overhead takes place. Subsequently the requests are queued for service at a fictitious queue called "query". Each image request creates 15 more requests, one for every block of image data. All these requests wait at the station queue for two other resources, namely buffer space in the main memory and the cpu of the PC. Once the resources are available, a block of image data can be retrieved from the disk. The time that is necessary for this comprises the service time of the station "disk".

After the completion of the block transfer, the processor is released, data is segmented into packets of smaller size and the necessary protocol overhead is added for transmission onto the network. This is done at the station NIU. Before the transmission of the last packet of a block the buffer space in the main memory is released. Here it is essential to point out that magnification requests do not undergo the same procedure for magnification data comprises only one block.

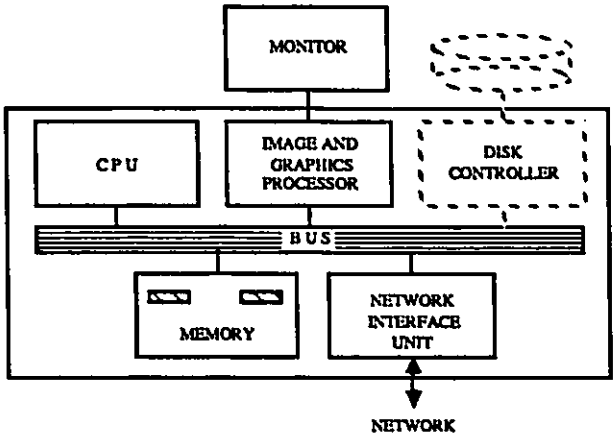

Figure 4.3: The PC-based workstation's internal architecture

In Fig. 4.3 the PC-based workstation is depicted. The configuration is the same except that there is an image and graphics processor which drives the workstation's monitor. The disk controller is taken into consideration only when we deal with workstations with local storage. Again image data is received on a packet by packet basis and then transferred with DMA to the available buffers, in the memory. When a buffer is full, its contents are transferred either to the screen, via the image and graphics processor, or to the local storage, if there is one. If the block that has been received though consists of magnification data, it is transferred directly for viewing at the screen in both cases. In the case of workstations with local storage, image data may be retrieved from the local storage (to be displayed) concurrently with the storing of image data coming from the file servers

to the local storage. Figure 4.4 gives the queueing network model of the workstation.

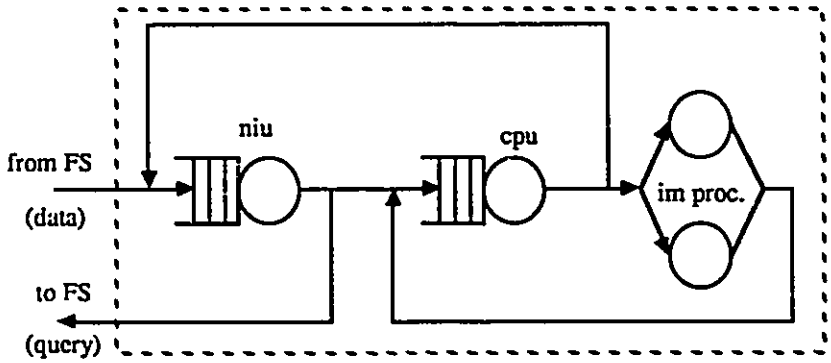

Figure 4.4: The workstation's queueing model

The NIU server performs the protocol processing for the outgoing requests and for the incoming data packets. After that, the blocks are transferred either to the screen, or to the disk using the server called "cpu". A delay center in the model represents the time required for the display of an image or magnification and for the examination of the image or magnification (user's thinking time). All (probabilistic) decisions regarding the request for the magnification and the request for the next image of the same case or for the first image of the next case are taken in this delay center as well.

4.1.3 Modeling Assumptions

The system is modeled by a closed queueing network as shown in Fig. 4.5. The boxes represent the models of the file server and the workstation, that have been presented in the previous section. The following assumptions are made.

- Retrieving data from the disk and storing data to the disk are performed in blocks of 64 Kbytes.
- The file servers and the workstations (if local storage is assigned) have infinite capacity of storage.

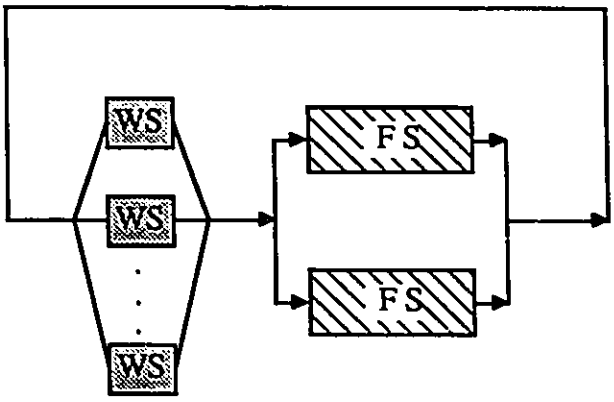

Figure 4.5: The System's Model

- The LAN access protocol is the well known CSMA/CD protocol. The Network Interface Unit performs all protocols for data transfers between the workstations and the file servers. The maximum length of data unit that is transferred between the memory and the Network Interface Unit is 64 Kbytes. The maximum packet size transmitted onto the LAN is 1 Kbyte. All the packets require the same processing time for packetizing and depacketizing in the Network Interface Unit .

- Images of 1024x1024 pixels and 2048x2048 pixels are stored in the image file servers in such a way, that we can retrieve the whole image or any particular part of the image from the file servers in the same way. The magnification of any particular part of an image always requires a transfer of 64 Kbyte data from the file server to the workstation.

- We study only the case of using the system in the emergency department where, in general, the number of x-ray images per case (patient's examination) varies between 2 to 6 uniformly. We assume an average of 4 images per case. In the model, this is represented by a random variable with a mean of 4 images per case.

- While working with any particular case, the radiologist or the attending physician may request to display any particular image more than once. The number of displayed images per case is also a random number from 4 to 10 with a mean of 5 displayed images per case.

- After the request for an image is issued and the image is displayed, the radiologist or the attending physician may spend, on the average, 5 seconds to decide whether or not he/she needs the magnification of any particular part of the image. Then, he/she will spend, on the average, 5 seconds to complete the examination of the present image. These thinking times are represented by a random variable with a mean of 5 seconds (10 seconds per image). We assume that 80% of the displayed images are followed by a magnification request.

- After the examination of a case is finished, a request is send to the file server to start loading another case to the local storage. In the mean time, the next image from the next patient's examination is fetched from the local storage (there is always an image ready to be viewed at the local storage).

- In the case of distributed file servers, balanced load to all servers is assumed. In other words, queries visit different file servers with equal probabilities. In this study, we assume only two file servers in the system. Queries for magnification have higher priority than queries for an entire image. Each query for an entire image requires the transfer of 16 blocks of 64 Kbytes data between the disk and the Network Interface Unit, via the two assigned buffers of 64 Kbytes in the memory, while each query for magnification requires only a transfer of 64 Kbytes data between the disk and the Network Interface Unit.

The time used to transfer a block of 64 Kbyte data between a buffer in the memory and the disk is assumed to be 400 ms. This is measured from the access time to a 20

Mbyte hard disk in an IBM PC-AT system with MS/DOS environment. The time for appending and stripping off the protocol overhead to form a packet is assumed to be 6 ms, result that was obtained by an experiment with the Sytek network adapter for the IBM PC Network.

The model has been built and solved by simulation, using the Queueing Network Analysis Package (QNAP2) [QNAP84]. For each simulation run, confidence intervals have been computed for a confidence level of 95%. This computation has been done by the spectral method of QNAP2. However, the obtained values, are very small to be perceived on the graphs. We have investigated two different data rates of LANs, 2 Mbits/s (Sytek 6000) and 10 Mbits/s (Ethernet) [TSIN87].

4.1.4 Results

Simulation results are given in Fig. 4.8 to 4.11. In the case that no local storage is assigned to the workstations, lower delays for both image and magnification are obtained in the case of distributed file servers than in the case of a centralized file server. However, as the number of workstations increases, the delay for magnification in both cases (distributed and centralized) is the same. It should be noted that lower delays are obtained from the LAN of higher data rate (10 Mbits/s).

The actual bottleneck in the configurations without local storage is the Network Interface Unit of the file server(s), because of the significant protocol processing time. The actual time for a packet to go through all the protocol layers, is greater than the packet's transmission time. Adding a second file server, we succeed in alleviating the bottleneck of the system for a small number of workstations. Increasing the number of workstations, however, we have reached again the point where the Network Interface Unit in the file servers becomes again the bottleneck. On the other hand, by increasing the network transmission rate, we manage to have lower transmission time for each packet and there-

fore increase the rate with which the Network Interface Unit processes the packets. It should be mentioned that since the transmission is performed on a per packet basis, the processing of a next packet does not begin unless the recent one has been transmitted completely (Fig. 4.6 and 4.7).

The same reasoning viewed from another point, can explain the results obtained for the waiting time for image magnification display. The fact that we have assigned higher priority to the magnification display requests, is the main reason for getting a constant delay as the number of workstations increases. Magnification display requests wait at the file server only for the transmission of image data being currently processed by the file server. As soon as a buffer in the memory is freed, it is instantaneously assigned to the first magnification display request waiting for service. Due to the fact that the buffer occupation time is relatively constant when the file server (and particularly the Network Interface Unit) is the bottleneck, the waiting time for the processing of magnification display requests remains constant.

In the case that an infinite capacity local storage is assigned to every workstation, the delay for displaying an image mainly depends on the access time to the local disk and is relatively constant for both centralized and distributed configurations. When the number of workstations is small, the waiting time for the display of an image is slightly higher than when the number of workstations is large. This is due to the contention for the disk access at the workstation between image display requests and blocks of image data coming from the file server. Image display requests need to retrieve image data from the local disk while blocks of image data coming from the file server, also require local disk access for storing new images to be examined. It should be noted that with a large number of workstations, the file server(s) has (have) to transfer images to a large number of other workstations after finishing the transmission towards a given workstation. As a result, as the number of workstations increases, the arrival rate of image data at each

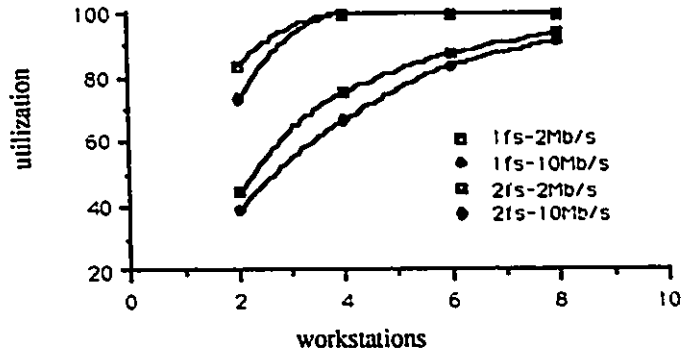

Figure 4.6: Utilization of the Network Interface Unit at the file server(no local storage)

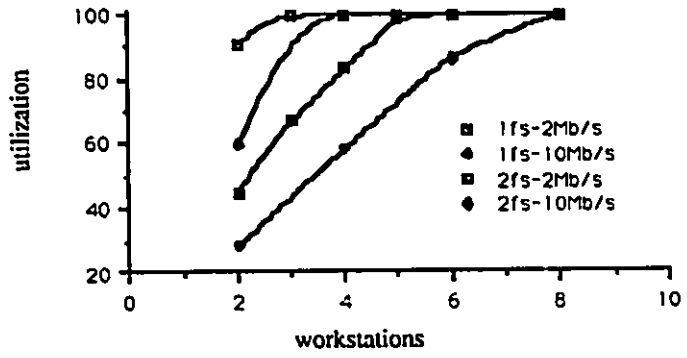

Figure 4.7: Utilization of the Network Interface Unit at the file server (with local storage)

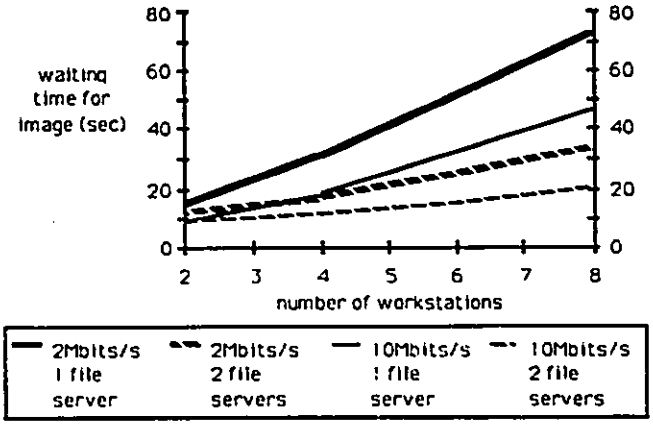

Figure 4.8: Waiting time for an image display (no local storage)

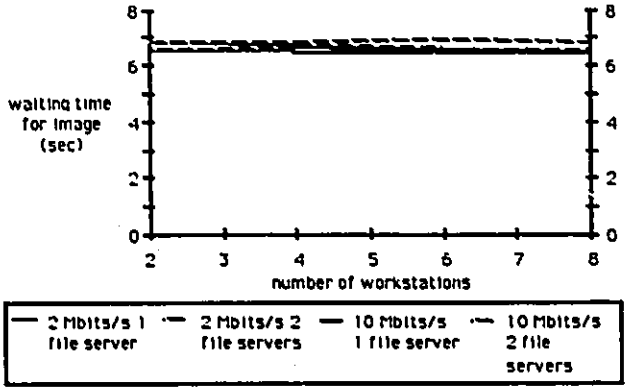

Figure 4.9: Waiting time for an image display (with local storage)

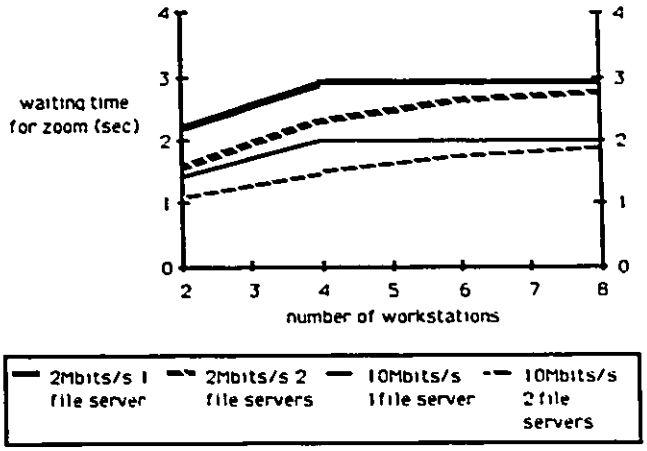

Figure 4.10: Waiting time for a magnification display (no local storage)

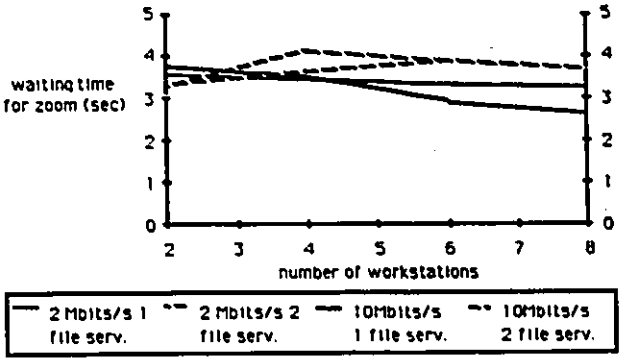

Figure 4.11: Waiting time for a magnification display (local storage)

workstation decreases and therefore there is less contention for local disk access.

The delay for magnification display consists of two parts, delay in the file server and delay in the workstation. The delay in the file server is actually the same as in the case of no local storage. However, when a local storage is assigned to the workstation, the (preloaded) image data coming from the file server may delay the processing of the magnification requests and magnification data at the workstation. We can see in Fig. 4.11 that as the number of workstations increases, which results in the decrease of the arrival rate of the preloaded image data of each workstation, the delay for magnification display will decrease to finally be the same as in the case of no local storage (Fig. 4.10). Therefore, the system configurations with local storage is more attractive for radiological image communications in the hospital. The delays for image and magnification displays may be reduced by using higher speed disks and disk controllers and by eliminating the protocol overhead in the Network Interface Unit for data transfers as discussed in [YATS85].

4.2 Image Filing System Using FDDI-II

4.2.1 System's Description

In this section the employment of an integrated services network in our system is discussed. The FDDI-II protocol discussed in extent previously is used. Image transfers are considered through its asynchronous and synchronous bandwidth (packet mode). The rest of the network's bandwidth is allocated to other applications, requiring isochronous transmission (e. g. voice).

The system consists as previously of a number of workstations which communicate through the FDDI-II network with a number of file servers. A Network Interface Unit is associated with each workstation and file server. As previously two cases are examined. The centralized file sever configuration and the distributed file server one. The difference between this study and the previous one for CSMA/CD is that we did not employ the

detailed models for the file servers and the workstations developed earlier in this chapter. Besides this, there is no local storage assigned to the workstations and all the requests have to be processed by the file server(s).

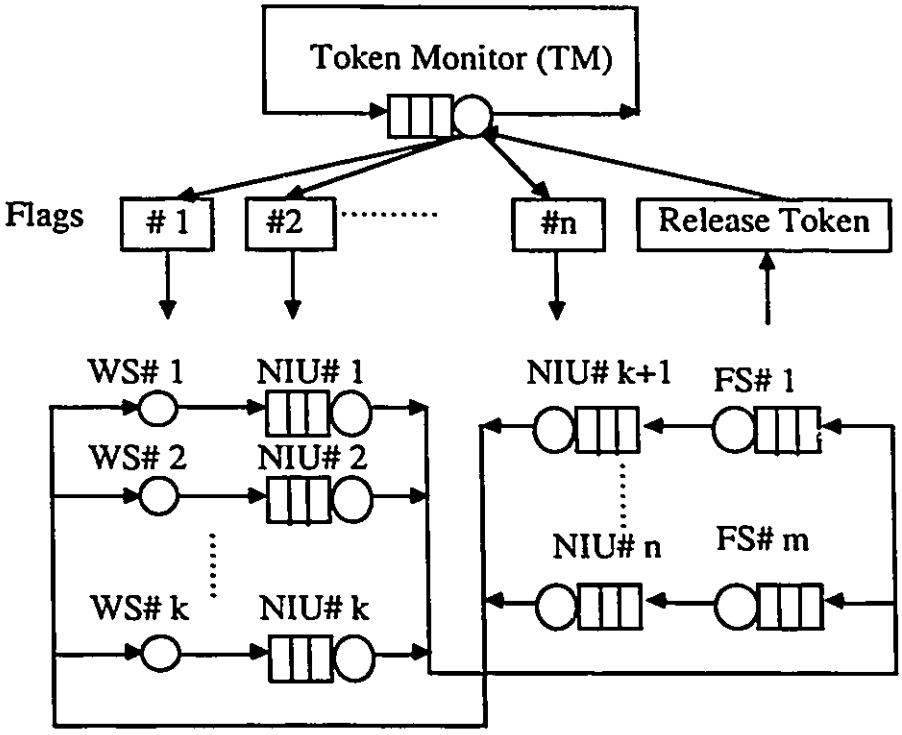

Figure 4.12: The system's model using the FDDI-II protocol

4.2.2 Modeling Assumptions

To model the system the already developed model for the FDDI-II protocol, shown in Fig. ??, is used. The difference is that the open queueing network model of chapter ?? is replaced by a closed one as shown in Fig. 4.12. The model is composed of a token passing model and the workstation-and-file-server model. The token passing model represents the circulation of the token around the ring as already explained. The workstation-and-file-server model consists of a number of workstations and file servers interconnected to

each other by an FDDI-II network via the Network Interface Units. The file servers and workstations are now represented as simple queues and their detailed representation was not taken into consideration in this study. The same software package (QNAP2) was used for computer simulation.

The following assumptions are made:

- For FDDI-II network parameters, the TTRT is assumed to be 4 ms and the length of the network is confined at 10 km. In this study, we assume that the overrun is permitted. During the transmission of a packet, if the TTRT expires, the station is permitted to continue the transmission of the packet and release the token afterwards. Compensation for that "lateness" of the token at the next stations is provided by prohibition of transmission at any station for the next circulations of the token around the ring until this accumulated "lateness" disappears. After each visit of the token at each station the counter associated with that "lateness" is decreased by the TTRT value. When this counter crosses the zero level the station is permitted to initiate a new transmission. This strategy supports the fairness at the bandwidth sharing among the stations.

- The propagation delay in the ring is 5μs/km. The station latency delay is assumed to be 5 ms. In consistency with Green [GREE87] we assume a packet length of 4096 bytes. Therefore, an image will be transmitted in 256 packets, while the magnification of a particular part of an image will require the transmission of 16 packets.

- At the file server, image and magnification data are retrieved from the storage and segmented into packets for transmission onto the network. The time required for data retrieval and packetization at the file server is assumed to be 25 ms/packet. This is the time that was measured by experimenting with our system and is in

conformance with the previously developed detailed model for the file server. After packetization, the packet will be sent to the network interface unit to be transmitted. We assume that the network interface unit has infinite storage for packets.

- The user at the workstation after receiving the entire image will spend in average 5 seconds before issuing a request either for the magnification or for another image. This "thinking" time is assumed to be exponentially distributed. The same thinking delay is applied to the case of the magnification. However, it is assumed that after examining the magnified part of an image, the user will issue only an image request. In other words, to examine an image, the user may issue at most one magnification request. The probability that the user requires a magnification for a particular part of an image is assumed to be 0.8, while the magnification request is deterministically followed by an image request.

- In the case of the distributed file servers, the display requests are addressed to the file servers with equal probability. That means that an image display request and the possible consecutive magnification display request from the same workstation, possibly concerning the same image, can be served by different file servers.

Two cases are examined. In the first case, we assume that 99% of the bandwidth provided by FDDI-II is allocated for isochronous communications (which is the maximum). Therefore, there is only a synchronous and asynchronous bandwidth of 1 Mb/s for image transfers. In the other case, a synchronous and asynchronous bandwidth of 10 Mb/s is available for image transfers. Furthermore, we assume that the network does not support any synchronous traffic.

4.2.3 Results

The results are given in Fig. 4.14 to 4.17 for both image and magnification display delays.

The most significant traffic in the network is the transfers of image and magnification data from the file servers to the workstations. Since image and magnification data are transferred in packets from the file server, the minimum delay (D) for transferring an image or a block of magnification data from a file server to a workstation can be expressed by

$$D_{min} = NT \tag{4.1}$$

where N is the number of packets for an image or a block of magnification data, and T is the time used to transmit a packet from the file server.

The time used to transmit a packet depends on the ring bit rate (R bits/s) and the packetization time (t_p). In fact, once the Network Interface Unit receives from the file server a packet of length l_p bits ready to be transmitted, it will require t_t seconds to complete the transmission of that packet, which can be obtained by

$$t_t = t_w + l_p/R \tag{4.2}$$

where t_w is the overhead due to the token passing protocol (waiting for the token) and depends mainly on the number of file servers that are actually providing load to the network.

In the case that $t_p \geq t_t$, the delays for image and magnification data transfers will depend mainly on t_p. In the contrary, if $t_t \geq t_p$ then the delays for image and magnification data transfers will depend on t_t. Therefore,

$$D_{min} = N(Max(t_p, t_t)) \tag{4.3}$$

In the first case, the file server will be the bottleneck of the system (Fig. 4.13, while in the second case, the Network Interface Unit of the file server will be the bottleneck of the system. In our system, if all 16 isochronous channels of the FDDI-II are assigned to the isochronous communications, then the Network Interface Unit will be the bottleneck

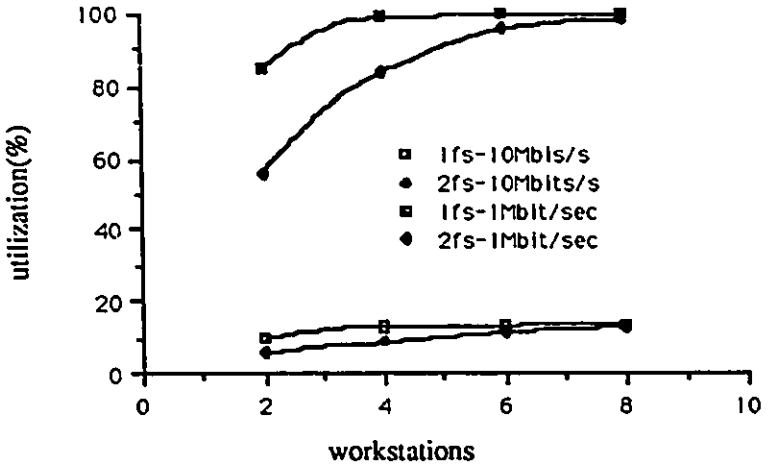

Figure 4.13: Utilization of the Network Interface Unit at the file server

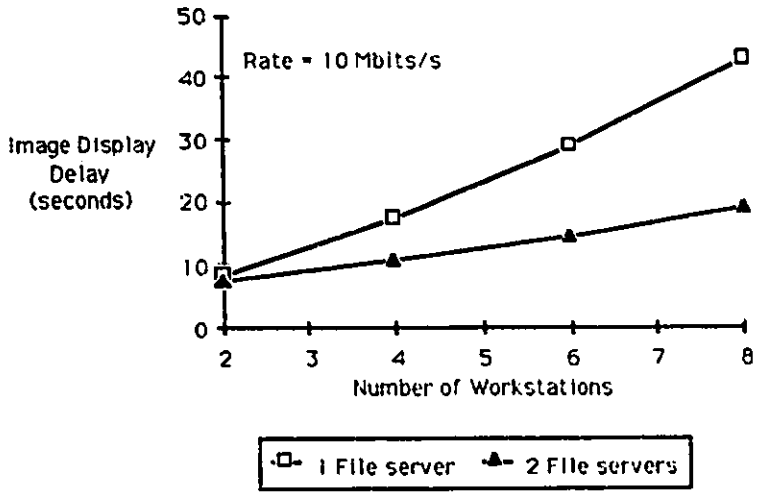

Figure 4.14: Waiting time for an image display (10 Mbits/s)

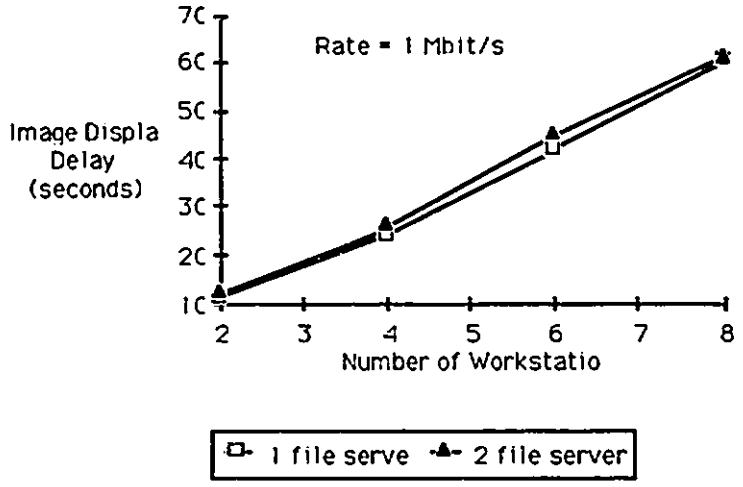

Figure 4.15: Waiting time for an image display (1 Mbit/s)

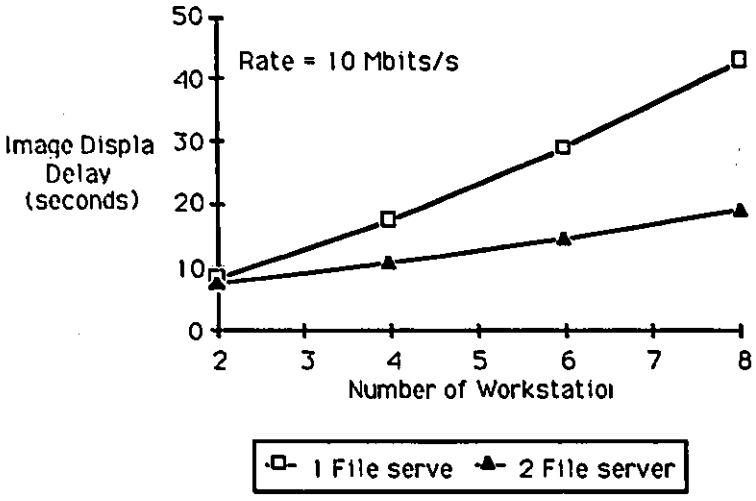

Figure 4.16: Waiting time for a magnification display (10 Mbits/s)

of the system and the image display delay will increase gradually upon the increase of the number of workstations and file servers (Fig. 4.15). On the other hand, if the Network Interface Unit is not the bottleneck but the file server then the delay may be reduced by using distributed file servers as shown in Fig. 4.15.

With regard to the magnification display delay, it is relatively constant compared to the image display delay as shown in Fig. 4.17 and 4.16. This is due to the fact that we have assigned higher priority to magnification data transfers and therefore there is no waiting time for the magnification request in the file server. The delays in the case of high available bandwidth (10 Mb/s) are slightly lower than the case of low bandwidth (1 Mb/s). This is due to the fact of assigning preemptive priorities to the magnification requests at the file server. Thus in the case of high bandwidth the processing of the magnification display requests is not confined by the existing bottleneck at the file server. Whereas, at the low bandwidth case, magnification display requests can get through the file server very easily and be delayed at the Network Interface Unit of the file server for the transmission of a preceding image packet. The situation deteriorates when we add more file servers at the network, mostly because the waiting time at the Network Interface Unit is increased by the contention for the network [TSINS8].

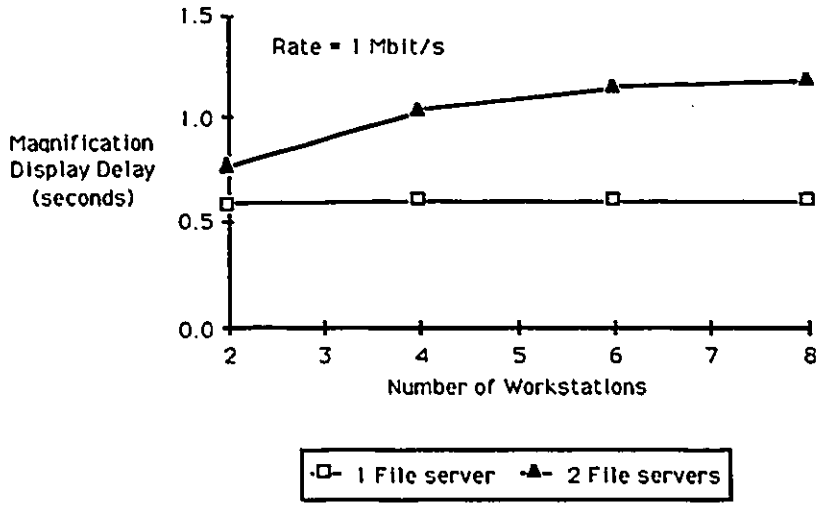

Figure 4.17: Waiting time for a magnification display (1 Mbit/s)

Chapter 5

Conclusions

Multimedia document architectures and multimedia communications systems are currently two very active areas in the research field of presentation, handling and dissemination of information. In chapter 2 we reviewed some existing Local Area Network protocols and we showed the evolution towards the integration of media handling presenting the state of the art Metropolitan Area Networks, FDDI-II and DQDB. Progress in technology enables us to harness the high bandwidth available in fiber optics with efficient protocol solutions such as FDDI-II and DQDB. A media integration is accomplished in a Medium Access Control layer level within a fairly large geographical area.

In chapter 3 trying to make a new contribution, the construction of an analytical model of FDDI-II using Petri-nets has been presented. Further research is needed in order to extent the solution techniques for a particular class of Petri-nets that is used, namely the DSPNs. This would facilitate the solution of our problem by permitting us to introduce fewer approximation assumptions. We feel that this fact will reduce the discrepancy between our analytical and simulation results.

As far as the main subject of this thesis is concerned, namely getting performance estimates of an image filing system, for use at the Radiology department within a hospital, two protocols (CSMA/CD and FDDI-II) and two main configurations were used (centralized and distributed file server). Generally speaking a distributed file server combined

with higher speed networks is a more advantageous configuration for image transfers. However, the total image display time is unacceptable for our application even in this favourable case. The use of faster file servers is advised, or otherwise the employment of a local storage is recommended. In the latter case, images should be downloaded to the local storage, off-line.

Further research is needed to take into consideration possible image compression and decompression techniques and find the impact they have on the total image display delay.

Appendix A

Program Listings

A.1 Simulation Program of the Ethernet Protocol

```
&                ETHERNET protocol
          attempt:=0;
&                CARRIER SENSING
          WAIT(net);
          attempt:=attempt+1;
&                CHANNEL ALLOCATION
          CST(t_prg);
          br:=br+1;
          RESET(net);
&                COLLISION DETECTION
          WHILE br>1 DO
          BEGIN
           COLLI:=COLLI+1;
           CST(t_jam);
&                BACKOFF CALCULATION
           IF attempt=1 THEN backoff:=2
                        ELSE backoff:=backoff*2;
           E:=RINT(0,backoff-1);
           br:=br-1;
           IF br=0 THEN SET(net);
&                  BACKOFF
           IF attempt>15 THEN BEGIN
&                  DISCARD INACTIVE
                            discard:=discard+1;
                            CST(t_slot*RINT(0,1023));
                            WAIT(net);
                            attempt:=attempt+1;
                            CST(t_prg);
                            br:=br+1;
                            RESET(net);
                            CST(t_prg);
                           END
                      ELSE
                      IF attempt>10 THEN
                      BEGIN
                       CST(t_slot*RINT(0,1023));
                       WAIT(net);
                       attempt:=attempt+1;
                       CST(t_prg);
                       br:=br+1;
                       RESET(net);
                       CST(t_prg);
                      END
                                 ELSE
                               BEGIN
                                CST(t_slot*E);
                                WAIT(net);
                                attempt:=attempt+1;
                                CST(t_prg);
                                br:=br+1;
                                RESET(net);
                                CST(t_prg);
                               END;
           END;
&                  NO COLLISION
&                  END OF TRANSMISSION
           CST(t_tmx-2*t_prg);
           br:=br-1;
           CST(t_del);
           SET(net);
```

A.2 Simulation Program of the FDDI-II Protocol

```
$run qnap02
$DECK
/CONTROL/OPTION=NSOURCE,NRESULT;

& this program simulates the fiber distributed data interface (FDDI)
& connecting several stations.

/DECLARE/INTEGER nb_st=2,              & number of stations connected
                 i,m,
                 idle_st,              & monitors the idleness of the ring
                 nb_mes,bound,
                 pack_l;               & average packet length in bits
         REAL lamda,                   & arrival rate of asynchronous data
              ttrt,                    & target token return time
              f,                       & % of bandwith allocated to voice.
              token_l,                 & token length in bits
              t_prg,                   & phys. ring latency+stat. latency
              frm_ovh,                 & frame overhead in bits
              tht,
              t_synchr,
              length,
              del,cdel,
              thr,cthr,
              ld,
              simtime,
              timer(nb_st),            & marker of the token arrival
              capac;                   & capacity of the network in bits/s
         QUEUE INTEGER is;
         QUEUE buffer(nb_st),
               pack_gen(nb_st),
               station(nb_st),
               poll;
         FLAG tx_perm(nb_st),
              end_sttx;
&        ------------------
         $MACRO give_prm
          RESET(tx_perm(is));
          SET(end_sttx);
          nb_mes:=1;
          $END
&         -----------------
/STATION/NAME=buffer;
         TYPE=SEMAPHORE,MULTIPLE(bound);

/STATION/NAME=pack_gen;
         TYPE=SOURCE;
         SERVICE=BEGIN
                   P(buffer(is));
                   EXP(1./lamda);
                   IF poll.NB=0 THEN BEGIN
&  creation of a new token
                                       m:=RINT(1,nb_st);
                                       idle_st:=0;
                                       timer:=TIME REPEAT nb_st;
                                       TRANSIT(NEW(CUSTOMER),poll);
                                     END;
                 END;
         TRANSIT=station(is);

/STATION/NAME=station;
```

```
        SERVICE=BEGIN
                  WAIT(tx_perm(is));
&  actions of the first customer after a polling
                  WHILE nb_mes=1 DO
                  BEGIN
                    nb_mes:=nb_mes+1;
                    tht:=ttrt-(TIME-timer(is));
                    timer(is):=TIME;
                    CST(t_synchr);
&  no available time for asynchronous packets transmission
                    IF tht<=0 THEN BEGIN
                                     timer(is):=timer(is)+tht;
                                     $give_prm
                                     WAIT(tx_perm(is));
                                   END;
                    END;
&  packet transmission
                    EXP(pack_l/((1-f)*capac));
                    CST(frm_ovh/((1-f)*capac));
                    V(buffer(is));
                    tht:=tht-(TIME-timer(is));
&  there is time for a second transmission
                    IF tht>0 THEN BEGIN
                                    IF station(is).NB>1 THEN TRANSIT(OUT)
                                                        ELSE BEGIN
                                                               $give_prm
                                                               TRANSIT(OUT);
                                                             END;
                                  END
&  no time for another transmission
                               ELSE BEGIN
                                      $give_prm
                                      TRANSIT(OUT);
                                    END;
                  END;

/STATION/NAME=poll;
         INIT=1;
         SERVICE=BEGIN
                   IF station(m).NB>0 THEN
                   BEGIN
                     SET(tx_perm(m));
                     WAIT(end_sttx);
                   END
                                    ELSE BEGIN
                                           idle_st:=idle_st+1;
&  token abortion after an idle circulation
                                           IF idle_st>=nb_st THEN TRANSIT(OUT);
                                           timer(m):=TIME;
                                           CST(t_synchr);
                                         END;
                   CST(token_l/((1-f)*capac)+t_prg/nb_st);
                   m:=m+1-INTREAL(m/nb_st)*nb_st;
                   RESET(end_sttx);
                   TRANSIT(poll);
                 END;

/CONTROL/TMAX=simtime;PERIOD=simtime/10.0;
         CLASS=station;
         ACCURACY=station;
```

```
        TEST=BEGIN
        IF (CRESPONSE(station(1))>1.0E-6)AND(CTHRUPUT(station(1))>1.0E-6) THEN
        IF CRESPONSE(station(1))<0.1*MRESPONSE(station(1)) THEN STOP;
            END;
        ENTRY=BEGIN
                PRINT("      INPUT PARAMETERS");
                PRINT("number of workstations          ",nb_st);
                PRINT("mean packet length for asynchr  ",pack_l,"bits");
                PRINT("isochronous bandwidth allocation",f*100,"%");
                PRINT("target token return time        ",ttrt*1000,"msec");
                PRINT("total cable length              ",length,"KM");
                PRINT("turn around propagation time    ",t_prg*1000,"msec");
              END;
        EXIT=BEGIN
               PRINT("               RESULTS");
               PRINT("simulation time                 ",TIME,"sec");
               FOR i:=1 STEP 1 UNTIL nb_st DO
               BEGIN
                 del:=del+MRESPONSE(station(i));
                 cdel:=cdel+CRESPONSE(station(i));
                 thr:=thr+MTHRUPUT(station(i));
                 cthr:=cthr+CTHRUPUT(station(i));
                 ld:=ld+MTHRUPUT(pack_gen(i));
               END;
               PRINT("total offered load              ",ld*pack_l*1.0E-6,
               "Mbits/sec");
               PRINT("total throughput                ",thr*pack_l*1.0E-6,
               "+/-",cthr/nb_st*1.0E-6,"Mbits/sec");
               PRINT("average packet delay            ",
               (del+t_prg)/REALINT(nb_st),"+/-",cdel/nb_st,"sec");
             END;

/EXEC/BEGIN
        FOR i:=1 STEP 1 UNTIL nb_st DO
        BEGIN
          station(i).is:=i;
          pack_gen(i).is:=i;
        END;
        m:=1;
        t_synchr:=0.0;
        nb_mes:=1;
        bound:=200;
        frm_ovh:=164;
        capac:=1.0E8;
        pack_l:=4096;
        token_l:=24;
        ttrt:=10.0E-3;
        f:=0.00672;
        length:=10;
        t_prg:=0.5085E-5*length+nb_st*0.6E-6;
        FOR i:=7,10,15 DO
        BEGIN
          lamda:=1220.8*i;
          simtime:=6000.0/lamda;
          del:=0;cdel:=0;thr:=0;cthr:=0;ld:=0;
          SIMUL;
        END;
      END;

/END/
```

A.3 Simulation Programs for Image Transfers Using CSMA/CD

A.3.1 a) Centralized File Server without Local Storage

```
$ run qnap02
$DECK
&
&                  file     :     C E N F S . Q
&
& This program simulates the behaviour of workstations connected through
& a LAN with a CENtralized image File Server.NO local storage is assumed.

/CONTROL/OPTION=NSOURCE;
         CLASS=ALL QUEUE;
         NMAX=35;

/DECLARE/INTEGER i,m,nb_ws=2,
                 nb_pck,                & number of packets that consist a block
                 br=0,                  & global variable
                 COLLI,                 & number of collisions
                 discard;               & number of potentially discarded packs
         CLASS INTEGER ic;
         QUEUE INTEGER is;
         QUEUE init,
               ws(nb_ws),
               niuws(nb_ws),
               niu,
               query,
               buffer,
               cpu,
               disk,
               imes(nb_ws),zmes(nb_ws); & for getting performance measures
         CLASS im(nb_ws),
               dim(nb_ws),
               zm(nb_ws),
               dzm(nb_ws);
         REAL
           t_slot,
           t_jam,
           t_del,
           t_prg,          & propagation time of the packet thru the net
           t_pclin,
           t_pclout,
           t_tmx,        & transmission time of a packet to the net
           t_disk,
           t_cpu,
           t_thk,           & thinking time at the workstation
           dlim,dlzm,
           simtime,
           capac;
         CUSTOMER INTEGER attempt,backoff,nb_blk,E;
         FLAG net;
         BOOLEAN luck;

/STATION/NAME=init;
         INIT(im(1))=1;
         SERVICE=BEGIN
                  SET(net);
                  TRANSIT(OUT);
                 END;

/STATION/NAME=zmes;
        TYPE=RESOURCE;
```

```
/STATION/NAME=imes;
        TYPE=RESOURCE;

$MACRO ethernet
&                       ETHERNET protocol
             attempt:=0;
&                       CARRIER SENSING
             WAIT(net);
             attempt:=attempt+1;
&                       CHANNEL ALLOCATION
             CST(t_prg);
             br:=br+1;
             RESET(net);
&                       COLLISION DETECTION
             WHILE br>1 DO
             BEGIN
              COLLI:=COLLI+1;
              CST(t_jam);
&                        BACKOFF CALCULATION
              IF attempt=1 THEN backoff:=2
                           ELSE backoff:=backoff*2;
              E:=RINT(0,backoff-1);
              br:=br-1;
              IF br=0 THEN SET(net);
&                        BACKOFF
              IF attempt>15 THEN BEGIN
&                        DISCARD INACTIVE
                                 discard:=discard+1;
                                 CST(t_slot*RINT(0,1023));
                                 WAIT(net);
                                 attempt:=attempt+1;
                                 CST(t_prg);
                                 br:=br+1;
                                 RESET(net);
                                 CST(t_prg);
                                END
                            ELSE
                            IF attempt>10 THEN
                            BEGIN
                             CST(t_slot*RINT(0,1023));
                             WAIT(net);
                             attempt:=attempt+1;
                             CST(t_prg);
                             br:=br+1;
                             RESET(net);
                             CST(t_prg);
                            END
                                       ELSE
                                  BEGIN
                                   CST(t_slot*E);
                                   WAIT(net);
                                   attempt:=attempt+1;
                                   CST(t_prg);
                                   br:=br+1;
                                   RESET(net);
                                   CST(t_prg);
                                  END;
              END;
&                        NO COLLISION
&                        END OF TRANSMISSION
```

```
            CST(t_tmx-2*t_prg);
            br:=br-1;
            CST(t_del);
            SET(net);
$END

/STATION/NAME=ws;
         INIT(zm)=IF ic=is THEN 1
                           ELSE 0;
         SERVICE(im)=BEGIN
                       EXP(t_thk/2);
                       luck:=DRAW(0.2);
                       IF luck=TRUE THEN TRANSIT(ws(is),zm(is))
                                    ELSE BEGIN
                                           P(zmes(is));
                                           TRANSIT(niuws(is),zm(is));
                                         END;
                     END;
         SERVICE(zm)=BEGIN
                       EXP(t_thk/2);
                       nb_blk:=16;
                       P(imes(is));
                       TRANSIT(niuws(is),im(is),0);
                     END;

/STATION/NAME=niuws;
         SERVICE(im,zm)=BEGIN
                          CST(t_pclout);
                          $ethernet
                        END;
         SERVICE(dzm)=BEGIN
                        CST(t_pclin);
                        V(zmes(ic));
                        TRANSIT(ws(ic),zm(ic));
                      END;
         SERVICE(dim)=BEGIN
                        CST(t_pclin);
                        IF nb_blk<>16 THEN TRANSIT(OUT)
                                      ELSE BEGIN
                                             V(imes(ic));
                                             TRANSIT(ws(ic),im(ic));
                                           END;
                      END;
        TRANSIT(im,zm)=niu;

/STATION/NAME=buffer;
         TYPE=RESOURCE,MULTIPLE(2);
         SCHED=FIFO,PRIOR;

/STATION/NAME=cpu;
        TYPE=RESOURCE;
         SCHED=FIFO,PRIOR;

/STATION/NAME=niu;
         SCHED=FIFO,PRIOR;
         PRIOR(zm,dzm)=1;
         SERVICE(zm)=BEGIN
                       CST(t_pclin);
                       TRANSIT(query);
                     END;
```

```
        SERVICE(im)=BEGIN
                      CST(t_pclin);
                      FOR i:=1 STEP 1 UNTIL  nb_blk-1 DO
                      TRANSIT(NEW(CUSTOMER),query);
                      TRANSIT(query);
                     END;
        SERVICE(dim,dzm)=BEGIN
                           FOR i:=1 STEP 1 UNTIL nb_pck-1 DO
                           BEGIN
                             CST(t_pclout);
                             $ethernet
                           END;
                           CST(t_pclout);
                           V(buffer);
                           $ethernet
                           TRANSIT(niuws(ic));
                         END;

/STATION/NAME=query;
        SCHED=FIFO,PRIOR;
        SERVICE(im,zm)=BEGIN
                         P(buffer);
                         P(cpu);
                         CST(t_cpu);
                         TRANSIT(disk);
                        END;

/STATION/NAME=disk;
        SCHED=FIFO,PRIOR;
        SERVICE(im,zm)=BEGIN
                         CST(t_disk);
                         CST(t_cpu);
                         V(cpu);
                        END;
        TRANSIT(zm)=niu,dzm(ic);
        TRANSIT(im)=niu,dim(ic);

/CONTROL/TMAX=simtime;PERIOD=simtime/20;TSTART=500;
        ACCURACY=imes,zmes,buffer,niu;
        TEST=BEGIN
               FOR i:=1 STEP 1 UNTIL nb_ws DO
               BEGIN
               IF (CRESPONSE(imes(i))>0.0001) AND
                  (CRESPONSE(zmes(i))>0.0001) THEN
                BEGIN
                 IF (CRESPONSE(imes(i))<0.1*MRESPONSE(imes(i))) AND
                    (CRESPONSE(zmes(i))<0.1*MRESPONSE(zmes(i))) THEN STOP;
                END;
               END;
             END;
        ENTRY=BEGIN
        PRINT("*                                                              *");
        PRINT("*          file     :     C E N F S . Q                        *");
        PRINT("*                           results                            *");
         PRINT("This program simulates the behaviour of ",nb_ws);
         PRINT("workstations connected through a LAN with a CENtralized");
         PRINT("image File Server.NO local storage is assumed.");
         PRINT(" ");
         PRINT("Data rate",capac,"Mbits/sec");
         PRINT(" ");
```

```
            END;
        EXIT=BEGIN
               FOR i:=1 STEP 1 UNTIL nb_ws DO
               BEGIN
               dlzm:=dlzm+MRESPONSE(zmes(i));
               dlim:=dlim+MRESPONSE(imes(i));
               PRINT("waiting time for an image display at the",i,
                     "workstation:",MRESPONSE(imes(i)));
               PRINT("waiting time for a  zoom display at the",i,
                     "workstation:",MRESPONSE(zmes(i)));
              END;
              PRINT(" ");
              PRINT("simulation time",TIME);
              PRINT("waiting time for an image display",dlim/nb_ws);
              PRINT("waiting time for a zoom display",dlzm/nb_ws);
              PRINT("number of collisions ",COLLI);
              PRINT("number of potentially discarded packets",discard);
            END;

/EXEC/BEGIN
        FOR i:=1 STEP 1 UNTIL nb_ws DO
                          BEGIN ws(i).is:=i;
                                dim(i).ic:=i;
                                im(i).ic:=i;
                                zm(i).ic:=i;
                                dzm(i).ic:=i;
                          END;
        t_thk:=10;
        t_disk:=0.4;
        t_cpu:=0.25E-4;
        t_prg:=0.5E-5;
        t_slot:=0.00512E-3;
        t_del:=0.96E-5;
        t_jam:=0.32E-5;
        t_pclin:=0.006;
        t_pclout:=0.006;
        nb_pck:=64;
        simtime:=10000;
        FOR m:=5 DO
        BEGIN
          capac:=2.0E+6*m;dlim:=0;dlzm:=0;
          t_tmx:=8192/capac;
          SIMUL;
        END;
      END;

/END/
$EOD
```

A.3.2 b) Distributed File Server without Local Storage

```
$ run qnap02
$DECK
&
&*                file     :       D I S F S . Q
&
& This program simulates the behaviour of workstations connected through
& a LAN with a DIStributed image File Server.NO local storage is assumed.

/CONTROL/OPTION=NSOURCE;
         CLASS=ALL QUEUE;
         NMAX=35;

/DECLARE/INTEGER i,nb_ws=2,
                 nb_fs=1,
                 nb_pck,                & number of packets that consist a block
                 br=0,                  & global variable
                 COLLI,                 & number of collisions
                 discard;               & number of potentially discarded packs
         CLASS INTEGER ic;
         QUEUE INTEGER is;
         QUEUE init,
               ws(nb_ws),
               niuws(nb_ws),
               niu(nb_fs),
               query(nb_fs),
               buffer(nb_fs),
               cpu(nb_fs),
               disk(nb_fs),
               imes(nb_ws),zmes(nb_ws); & for getting performance measures
         CLASS im(nb_ws),
               dim(nb_ws),
               zm(nb_ws),
               dzm(nb_ws);
         REAL t_slot,
              t_jam,
              t_del,
              t_prg,          & propagation time of the packet thru the net
              t_tmx,        & transmission time of a packet to the net
              t_thk,           & thinking time at the workstation
              t_pclin,
              t_pclout,
              t_disk,
              t_cpu,
              capac,
              simtime,
              dlzm,dlim;
         CUSTOMER INTEGER attempt,backoff,nb_blk,E;
         FLAG net;
         BOOLEAN luck;

/STATION/NAME=imes;
        TYPE=RESOURCE,SINGLE;

/STATION/NAME=zmes;
        TYPE=RESOURCE,SINGLE;

/STATION/NAME=init;
         INIT(im(1))=1;
         SERVICE=BEGIN
                   SET(net);
```

```
                    TRANSIT(OUT);
                  END;

$MACRO ethernet
&                       ETHERNET protocol
              attempt:=0;
&                       CARRIER SENSING
            WAIT(net);
            attempt:=attempt+1;
&                       CHANNEL ALLOCATION
            CST(t_prg);
            br:=br+1;
            RESET(net);
&                       COLLISION DETECTION
            CST(t_prg);
            WHILE br>1 DO BEGIN
            COLLI:=COLLI+1;
            CST(t_jam);
&                     BACKOFF CALCULATION
            IF attempt=1 THEN backoff:=2
                         ELSE backoff:=backoff*2;
            E:=RINT(0,backoff-1);
            br:=br-1;
            IF br=0 THEN SET(net);
&                       BACKOFF
            IF attempt>15 THEN BEGIN
&                       DISCARD INACTIVE
                              discard:=discard+1;
                              CST(t_slot*RINT(0,1023));
                              WAIT(net);
                              attempt:=attempt+1;
                              CST(t_prg);
                              br:=br+1;
                              RESET(net);
                              CST(t_prg);
                            END
                        ELSE
                        IF attempt>10 THEN
                        BEGIN
                          CST(t_slot*RINT(0,1023));
                          WAIT(net);
                          attempt:=attempt+1;
                          CST(t_prg);
                          br:=br+1;
                          RESET(net);
                          CST(t_prg);
                        END
                                       ELSE
                                     BEGIN
                                       CST(t_slot*E);
                                       WAIT(net);
                                       attempt:=attempt+1;
                                       CST(t_prg);
                                       br:=br+1;
                                       RESET(net);
                                       CST(t_prg);
                                     END;
                         END;
&                          NO COLLISION
&                          END OF TRANSMISSION
```

```
                        CST(t_tmx-2*t_prg);
                        br:=br-1;
                        CST(t_del);
                        SET(net);
$END

/STATION/NAME=ws;
        INIT(zm)=IF ic=is THEN 1
                          ELSE 0;
        SERVICE(im)=BEGIN
                     EXP(t_thk/2);
                     luck:=DRAW(0.2);
                     IF luck=TRUE THEN TRANSIT(ws(is),zm(is))
                                  ELSE BEGIN
                                         P(zmes(is));
                                         TRANSIT(niuws(is),zm(is));
                                       END;
                   END;
        SERVICE(zm)=BEGIN
                      EXP(t_thk/2);
                      nb_blk:=16;
                      P(imes(is));
                      TRANSIT(niuws(is),im(is),0);
                    END;

/STATION/NAME=niuws;
        SERVICE(im,zm)=BEGIN
                         CST(t_pclout);
                         $ethernet
                       END;
        SERVICE(dzm)=BEGIN
                       CST(t_pclin);
                       V(zmes(ic));
                       TRANSIT(ws(ic),zm(ic));
                     END;
        SERVICE(dim)=BEGIN
                       CST(t_pclin);
                       IF nb_blk<>16 THEN TRANSIT(OUT)
                                     ELSE BEGIN
                                            V(imes(ic));
                                            TRANSIT(ws(ic),im(ic));
                                          END;
                     END;
        TRANSIT(im,zm)=niu(1 STEP 1 UNTIL nb_fs),1.0/nb_fs REPEAT nb_fs-1;

/STATION/NAME=buffer;
        TYPE=RESOURCE,MULTIPLE(2);
        SCHED=FIFO,PRIOR;

/STATION/NAME=cpu;
       TYPE=RESOURCE;
        SCHED=FIFO,PRIOR;

/STATION/NAME=niu;
        SCHED=FIFO,PRIOR;
        PRIOR(zm,dzm)=1;
        SERVICE(zm)=BEGIN
                      CST(t_pclin);
                      TRANSIT(query(is));
                    END;
```

```
        SERVICE(im)=BEGIN
                      CST(t_pclin);
                      FOR i:=1 STEP 1 UNTIL  nb_blk-1 DO
                      TRANSIT(NEW(CUSTOMER),query(is));
                      TRANSIT(query(is));
                    END;
        SERVICE(dim,dzm)=BEGIN
                          FOR i:=1 STEP 1 UNTIL nb_pck-1 DO
                          BEGIN
                            CST(t_pclout);
                            $ethernet
                          END;
                          CST(t_pclout);
                          V(buffer(is));
                          $ethernet
                          TRANSIT(niuws(ic));
                        END;

/STATION/NAME=query;
        SCHED=FIFO,PRIOR;
        SERVICE(im,zm)=BEGIN
                         P(buffer(is));
                         P(cpu(is));
                         CST(t_cpu);
                         TRANSIT(disk(is));
                       END;

/STATION/NAME=disk;
        SCHED=FIFO,PRIOR;
        SERVICE(im,zm)=BEGIN
                         CST(t_disk);
                         CST(t_cpu);
                         V(cpu(is));
                       END;
        TRANSIT(zm)=niu(is),dzm(ic);
        TRANSIT(im)=niu(is),dim(ic);

/CONTROL/TMAX=simtime;PERIOD=simtime/100;TSTART=500;
        ACCURACY=imes,zmes,buffer,niu;
        TEST=BEGIN
              FOR i:=1 STEP 1 UNTIL nb_ws DO
              BEGIN
                IF (CRESPONSE(imes(i))>0.0001) AND
                (CRESPONSE(zmes(i))>0.0001) THEN
                 BEGIN
                  IF (CRESPONSE(imes(i))<0.1*MRESPONSE(imes(i))) AND
                     (CRESPONSE(zmes(i))<0.1*MRESPONSE(zmes(i))) THEN STOP;
                 END;
               END;
              END;
       ENTRY=BEGIN
        PRINT("*                                                              *");
        PRINT("*          file   :     D I S F S . Q                          *");
        PRINT("*                          results                             *");
        PRINT("This program simulates the behaviour of ",nb_ws);
        PRINT("workstations connected through a LAN with a");
        PRINT("DIStributed image file server.NO local storage is assumed.");
        PRINT(" ");
        PRINT("Data rate",capac,"Mbits/sec");
        PRINT(" ");
```

```
        PRINT("number of file servers :",nb_fs);
        PRINT(" ");
              END;
        EXIT=BEGIN
               FOR i:=1 STEP 1 UNTIL nb_ws DO
               BEGIN
                   dlzm:=dlzm+MRESPONSE(zmes(i));
                   dlim:=dlim+MRESPONSE(imes(i));
                 PRINT("waiting time for an image display at",i,
                       "workstation:",MRESPONSE(imes(i)));
                 PRINT("waiting time for a zoom display at",i,"workstation:",
                     MRESPONSE(zmes(i)));
              END;
                 PRINT("simulation time",TIME);
                 PRINT("waiting time for an image display ",dlim/nb_ws);
                 PRINT("waiting time for a zoom display ",dlzm/nb_ws);
                 PRINT("number of collisions :",COLLI);
                 PRINT("number of potentially discarded packets",discard);
           END;

/EXEC/BEGIN
       FOR i:=1 STEP 1 UNTIL nb_ws DO
                          BEGIN im(i).ic:=i;
                                ws(i).is:=i;
                                dim(i).ic:=i;
                                zm(i).ic:=i;
                                dzm(i).ic:=i;
                          END;
      FOR i:=1 STEP 1 UNTIL nb_fs DO BEGIN
                                niu(i).is:=i;
                                query(i).is:=i;
                                buffer(i).is:=i;
                                cpu(i).is:=i;
                                disk(i).is:=i;
                                    END;
       t_thk:=10;
       t_disk:=0.4;
       t_cpu:=0.25E-4;
       simtime:=10000;
       t_prg:=0.5E-5;
       t_slot:=0.00512E-3;
       t_del:=0.96E-5;
       t_jam:=0.32E-5;
       t_pclin:=0.006;
       t_pclout:=0.006;
       nb_pck:=64;
       simtime:=20000;
       capac:=2.0E+6;
       t_tmx:=8192/capac;
       SIMUL;
     END;

/END/
$EOD
```

A.3.3 c) Centralized File Server with Local Storage

```
$ run qnap02
$DECK
&*                                                                    *
&*              file     :     C E N F S L S . Q                      *
&*                                                                    *
& This program simulates the behaviour of workstations connected through
& a LAN with a CENtralized image File Server.LOCAL STORAGE is assumed.

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

/DECLARE/INTEGER i,nb_ws=3,
                 min_im=2,
                 max_im=6,
                 nb_pck,
                 br=0,
                 COLLI,
                 discard,
                 dsp_im(nb_ws),
                 req_im(nb_ws);
         CLASS INTEGER ic;
         QUEUE INTEGER is;
         QUEUE init,
               ws(nb_ws),
               cpuws(nb_ws),
               niuws(nb_ws),
               niu,
               query,
               buffer,
               cpu,
               disk,
               imwsmes(nb_ws),
               zmes(nb_ws),
               csmes(nb_ws);
         CLASS im(nb_ws),
               cs(nb_ws),
               imws(nb_ws),
               dim(nb_ws),
               zm(nb_ws),
               dzm(nb_ws),
               dcs(nb_ws);
         REAL t_slot,
              t_jam,
              t_del,
              t_prg,
              t_pclin,
              t_pclout,
              t_tmx,
              t_disk,
              t_cpu,
              t_thk,
              capac,
              simtime,
              dlzm,dlim,dlcs;
         CUSTOMER INTEGER attempt,backoff,nb_blk,E;
         FLAG net;
         BOOLEAN luck;

/STATION/NAME=init;
```

```
          INIT(im(1))=1;
          SERVICE=BEGIN
                    SET(net);
                    TRANSIT(OUT);
                  END;

/STATION/NAME=zmes;
         TYPE=RESOURCE;

/STATION/NAME=imwsmes;
         TYPE=RESOURCE;

/STATION/NAME=csmes;
         TYPE=RESOURCE,INFINITE;

$MACRO ethernet
&                       ETHERNET protocol
              attempt:=0;
            &CARRIER SENSING
             WAIT(net);
             attempt:=attempt+1;
            & CHANNEL ALLOCATION
             CST(t_prg);
             br:=br+1;
             RESET(net);
            & COLLISION DETECTION
             CST(t_prg);
               WHILE br>1 DO BEGIN
               CST(t_jam);
                  &BACKOFF CALCULATION
                       IF attempt=1 THEN backoff:=2
                                    ELSE backoff:=backoff*2;
                       E:=RINT(0,backoff-1);
                       br:=br-1;
                       IF br=0 THEN SET(net);
                       &BACKOFF
                       IF attempt>15 THEN BEGIN
                       &DISCARD INACTIVE
                                          CST(t_slot*RINT(0,1023));
                                          WAIT(net);
                                          attempt:=attempt+1;
                                          CST(t_prg);
                                          br:=br+1;
                                          RESET(net);
                                          CST(t_prg);
                                          END
                                     ELSE
                                     IF attempt>10 THEN
                                      BEGIN
                                          CST(t_slot*RINT(0,1023));
                                          WAIT(net);
                                          attempt:=attempt+1;
                                          CST(t_prg);
                                          br:=br+1;
                                          RESET(net);
                                          CST(t_prg);
                                      END
                                                    ELSE
                                      BEGIN
                                          CST(t_slot*E);
```

```
                                        WAIT(net);
                                        attempt:=attempt+1;
                                        CST(t_prg);
                                        br:=br+1;
                                        RESET(net);
                                        CST(t_prg);
                                    END;
                        END;
                        &NO COLLISION
                        & END OF TRANSMISSION
                          CST(t_tmx-2*t_prg);
                          br:=br-1;
                          CST(t_del);
                          SET(net);
$END

/STATION/NAME=ws;
        INIT(zm)=IF ic=is THEN 1
                          ELSE 0;
        SERVICE(zm)=BEGIN
                        EXP(t_thk/2);
                        dsp_im(is):=dsp_im(is)+1;
                        WHILE dsp_im(is)>=4 DO
                        BEGIN
                          luck:=DRAW(0.1*dsp_im(is));
                          IF luck=TRUE THEN
                          BEGIN
                            dsp_im(is):=0;
                            P(csmes(ic));
                            TRANSIT(NEW(CUSTOMER),cpuws(is),cs(is));
                          END
                                        ELSE
                          BEGIN
                            nb_blk:=16;
                            P(imwsmes(is));
                            FOR i:=1 STEP 1 UNTIL nb_blk-1 DO
                            TRANSIT(NEW(CUSTOMER),cpuws(is),imws(is));
                            TRANSIT(cpuws(is),imws(is));
                          END;
                        END;
                        nb_blk:=16;
                        P(imwsmes(is));
                        FOR i:=1 STEP 1 UNTIL nb_blk-1 DO
                        TRANSIT(NEW(CUSTOMER),cpuws(is),imws(is));
                        TRANSIT(cpuws(is),imws(is));
                      END;
        SERVICE(imws)=BEGIN
                         EXP(t_thk/2);
                         luck:=DRAW(0.2);
                         IF luck=TRUE THEN TRANSIT(ws(is),zm(is))
                                      ELSE BEGIN
                                             P(zmes(is));
                                             TRANSIT(cpuws(is),zm(is));
                                           END;
                      END;

/STATION/NAME=cpuws;
        SERVICE(imws)=BEGIN
                         CST(t_cpu);
                         CST(t_disk);
```

```
                    IF nb_blk<>16 THEN TRANSIT(OUT)
                                  ELSE BEGIN
                                         V(imwsmes(ic));
                                         TRANSIT(ws(ic));
                                       END;
                    END;
        SERVICE(zm,cs)=BEGIN
                         CST(t_cpu);
                         TRANSIT(niuws(ic));
                       END;
        SERVICE(dim)=BEGIN
                       CST(t_cpu);
                       CST(t_disk);
                       TRANSIT(OUT);
                     END;
        SERVICE(dcs)=BEGIN
                       CST(t_cpu);
                       CST(t_disk);
                       V(csmes(ic));
                       TRANSIT(OUT);
                     END;
        SERVICE(dzm)=BEGIN
                       CST(t_cpu);
                       V(zmes(ic));
                       TRANSIT(ws(ic),zm(ic));
                     END;

/STATION/NAME=niuws;
        SERVICE(dim,dzm,dcs)=CST(t_pclin);
        SERVICE(cs,zm)=BEGIN
                         CST(t_pclout);
                         $ethernet
                       END;
        TRANSIT(dim,dzm,dcs)=cpuws(ic);
        TRANSIT(cs,zm)=niu;

/STATION/NAME=buffer;
        TYPE=RESOURCE,MULTIPLE(2);
        SCHED=FIFO,PRIOR;

/STATION/NAME=cpu;
       TYPE=RESOURCE;
        SCHED=FIFO,PRIOR;

/STATION/NAME=niu;
        SCHED=FIFO,PRIOR;
        PRIOR(zm,dzm)=1;
        SERVICE(zm)=BEGIN
                      CST(t_pclin);
                      TRANSIT(query);
                    END;
        SERVICE(cs)=BEGIN
                      CST(t_pclin);
                      req_im(ic):=RINT(min_im,max_im);
                      FOR i:=1 STEP 1 UNTIL  req_im(ic)*nb_blk-1 DO
                      TRANSIT(NEW(CUSTOMER),query,im(ic));
                      TRANSIT(query);
                    END;
        SERVICE(dim,dcs)=BEGIN
                           FOR i:=1 STEP 1 UNTIL nb_pck-1 DO
```

```
                        BEGIN
                          CST(t_pclout);
                          $ethernet
                        END;
                        CST(t_pclout);
                        V(buffer);
                        $ethernet
                        TRANSIT(niuws(ic));
                      END;
        SERVICE(dzm)=BEGIN
                       FOR i:=1 STEP 1 UNTIL nb_pck-1 DO
                       BEGIN
                         CST(t_pclout);
                         $ethernet
                       END;
                       CST(t_pclout);
                       V(buffer);
                       $ethernet
                       TRANSIT(niuws(ic),0);
                     END;

/STATION/NAME=query;
         SCHED=FIFO,PRIOR;
         SERVICE(im,cs,zm)=BEGIN
                             P(buffer);
                             P(cpu);
                             CST(t_cpu);
                             TRANSIT(disk);
                           END;

/STATION/NAME=disk;
         SCHED=FIFO,PRIOR;
         SERVICE=BEGIN
                   CST(t_disk);
                   CST(t_cpu);
                   V(cpu);
                 END;
         TRANSIT(zm)=niu,dzm(ic);
         TRANSIT(cs)=niu,dcs(ic);
         TRANSIT(im)=niu,dim(ic);

/CONTROL/TMAX=simtime;
         TSTART=10;
         ACCURACY=zmes,imwsmes,buffer,niu;
         ENTRY=BEGIN
         PRINT("*                                                              *");
         PRINT("*        file    :    C E N F S L S . M U S                    *");
         PRINT("*                          results                             *");
          PRINT("This program simulates the behaviour of ",nb_ws);
          PRINT("workstations connected through a LAN with a");
          PRINT("centralized IMAGE FILE SERVER.LOCAL STORAGE assumed.");
          PRINT(" ");
          PRINT("Transmission rate",capac,"Mbits/sec");
          PRINT(" ");
               END;
         EXIT=BEGIN
                FOR i:=1 STEP 1 UNTIL nb_ws DO
                BEGIN
                 dlcs:=dlcs+MRESPONSE(csmes(i))+
                           MRESPONSE(cpuws(i),cs(i));
```

```
                dlim:=dlim+MRESPONSE(imwsmes(i));
                dlzm:=dlzm+MRESPONSE(zmes(i));
                PRINT("waiting time for an image display at",i,"workstation:",
                      MRESPONSE(imwsmes(i)));
                PRINT("waiting time for a zoom display at",i,"workstation:",
                      MRESPONSE(zmes(i)));
                PRINT("waiting time for a patient case at",i,"workstation:",
                      MRESPONSE(csmes(i))+MRESPONSE(cpuws(i),cs(i))-t_cpu);
              END;
              PRINT(" ");
              PRINT("waiting time for an image ",dlim/nb_ws);
              PRINT("waiting time for a zoom ",dlzm/nb_ws);
              PRINT("waiting time for a case ",(dlcs/nb_ws)-t_cpu);
              PRINT("number of collisions",COLLI);
              PRINT("number of potentially discarded packets",discard);
            END;

/EXEC/BEGIN
        FOR i:=1 STEP 1 UNTIL nb_ws DO
                            BEGIN
                              ws(i).is:=i;
                              dsp_im(i):=-1;
                              im(i).ic:=i;
                              dim(i).ic:=i;
                              imws(i).ic:=i;
                              zm(i).ic:=i;
                              dzm(i).ic:=i;
                              cs(i).ic:=i;
                              dcs(i).ic:=i;
                            END;
        t_thk:=10;              & thinking time at the workstation
        t_disk:=0.4;
        t_cpu:=0.25E-4;
        t_prg:=0.5E-5;          & propagation time of the packet thru the net
        t_slot:=0.00512E-3;
        t_del:=0.96E-5;
        t_jam:=0.32E-5;
        t_pclin:=0.006;
        t_pclout:=0.006;
        nb_pck:=64;             & number of packets per block
        capac:=2.0E+6;
        t_tmx:=8192/capac;
        simtime:=10000;
        SIMUL;
      END;

/END/
$EOD
```

A.3.4 d) Distributed File Server with Local Storage

```
$ run qnap02
$DECK
&*                                                                    *
&*            file     :       D I S F S L S . Q                     *
&*                                                                    *
& This program simulates the behaviour of workstations connected through
& a LAN with a DIStributed image File Server.LOCAL STORAGE is assumed.

/CONTROL/OPTION=NSOURCE;
         CLASS=ALL QUEUE;
         NMAX=35;

/DECLARE/INTEGER i,nb_ws=2,
                 nb_fs=1,
                 max_im=6,
                 min_im=2,
                 nb_pck,
                 br=0,
                 COLLI,
                 discard,
                 dsp_im(nb_ws),
                 req_im(nb_ws);
         CLASS INTEGER ic;
         QUEUE INTEGER is;
         QUEUE init,
               ws(nb_ws),
               cpuws(nb_ws),
               niuws(nb_ws),
               niu(nb_fs),
               query(nb_fs),
               buffer(nb_fs),
               cpu(nb_fs),
               disk(nb_fs),
               network,
               zmes(nb_ws),
               imwsmes(nb_ws),
               csmes(nb_ws);
         CLASS im(nb_ws),
               imws(nb_ws),
               dim(nb_ws),
               zm(nb_ws),
               dzm(nb_ws),
               cs(nb_ws),
               dcs(nb_ws);
         REAL t_slot,
              t_jam,
              t_del,
              t_prg,
              t_pclin,
              t_pclout,
              t_tmx,
              t_disk,
              t_cpu,
              t_thk,
              simtime,
              capac,
              dlzm,
              dlim,
              dlcs;
         CUSTOMER INTEGER attempt,backoff,nb_blk,E;
```

```
          FLAG net;
          BOOLEAN luck;

/STATION/NAME=init;
          INIT(im(1))=1;
          SERVICE=BEGIN
                    SET(net);
                    TRANSIT(OUT);
                  END;

/STATION/NAME=network;
         TYPE=RESOURCE;

/STATION/NAME=zmes;
         TYPE=RESOURCE;

/STATION/NAME=imwsmes;
         TYPE=RESOURCE;

/STATION/NAME=csmes;
         TYPE=RESOURCE,INFINITE;

$MACRO ethernet
&                       ETHERNET protocol
              attempt:=0;
            &CARRIER SENSING
             WAIT(net);
             attempt:=attempt+1;
            & CHANNEL ALLOCATION
             CST(t_prg);
             br:=br+1;
             RESET(net);
            & COLLISION DETECTION
             CST(t_prg);
               WHILE br>1 DO BEGIN
               COLLI:=COLLI+1;
               CST(t_jam);
                  &BACKOFF CALCULATION
                       IF attempt=1 THEN backoff:=2
                                    ELSE backoff:=backoff*2;
                       E:=RINT(0,backoff-1);
                       br:=br-1;
                       IF br=0 THEN SET(net);
                       &BACKOFF
                       IF attempt>15 THEN BEGIN
                       &DISCARD INACTIVE
                                          discard:=discard+1;
                                          CST(t_slot*RINT(0,1023));
                                          WAIT(net);
                                          attempt:=attempt+1;
                                          CST(t_prg);
                                          br:=br+1;
                                          RESET(net);
                                          CST(t_prg);
                                          END
                                     ELSE
                                     IF attempt>10 THEN
                                      BEGIN
                                          CST(t_slot*RINT(0,1023));
                                          WAIT(net);
```

```
                                    attempt:=attempt+1;
                                    CST(t_prg);
                                    br:=br+1;
                                    RESET(net);
                                    CST(t_prg);
                                END
                                            ELSE
                                BEGIN
                                    CST(t_slot*E);
                                    WAIT(net);
                                    attempt:=attempt+1;
                                    CST(t_prg);
                                    br:=br+1;
                                    RESET(net);
                                    CST(t_prg);
                                END;
                        END;
                        &NO COLLISION
                        & END OF TRANSMISSION
                          P(network);
                          CST(t_tmx-2*t_prg);
                          V(network);
                          br:=br-1;
                          CST(t_del);
                          SET(net);
$END

/STATION/NAME=ws;
        INIT(zm)=IF ic=is THEN 1
                          ELSE 0;
        SERVICE(zm)=BEGIN
                        EXP(t_thk/2);
                        dsp_im(is):=dsp_im(is)+1;
                        WHILE dsp_im(is)>=4 DO
                        BEGIN
                          luck:=DRAW(0.1*dsp_im(is));
                          IF luck=TRUE THEN
                          BEGIN
                            dsp_im(is):=0;
                            TRANSIT(NEW(CUSTOMER),cpuws(is),cs(is));
                          END
                                        ELSE
                          BEGIN
                            nb_blk:=16;
                            P(imwsmes(is));
                            FOR i:=1 STEP 1 UNTIL nb_blk-1 DO
                            TRANSIT(NEW(CUSTOMER),cpuws(is),imws(is));
                            TRANSIT(cpuws(is),imws(is));
                          END;
                        END;
                        nb_blk:=16;
                        P(imwsmes(is));
                        FOR i:=1 STEP 1 UNTIL nb_blk-1 DO
                        TRANSIT(NEW(CUSTOMER),cpuws(is),imws(is));
                        TRANSIT(cpuws(is),imws(is));
                    END;
        SERVICE(imws)=BEGIN
                        EXP(t_thk/2);
                        luck:=DRAW(0.2);
                        IF luck=TRUE THEN TRANSIT(ws(is),zm(is))
```

```
                                        ELSE BEGIN
                                               P(zmes(is));
                                               TRANSIT(cpuws(is),zm(is));
                                             END;
                        END;

/STATION/NAME=cpuws;
          SERVICE(imws)=BEGIN
                          CST(t_cpu);
                          CST(t_disk);
                          IF nb_blk<>16 THEN TRANSIT(OUT)
                                                ELSE BEGIN
                                                       V(imwsmes(ic));
                                                       TRANSIT(ws(ic));
                                                     END;
                        END;
          SERVICE(cs)=BEGIN
                        P(csmes(ic));
                        nb_blk:=16;
                        CST(t_cpu);
                        TRANSIT(niuws(ic));
                      END;
          SERVICE(zm)=BEGIN
                        CST(t_cpu);
                        TRANSIT(niuws(ic));
                      END;
          SERVICE(dim)=BEGIN
                        CST(t_cpu);
                        CST(t_disk);
                        TRANSIT(OUT);
                      END;
          SERVICE(dcs)=BEGIN
                         CST(t_cpu);
                         CST(t_disk);
                         V(csmes(ic));
                         TRANSIT(OUT);
                       END;
          SERVICE(dzm)=BEGIN
                         CST(t_cpu);
                         V(zmes(ic));
                         TRANSIT(ws(ic),zm(ic));
                       END;

/STATION/NAME=niuws;
          SERVICE(dim,dzm,dcs)=CST(t_pclin);
          SERVICE(cs,zm)=BEGIN
                           CST(t_pclout);
                           $ethernet
                         END;
         TRANSIT(dim,dzm,dcs)=cpuws(ic);
         TRANSIT(cs,zm)=niu(1 STEP 1 UNTIL nb_fs),1.0/nb_fs REPEAT nb_fs-1;

/STATION/NAME=buffer;
          TYPE=RESOURCE,MULTIPLE(2);
          SCHED=FIFO,PRIOR;

/STATION/NAME=cpu;
         TYPE=RESOURCE;
          SCHED=FIFO,PRIOR;
```

```
/STATION/NAME=niu;
        SCHED=FIFO,PRIOR;
        PRIOR(zm,dzm)=1;
        SERVICE(zm)=BEGIN
                      CST(t_pclin);
                      TRANSIT(query(is));
                    END;
        SERVICE(cs)=BEGIN
                     req_im(ic):=RINT(min_im,max_im);
                     CST(t_pclin);
                     FOR i:=1 STEP 1 UNTIL req_im(ic)*nb_blk-1 DO
                     TRANSIT(NEW(CUSTOMER),query(is),im(ic));
                     TRANSIT(query(is));
                   END;
        SERVICE(dim,dcs)=BEGIN
                           FOR i:=1 STEP 1 UNTIL nb_pck-1 DO
                           BEGIN
                             CST(t_pclout);
                             $ethernet
                           END;
                           CST(t_pclout);
                           V(buffer(is));
                           $ethernet
                           TRANSIT(niuws(ic));
                         END;
        SERVICE(dzm)=BEGIN
                       FOR i:=1 STEP 1 UNTIL nb_pck-1 DO
                       BEGIN
                         CST(t_pclout);
                         $ethernet
                       END;
                       CST(t_pclout);
                       V(buffer(is));
                       $ethernet
                       TRANSIT(niuws(ic),0);
                     END;

/STATION/NAME=query;
        SCHED=FIFO,PRIOR;
        SERVICE(im,cs,zm)=BEGIN
                            P(buffer(is));
                            P(cpu(is));
                            CST(t_cpu);
                            TRANSIT(disk(is));
                          END;

/STATION/NAME=disk;
        SCHED=FIFO,PRIOR;
        SERVICE=BEGIN
                  CST(t_disk);
                  CST(t_cpu);
                  V(cpu(is));
                END;
        TRANSIT(cs)=niu(is),dcs(ic);
        TRANSIT(zm)=niu(is),dzm(ic);
        TRANSIT(im)=niu(is),dim(ic);

/CONTROL/TMAX=simtime;PERIOD=simtime/100;
        ACCURACY=imwsmes,zmes,buffer,niu;
        TEST=BEGIN
```

```
            FOR i:=1 STEP 1 UNTIL nb_ws DO
            BEGIN
              IF (CRESPONSE(imwsmes(i))>0.0001) AND
              (CRESPONSE(zmes(i))>0.0001) THEN
               BEGIN
                IF (CRESPONSE(imwsmes(i))<0.1*MRESPONSE(imwsmes(i))) AND
                   (CRESPONSE(zmes(i))<0.1*MRESPONSE(zmes(i))) THEN STOP;
               END;
             END;
            END;
        ENTRY=BEGIN
         PRINT("*                                                        *");
         PRINT("*        file  :    D I S F S L S . Q                    *");
         PRINT("*                    results                             *");
         PRINT("This program simulates the behaviour of ",nb_ws);
         PRINT("workstations connected through a LAN with a");
         PRINT("DIStributed image File Server.LOCAL STORAGE is assumed.");
         PRINT(" ");
         PRINT("transmission rate",capac,"Mbits/sec");
         PRINT("number of file servers :",nb_fs);
         PRINT(" ");
               END;
         EXIT=BEGIN
                FOR i:=1 STEP 1 UNTIL nb_ws DO
                BEGIN
                  dlim:=dlim+MRESPONSE(imwsmes(i));
                  dlcs:=dlcs+MRESPONSE(csmes(i))+
                            MRESPONSE(cpuws(i),cs(i));
                  dlzm:=dlzm+MRESPONSE(zmes(i));
                  PRINT("Waiting time for an image display at ",i,
                        "workstation :",MRESPONSE(imwsmes(i)));
                  PRINT("Waiting time for a zoom display at ",i,
                         "workstation :",MRESPONSE(zmes(i)));
                  PRINT("Waiting time for a case at ",i,"workstation :",
                        MRESPONSE(csmes(i))+MRESPONSE(cpuws(i),cs(i))-t_cpu);
              END;
              PRINT("Total throughput",MTHRUPUT(network)*8.192E-3,"Mbits/s");
              PRINT("Waiting time for a zoom display",dlzm/nb_ws);
              PRINT("Waiting time for an image display",dlim/nb_ws);
              PRINT("Waiting time for a case ",dlcs/nb_ws-t_cpu);
              PRINT("number of collisions",COLLI);
              PRINT("number of potentially discarde packets",discard);
          END;

/EXEC/BEGIN
         FOR i:=1 STEP 1 UNTIL nb_ws DO
                           BEGIN im(i).ic:=i;
                                 ws(i).is:=i;
                                 imws(i).ic:=i;
                                 dim(i).ic:=i;
                                 zm(i).ic:=i;
                                 dzm(i).ic:=i;
                                 cs(i).ic:=i;
                                 dcs(i).ic:=i;
                                 dsp_im(i):=0;
                           END;
       FOR i:=1 STEP 1 UNTIL nb_fs DO BEGIN
                                 niu(i).is:=i;
                                 query(i).is:=i;
                                 buffer(i).is:=i;
```

```
                                    cpu(i).is:=i;
                                    disk(i).is:=i;
                                        END;
        t_thk:=10;                & thinking time at the workstation
        t_disk:=0.4;
        t_cpu:=0.25E-4;
        t_prg:=0.5E-5;            & propagation time of the packet thru the net
        t_slot:=0.00512E-3;
        t_del:=0.96E-5;
        t_jam:=0.32E-5;
        t_pclin:=0.006;
        t_pclout:=0.006;
        simtime:=20000;
        nb_pck:=64;               & number of packets per block
        capac:=2.0E+6;
        t_tmx:=8192/capac;
        SIMUL;
      END;

/END/
$EOD
```

A.4 Simulation Program for Image Transfers Using FDDI-II

```
$run qnap02

/CONTROL/OPTION=NSOURCE,RESULT;
         NMAX=30;
         CLASS=ALL QUEUE;

& this program simulates the fiber distributed data
& interface (FDDI) connecting several stations tarnsmitting images
& and voice.

/DECLARE/INTEGER nb_st=3,                & number of stations connected
                 nb_fs=1,                 & number of file servers
                 i,
                 m,
                 mes_l,                   & average message length in bits
                 nb_mes,
                 nb_pim,
                 nb_pzm,
                 lbl(nb_st);
          REAL t_thk,                     & thinking time at the station
               ttrt,                      & target token return time
               f,                         & % of bandwith allocated to voice.
               t_token,                   & time for the token transmission
               t_query,                    & time for the transm. of a query
               t_prg,                     & phys. ring latency+stat. latency
               t_sgm,                     & segmentation time per packet
               a,
               length,
               timer(nb_st),              & marker of the token arrival
               capac;                     & capacity of the network in bits/s
          QUEUE INTEGER is;
          CLASS INTEGER ic;
          CLASS zm(nb_st),
                im(nb_st),
                junk_zm,
                junk_im;
          QUEUE network,
                im_mes(nb_st),
                zm_mes(nb_st),
                statio(nb_st),
                niu(nb_st),
                poll;
          REF CUSTOMER C;
          REF QUEUE t_q;
          FLAG fl(nb_st),flg;
          BOOLEAN luck;

/STATION/NAME=network;
         TYPE=RESOURCE,SINGLE;

/STATION/NAME=im_mes(1 STEP 1 UNTIL nb_st);
         TYPE=RESOURCE,INFINITE;

/STATION/NAME=zm_mes(1 STEP 1 UNTIL nb_st);
         TYPE=RESOURCE,INFINITE;

/STATION/NAME=statio(1 STEP 1 UNTIL nb_st-nb_fs);
         INIT(zm)=IF is=ic THEN 1
                           ELSE 0;
         SERVICE(zm)=BEGIN
```

```
                    EXP(t_thk/2);
                    P(im_mes(is));
                    TRANSIT(niu(is),im(is),0);
                  END;
        SERVICE(im)=BEGIN
                    EXP(t_thk/2);
                    luck:=DRAW(0.2);
                    IF luck=TRUE THEN TRANSIT(statio(is),zm(is))
                                 ELSE BEGIN
                                        P(zm_mes(is));
                                        TRANSIT(niu(is),zm(is),1);
                                      END;
                  END;

/STATION/NAME=statio(nb_st-nb_fs+1 STEP 1 UNTIL nb_st);
        SCHED=FIFO,PRIOR,PREEMPT;
        SERVICE(im)=BEGIN
                    FOR i:=1 STEP 1 UNTIL nb_pim-1 DO
                    BEGIN
                      CST(t_sgm);
                      C:=NEW(CUSTOMER);
                      TRANSIT(C,niu(is),junk_im);
                    END;
                    CST(t_sgm);
                  END;
        SERVICE(zm)=BEGIN
                    FOR i:=1 STEP 1 UNTIL nb_pzm-1 DO
                    BEGIN
                      CST(t_sgm);
                      C:=NEW(CUSTOMER);
                      TRANSIT(C,niu(is),junk_zm);
                    END;
                    CST(t_sgm);
                  END;
        TRANSIT=niu(is);

/STATION/NAME=niu(1 STEP 1 UNTIL nb_st-nb_fs);
        SERVICE=BEGIN
                  lbl(is):=0;
                  WHILE lbl(is)=0 DO
                  BEGIN
                   WAIT(fl(is));
                   a:=ttrt-(TIME-timer(is));
                   timer(is):=TIME;
                   IF a<=0 THEN
                   BEGIN
                     timer(is):=timer(is)+a;
                     RESET(fl(is));
                     SET(flg);
                   END
                           ELSE
                   BEGIN
                     P(network);
                     CST(t_query);
                     V(network);
                     RESET(fl(is));
                     SET(flg);
                     lbl(is):=1;
                   END;
                 END;
```

```
                END;
          TRANSIT=station(nb_st-nb_fs+1 STEP 1 UNTIL nb_st),
                          1,nb_fs REPEAT nb_fs-1;

/STATION/NAME=niu(nb_st-nb_fs+1 STEP 1 UNTIL nb_st);
         SCHED=FIFO,PRIOR;
         SERVICE(junk_im,junk_zm)=BEGIN
                         WAIT(fl(is));
                         WHILE nb_mes=1 DO
                         BEGIN
                           nb_mes:=nb_mes+1;
                           a:=ttrt-(TIME-timer(is));
                           timer(is):=TIME;
                           IF a<=0 THEN
                           BEGIN
                             timer(is):=timer(is)+a;
                             RESET(fl(is));
                             SET(flg);
                             nb_mes:=1;
                             WAIT(fl(is));
                           END;
                         END;
                         P(network);
                         CST(mes_l/((1-f)*capac));
                         V(network);
                         a:=a-(TIME-timer(is));
                         IF a>0 THEN
                         BEGIN
                           IF niu(is).NB>1 THEN TRANSIT(OUT)
                                           ELSE
                         BEGIN
                           RESET(fl(is));
                           SET(flg);
                           nb_mes:=1;
                           TRANSIT(OUT);
                         END;
                       END
                                 ELSE
                       BEGIN
                         RESET(fl(is));
                         SET(flg);
                         nb_mes:=1;
                         TRANSIT(OUT);
                       END;
                     END;
      SERVICE(im)=    BEGIN
                         WAIT(fl(is));
                         WHILE nb_mes=1 DO
                         BEGIN
                           nb_mes:=nb_mes+1;
                           a:=ttrt-(TIME-timer(is));
                           timer(is):=TIME;
                           IF a<=0 THEN
                           BEGIN
                             timer(is):=timer(is)+a;
                             RESET(fl(is));
                             SET(flg);
                             nb_mes:=1;
                             WAIT(fl(is));
                           END;
```

```
                    END;
                    P(network);
                    CST(mes_l/((1-f)*capac));
                    V(network);
                    a:=a-(TIME-timer(is));
                    IF a>0 THEN
                    BEGIN
                      IF niu(is).NB>1 THEN
                      BEGIN
                        V(im_mes(ic));
                        TRANSIT(statio(ic));
                      END
                                        ELSE
                      BEGIN
                        RESET(fl(is));
                        SET(flg);
                        nb_mes:=1;
                        V(im_mes(ic));
                        TRANSIT(statio(ic));
                      END;
                    END
                            ELSE
                    BEGIN
                      RESET(fl(is));
                      SET(flg);
                      nb_mes:=1;
                      V(im_mes(ic));
                      TRANSIT(statio(ic));
                    END;
                  END;
SERVICE(zm)=      BEGIN
                    WAIT(fl(is));
                    WHILE nb_mes=1 DO
                    BEGIN
                      nb_mes:=nb_mes+1;
                      a:=ttrt-(TIME-timer(is));
                      timer(is):=TIME;
                      IF a<=0 THEN
                      BEGIN
                        timer(is):=timer(is)+a;
                        RESET(fl(is));
                        SET(flg);
                        nb_mes:=1;
                        WAIT(fl(is));
                      END;
                    END;
                    P(network);
                    CST(mes_l/((1-f)*capac));
                    V(network);
                    a:=a-(TIME-timer(is));
                    IF a>0 THEN
                    BEGIN
                      IF niu(is).NB>1 THEN
                      BEGIN
                        V(zm_mes(ic));
                        TRANSIT(statio(ic));
                      END
                                        ELSE
                      BEGIN
                        RESET(fl(is));
```

```
                        SET(flg);
                        nb_mes:=1;
                        V(zm_mes(ic));
                        TRANSIT(statio(ic));
                      END;
                    END
                            ELSE
                    BEGIN
                      RESET(fl(is));
                      SET(flg);
                      nb_mes:=1;
                      V(zm_mes(ic));
                      TRANSIT(statio(ic));
                    END;
                  END;

/STATION/NAME=poll;
        INIT(junk_im)=1;
        SERVICE=BEGIN
                  IF niu(m).NB>0 THEN
                  BEGIN
                    SET(fl(m));
                    WAIT(flg);
                  END
                                    ELSE timer(m):=TIME;
                  CST(t_token+t_prg/nb_st);
                  m:=m+1-INTREAL(m/nb_st)*nb_st;
                  RESET(flg);
                  TRANSIT(poll);
                END;

/CONTROL/TMAX=600;PERIOD=300;
        TEST=OUTPUT;
        ACCURACY=im_mes(1 STEP 1 UNTIL nb_st-nb_fs),
                 zm_mes(1 STEP 1 UNTIL nb_st-nb_fs),
                 niu(nb_st-nb_fs+1 STEP 1 UNTIL nb_st);
        ENTRY=BEGIN
                PRINT("      INPUT PARAMETERS");
                PRINT("number of workstations          ",nb_st);
                PRINT("number of file servers          ",nb_fs);
                PRINT("buffer size at the file server  ",mes_1,"bps");
                PRINT("isochronous bandwidth allocation",f*100,"%");
                PRINT("target token return time        ",ttrt*1000,"msec");
                PRINT("total cable length              ",length,"KM");
                PRINT("turn around propagation time    ",t_prg*1000,"msec");
                PRINT("number of packets per image     ",nb_pim);
                PRINT("thinking time at the workstation",t_thk,"sec");
              END;

/EXEC/BEGIN
       FOR i:=1 STEP 1 UNTIL nb_st DO
       BEGIN
         statio(i).is:=i;
         niu(i).is:=i;
         timer(i):=0;
         zm(i).ic:=i;
         im(i).ic:=i;
       END;
       m:=1;
       t_thk:=10;
```

```
        t_sgm:=0.25E-1;
        nb_mes:=1;
        capac:=1.0E8;
        mes_l:=32768;
        t_token:=5E-6;
        t_query:=5E-6;
        length:=10.0;
        nb_pim:=256;
        nb_pzm:=16;
        t_prg:=0.5E-5*length+nb_st*0.5E-5;
        ttrt:=4.0E-3;
        f:=0.0;
        BEGIN
          SIMUL;
        END;
      END;

/END/
```

References

[ATKI87] M. W. Atkinson A. R. Sastry, "A Simulation Model for the FDDI Token Passing Scheme", Proc. of *IEEE International Conference on Communications' 87 : Communications-Sound to Light*, pp. 1300–1304, June 1987.

[BURD88] R. M. Newman, Z. L. Burdikis, J. L. Hullet, "The QPSX MAN", *IEEE Communications Magazine*, vol 26, no. 4, pp.20–28, Apr.1988.

[CHIO87] G. Chiola, "GreatSPN Users' Manual", Torino, Italy, Sept 1987.

[CCIT84] CCITT Study Group VII, Draft Recommendations: Message Handling Systems, October 1984.

[DYKE88] D. Dykeman and W. Bux, "Analysis and Tuning of the FDDI Media Access Control Protocol", *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 6, pp. 997–1010, July 1988.

[FORS85] H. Forsdick, "Explorations into Real-time Multimedia Conferencing Systems", *Computer Message Systems-85* R. Uhlig (editor), Elsevier Science Publishers B. V. (North-Holland), 1985, pp. 331-347.

[GEOR88] N. D. Georganas, M. Golberg, L. O. Barbosa, E. Roger, P. Gross, "A Multimedia Communications System for Medical Applications", submitted to ICC'89, Boston, Massachusetts, June 1989.

[GREE87] L. Green, "Performance analysis of FDDI", *COMPCON SPRING'87*, 32nd *IEEE Computer Society International Conference*, San Francisco, pp. 441–443, 1987.

[GROW82] R. M. Grow, "A Timed Token Protocol for Local Area Networks", *Electro '82*, Token Access Protocols (17/3), May 1982.

[HORA85] W. Horak, "Office Document Architecture and Office Document Interchange Formats: Current Status of International Standardization", *IEEE Computer magazine*,vol. 18, No 10, Oct 1985, pp 50–59.

[IEEE84] ANSI/IEEE Standard 802.3 "CSMA/CD Access Method", Dec. 21, 1984.

[IEEE84] ANSI/IEEE Standard 802.4 "Token-Passing Bus Access Method", Dec. 17, 1984.

[IEEE85] ANSI/IEEE Standard 802.5 "Token-Ring Access Method", March 19, 1985.

[IEEE88] IEEE Draft Proposal Standard 802.6 "Metropolitan Area Network (MAN), DQDB Dual Bus Media Access Control and Physical Layer Protocol Document", Feb. and July 1988.

[JOHN85] M. J. Johnson, "Proof That Timing Requirements of the FDDI Token Ring Protocol are Satisfied", *IEEE Transactions on Communications*, vol. C-35, no. 6, pp. 620–625, June 1987.

[KLES86] R. W. Klessing, "Overview of Metropolitan Area Networks", *IEEE Communications Magazine*, vol 24, no. 1, pp.9–15, Jan.1986.

[LANT86] K. A. Lantz, "An Experiment in Integrated Multimedia Conferencing", Proceedings of the Conference on Computer-supported Cooperative Work, Austin, Texas, December 1986, pp. 267-275.

[LAM80] S. S. Lam, "A Carrier-sense Multiple Access Protocol for Local Area Networks", *Computer Networks*, vol 4, 1980.

[MARSA87] M. A. Marsan, G. Chiola, "On Petri-nets with Deterministic and Exponentially Distributed Firing Times", *Lecture Notes in Computer Science, Advances in Petri Nets 1987*, vol 266, Springer Verlag, pp. 132–145, .

[MAXE82] N. F. Maxemchuk, "A Variation on CSMA/CD That Yields Movable TDM Slots in Integrated Voice/Data Local Networks", *Bell Syst. Tech. J.*, vol 61, no. 7, Sept. 1982.

[MOLL88] J. F. Mollenauer, "Standards for Metropolitan Area Networks", *IEEE Communications Magazine*, vol 26, no. 4, pp.15–19, Apr.1988.

[PETE81] J. L. Peterson, *Petri-net Theory and the Modeling of Systems*, Prentice-Hall, Englewood Cliffs, 1981.

[POGG85] A. Poggio, J. J. Garcia Luna Aceves, E. J. Craighill, D. Moran, L. Aguilar, D. Worthington and J. Hight, " CCWS : A Computer-based Multimedia Information System", *IEEE Computer Magazine*, vol 18, no. 10, pp. 92–103, Oct.1985.

[QNAP84] D. Potier, "New Users' Introduction to QNAP2", *Technical report no. 40*, INRIA, France, October 1984.

[REYN85] J. K. Reynolds, J.B. Postel, A. R. Katz, G. G. Finn and A. L. DeSchon, "The DARPA Experimental Multimedia Mail System", *IEEE Computer Magazine*, vol 18, no. 10, pp.82–89, Oct.1985.

[ROSS86] F. E. Ross, "FDDI-a Tutorial", *IEEE Communications Magazine*, vol 24, no. 5, pp.10–17, Apr.1988.

[SACH88] S. R. Sachs, "Alternative local area network Protocols", *IEEE Communications Magazine*, vol 26, no. 3, pp.25–45, March 1988.

[SAKA85] S. Sakata and T. Ueda, " A Distributed Interoffice Mail System", *IEEE Computer Magazine*, vol 18, no. 10, pp.106–116, Oct.1985.

[SARI85] S. Sarin and I. Greif, "Computer-Based real-time Conferencing Systems", *IEEE Computer Magazine*, vol 18, no. 10, pp. 33–45, Oct.1985.

[SCIL87] A. Scill, M. Zieher, "Performance Analysis of the FDDI 100 Mbit/s Optical Token Ring", *Proc. of High Speed Local Area Networks*, Elsevier Science Publishers B.V., pp. 53–74, 1987.

[SEVC86] K. C. Sevcik, M. J. Johnson, "Cycle Time Properties of the FDDI Token Ring Protocol", *Technical Report CSRI-179*, University of Toronto, April 1986.

[STAL84] W. Stallings, *Local Area Networks-An Introduction*, New York: Macmillan Publishing Company, 1984.

[STEF87] M. Stefik, D. G. Bobrow, G. Foster, S. Lanning, D. Tatar, "*WYSIWIS Revised: Early Expiriences with Multiuser Interfaces*", ACM Transactions on Office Information Systems, vol. 5, no 2, April 1987, pp. 147–167.

[THOM85] R. H. Thomas, H. C. Forsdick, T.R. Crowley, R.W. Schaaf, R.S. Tomlinson, V. M. Travers and G. G. Robertson, " Diamond : A Multimedia Message System Built on a Distributed Architecture", *IEEE Computer Magazine*, vol 18, no. 12, pp.65–78, Dec.1985.

[TOBA80] F. A. Tobagi and V. B. Hunt, "Performance Analysis of Carrier Sense Multiple Access with Collision Detection", *Computer Networks*, vol 4, 1980.

[TSAO84] C. David Tsao, "A Local Area Architecture Overview", *IEEE Communications Magazine*, vol 22, no. 8, pp.7–11, Aug.1984.

[TSIN87] P. Tsingotjidis, R. Kositpaiboon, N. D. Georganas, "Radiological Image Transfers over Local Area Networks", *MONTECH'87, Conference on Communications, Montreal, Canada*, pp.71–74, Nov. 1987.

[TSIN88] P. Tsingotjidis, R. Kositpaiboon, N. D. Georganas, "Using FDDI-II for Radiological Image Transfers: a Simulation Study", *EURINFO'88, First European Conference on Information Technology for Organisational Systems, Athens, Greece*, pp.327–333, May 1988.

[ULM82] J. M. Ulm, "A Timed Token Local Area Network and its Performance Characteristics", *Proc. of the 7th Conf. on Local Computer Networks*, IEEE, pp. 50–56, Feb. 1982.

[YATS85] R. Yatsuboshi, A. Takeyama, I. Iida "LAN Architecture for Image Database Using Classified Network Access Methods", *Passific Computer Communications '85*, Elsevier Science Publishers B.V. pp.563–567