PERFORMANCE EVALUATION OF A COMMUNICATION NETWORK

by

Ariage T Ncube

Thesis prepared in partial fulfilment of the requirements for the degree of MSc in Electrical Engineering at the University of Cape Town

Cape Town **1995**

Acknowledgements

I wish to thank a number of people for their help with this project. My special thanks and appreciation go to Mr Neco Ventura, my Supervisor, for patience, guidance and support throughout the work. My thanks are due also to the Communications Research Group team, Tony Putman, Heidi de Wet as well as to Alvin Rajoo formerly with Tran Systems, for their efforts to produce careful documentation on the DM system. This documentation made my task of understanding the DM system so much easier. I also wish to extend my thanks to Derek Organ of Telkom SA for assisting with the tests on the Easy Access Network. And to my mate, Matthew Wicks - I have ever valued your friendship, cooperation and encouragement. I also wish to express my gratitude to my family whose unfailing encouragement and love have kept me going over the years, and it is to them that this thesis is dedicated. My final thanks go to UCT for providing vital financial assistance in my studies.

Terms of Reference

Brief

The brief on the thesis was given by Mr M J Ventura, my Supervisor in 1992, for the analysis of a Value Added Network using a computer simulation package, Network II.5 produced by CACI.

Objectives

The general objectives of the project were

(i) To analyze the performance of the NC and the NA elements of a specific Value Added Network Communication Network in order to obtain results that would help in a future upgrade of elements of the network.

(ii) To evaluate the performance of the Domain Manager, a proposed upgrade of the NA, one of the key elements of the original network, and highlight any performance improvements or otherwise realised in the upgrade.

Project Scope

(a) To model the performance of the original Value Added Network comprising a Network Administrator and a cluster of Network Concentrators using suitable performance measures.

(b) To model the performance of the new Domain Manager using indices similar to those used for the NA.

(c) Draw a performance comparison between the network using the NA and the one using the DM system.

This work was to constitute a half thesis.

Abstract

This work uses a simulation package to analyze the performance of a specific X.25 Value Added Communication Network (VAN), comprising a Network Administrator and a cluster of Network Concentrators (NCs). Detailed models of the network elements are developed, and system performance is analyzed in terms of network response time for the terminal users communicating across the network, identification of areas of bottlenecks in the NC system, and NC system throughput. The througput parameter is represented by the percentage of the output traffic to the load offered from the network of terminals. The performance of the Network Administrator system is also modeled, focusing particularly on the automatic restoration of service to failed NCs through downloading of operating software sets over the X.25 network. The final part consists of a model of a modification of the network manager components of the network as proposed in the DM upgrade project. In this section, NC loading time is again the desired performance indicator. It is shown that there is no noticeable improvement in this parameter between the original system and the proposed upgrade, both systems having a minimum loading time of about 3.5 to 6 minutes for a small network.

Glossary of Terms and Abbreviations

ACK Acknowledgement

ARQ Automatic Repeat Request

ATX Asynchronous-to-X.25 concentrator. An NC serving asynchronous terminals.

BPM Block Processor Module. The multiprocessor structure that forms the fundamental unit of the NC and the NA.

CRG Communications Research Group at the University of Cape Town. The CRG in conjunction with Tran Systems developed the Domain Manager system.

DCE Data Circuit-terminating Equipment. The name given to the equipment provided by the network provider for the attachment of user devices to the network. The NC can be seen as a DCE for asynchronous terminals, on the other hand the NC can be viewed as a DTE from the X.25 network.

DM Domain Manager. A network management system developed to replace the NA.

DTE Data Terminal Equipment A generic term for any user device connected to a data network, eg a computer or terminal.

FCFS A scheduling discipline useful in analysis of queuing systems. Based on the first come first rule.

FCS Frame Check Sequence. A generic term given to the additional bits appended to a transmitted frame or message by the source to enable the receiver to detect possible transmission errors.

HDLC High Level Data Link Control. An ANSI standard protocol set defined to control the exchange of data across either a point-to-point data link or a multidrop data link. X.25 is an element of the HDLC.

IOM Input/Output Module. An ATX is sometimes referred to as an IOM.

IOP I/O processor. Either one of the two Z80A processors in the BPM or used to refer to the X.25 card in the DM system.

NA Network Administrator. A kind of a network management system that provides management services to a cluster of network elements.

NC Network Concentrator. A network element used to provide PAD functions to asynchronous terminal in a data network, then specifically called an ATX. Alternatively the NC can be configured as an XTX.

NETWORK II.5 The computer simulation package used in this project.

Network Management A generic term used to embrace all the functions and entities involved in the management of a data network. This includes configuration management, fault handling, and the gathering of statistics relating to the usage of the network.

NMS Network Management System

PAD Packet Assembler/Disassembler, a device used with an X.25 packet switching network to allow character mode terminals to communicate with a packet mode device such as a computer. Used interchangeably with ATX.

PC-320 A proprietary term used to refer to the X.25 card in the DM system. Sometimes called the Coprocessor card.

PC-321 An essentially passive system that adds another channel to the PC-320 card.

ROM-State NC The initial state of an NC after power-on, before the downloadable software sets have been installed, when the NC is running code from its ROM.

TUP Trans Unix Platform. The hardware and software platform used for Tran Systems products, including the DM.

WAN Wide Area Network. A generic term used to describe any form of network - private or public, that covers a wide geographical area.

WLE Abbreviation for Work List Entry. One of the structures used for interprocess communication in the NA/NC system. 16 bytes long.

X.25 An international CCITT standard protocol defined for the interface of a data terminal device, such as a computer to a packet switched data network.

XTX X.25-to-X.25 NC. Operates as a virtual circuit switch between ATXs.

Table of Contents

Chapter One

Introduction

1.1 Purpose and Problem Definition

Since about two years ago, the Communications Research Group at the University of Cape Town has been involved in a number of projects related to the development of an X.25 Value Added Communication Network (VAN). One of these projects involves the replacement of the original Network Administrator (NA) with a Domain Manager (DM) that is based on a newer PC platform running the application software on the Unix Operating System. There was a requirement to obtain a fuller understanding of the performance of the original system using the NA, and the performance of the same network using the DM. This study represents one attempt to provide such information.

Objectives

This work presents a comprehensive performance analysis of a Network Concentrator (NC) and a Network Administrator (NA) of the above X.25 network. For the NC, the main aims of the study were to derive results concerning any system bottlenecks and to measure the throughput of the system. In addition network response time is analyzed using some typical configurations. For the NA, the aim was to obtain a measure of the time required to download operating software sets from the NA to a failed NC element over the X.25 network. The same performance index used for the NA was also used for the Domain Manager. The simulation results were validated against those obtained from tests on the actual system.

1.2 X.25 Network Performance Issues

X.25 is a CCITT standard which defines the interface between a DTE and a DCE for terminals operating in the packet mode, and connected to a public data network by a dedicated switch. Computer-based packet switches transport data through the network via individual packets or groups of packets. A data packet can vary in size from 128 bytes to

1024 bytes. A data packet size of 128 bytes is used throughout this report unless otherwise stated. Along the path, the packets travel from node to node where they are stored, checked for errors and forwarded to the next node. The packet switching method allows data concentration, sharing of facilities and some circuit redundancy to protect against susceptibility to network failures.

A number of performance issues are open for investigation in an X.25 Value Added Network of the kind described above.

a) The packet switching nodes introduce a finite delay in the delivery time of each frame. This delay impacts on the user-perceived network response time.

b) The nodes must possess sufficient processing power to perform the switching function and at the same time carry out other protocol processing in an optimal manner, again in respect of the delay in the execution/completion of users tasks.

c) The environment in which the X.25 network operates, in regard to transmission errors, and the error correction scheme used in the X.25 protocol impacts on the network performance.

d) The manner in which flow control is implemented will also affect the system throughput.

e) The routing mechanism adopted for the XTX.

f) Tunable and other variables within the X.25 protocol such as frame size, number of active virtual circuits, the amount of resources consumed by buffer management, and protocol overhead will also affect network performance.

Low volume users can access the X.25 network nodes by dialling in packet assembler/disassembler (PADs) or Network Concentrators (NCs) over an asynchronous dialup connection. Users of this interactive terminal interface (X.3, X.28 and X.29) will not have any error correction on the dial-up links between the terminal and the NC because there is no provision in the X.28 protocol. Some of these considerations are used to analyze the performance of the Value Added Network.

1.3 The Simulation Option

Performance evaluation and trade-off analysis are the central issues in the design, analysis and management of communication networks and distributed systems. Simulations can be used to predict performance of existing networks and for evaluating the comparative performance of alternate designs for new systems. Detailed models of a network can be simulated and hence the design space can be finely explored. It is relatively easy to combine simulation with analytical as well as empirical models and measured data, to create a rapid prototyping environment for communication networks.

1.4 Simulation Tools

The package used in this work is called Network II.5 developed by CACI. It uses modeling paradigms such as queues, petri nets and finite state machines [A-2], to give an easy to use, English like description of the input program. It is an event driven system. Network II.5 was used because it supports a powerful set of commands and statements, thereby reducing programming complexity and time. Moreover, it provides a good range of statistical distributions and can be used to derive a number of useful network performance measures like throughput, message statistics, and hardware utilization figures.

1.5 Limitations

The Value Added Network that is the focus of this project has been operational for some years now and has also gone through various stages of upgrading and enhancement. This has greatly increased the amount of documentation required in order for a complete understanding of the system to be obtained. Hence the results reported here are only a reflection of the understanding that could be built about the system within the project time constraints. In carrying out verification tests for the NC loading time, only a few cases could be performed on the live network. The main results were obtained from the Network operators' own observations made in the daily logs. In order to minimise the effect of inaccuracies, the following approach was used;

(a) to identify and isolate major network blocks and processes and simulate these aspects of interest to the required detail level but treat the rest of the system coarsely.

(b) make assumptions that could easily be identified and changed by a different user of the model, and practise good documentation discipline in the simulation programs.

1.6 Workload Characterisation & Modeling Approach

1.6.1 Statistical Representation

Estimating the performance of a system requires the definition of a model that captures the relevant features of the actual system. Essentially two types of approach can be followed to define the workload model, using either a deterministic approach or a probabilistic view of the system operations. In the deterministic case the workload of each single processor is completely known, but the results obtained by this technique are limited to the specific case under test, and functional relationships between input parameters and output results cannot easily be obtained.

Using instead a probabilistic view of the system, it is possible to develop a stochastic model, in which many details of the system workload are not individually represented. A stochastic model tries to capture the essence of the system operations by defining the workload parameters to be random variables with appropriate probability distributions, and providing results in the form of functional relationships between system parameters and performance [C-5]. The main motivation for the adoption of probabilistic viewpoint is that it was not possible to capture all the aspects of the system under study as this would make the models too large. Examples of parameters that were modeled by use of statistical distribution functions are memory access times, traffic patterns, instruction execution times, service times etc. In Chapters 3 and 4 the simulations had to be performed with a number of deterministic input parameters because a form of performance measurement was desired, but again this deterministic modeling yields results of relevance only at particular design points.

1.6.2 Modeling Approach

Fig.1-1 summarises the modeling cycle in the whole project. It reflects throughout the refinement process via iteration. Initially, only an approximate model was constructed and as more knowledge was gained, so the results were more critically analyzed and the model elevated to the required level of complexity/detail. Because each simulation program run was

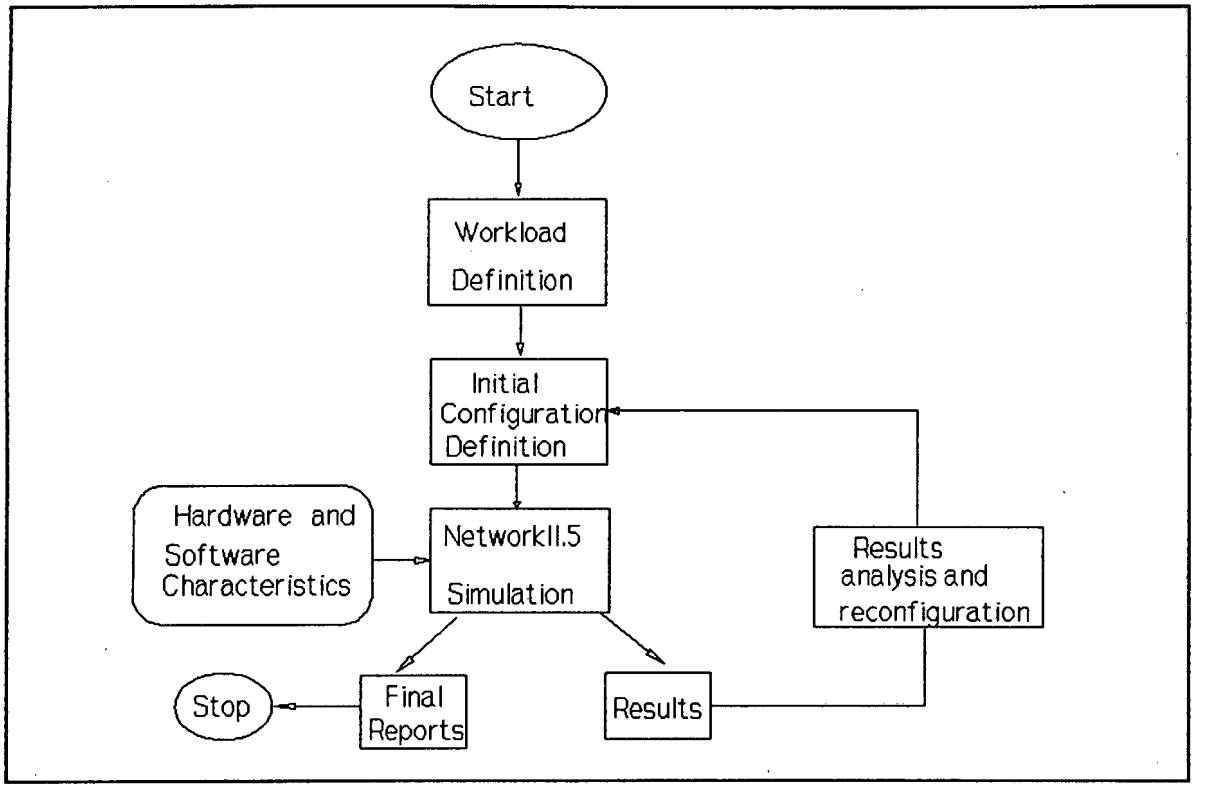


Fig.1-1 Modeling Methodology

used to obtain a single point in a graph or table of results, the internally produced GWORK [C-1] graphs were hardly used. These would be of interest only at a particular design point. In all cases, each model begins with a clear system description, and then a detailed decomposition of the system's hardware blocks into processing elements, transfer devices and storage devices, complete with device attributes of basic cycle times, bus widths, bus protocols, storage capacities, access times, access ports etc is included. A detailed set of operating assumptions is presented and based on the above, a software model was built to drive the simulation. Where appropriate, analyses were carried out to derive input values required in the software model.

1.6.3 Model Abstraction

The definition of the level of abstraction drives the choice of the level of detail of the description of each element, as well as functional relationships and the rules of communication among building blocks. The level of abstraction was chosen around the parameters that significantly and adequately describe system performance. The evaluation (or

estimation) of such parameters is the actual goal of the analysis. In the early stages of this work, the Supervisor was occasionally asked to provide a form of requirements specification to ensure that the goals and objectives of the study would be met. There was thus particular effort to identify the performance measures of each model as efficiently as possible. In all the models, the abstraction is at the functional and system levels.

1.7 The Organization of this Report

Chapter 2 presents a model of a Value Added Network in which a cluster of asynchronous terminals are connected to a Network Concentrator (NC), which provides the functionality required to access the X.25 network. The main concern is the determination of the throughput of the NC, i.e the number of packets from the asynchronous terminals that are processed and forwarded by the NC to the X.25 network in given time, and vice versa. To that end some effort was expended on investigating the NC for areas of potential bottleneck under different conditions of loading, configuration and other functionality. Another important performance measure investigated is the user-to-user response time of the network. Semi-analytical models were developed for various hardware and software aspects of the Block Processor Module (BPM) of the NC in order to derive the mean values for use in the input simulation programs.

The same Value Added Network is analyzed in Chapter 3, with a focus on the Network Administrator, which provides management functionality for the NCs. In this case the central performance index is the time required for the NA to automatically restore service to a failed NC. Several issues are considered in an effort to identify those parameters that impact on this figure, such as network loading, transmission error conditions, system configuration etc.

Chapter 4 treats an upgrade of the NA called the Domain Manager. Using the same performance measure as for the NA, the analysis also takes into account issues relating to the design features of the 80486 and the resource management aspects of the Unix Operating System on which the NMS is based. Considerable effort is spent on the development of semi-analytic models to assist in the construction of the input simulation programs. A discussion of the findings, and a comparison of the performance of the DM to that of the NA is also given in Chapter 4. Chapter 5 presents a Summary of the report, and a comparison of the simulation and measured results. Some aspects of the simulation package are discussed in Appendix A. Some notes on the usage of the simulation programs are given in Appendix D.

Chapter Two

Model of a Network Concentrator

2.1 Introduction

In data networks, a network concentrator (NC) performs the packet assembly and disassembly (PAD) functions that are required for terminals to transmit and receive data on the network. It combines data traffic from multiple terminals or computer ports, and transmits it via X.25 protocol over a single line to the network. Terminals can only access a packet network through a PAD. The NC can also be used as a front-end for non-X.25 host computers, as well as for its usual task of allocating a limited number of ports or virtual circuits at one or more remote locations to a large number of terminals connected to it, in some contention-

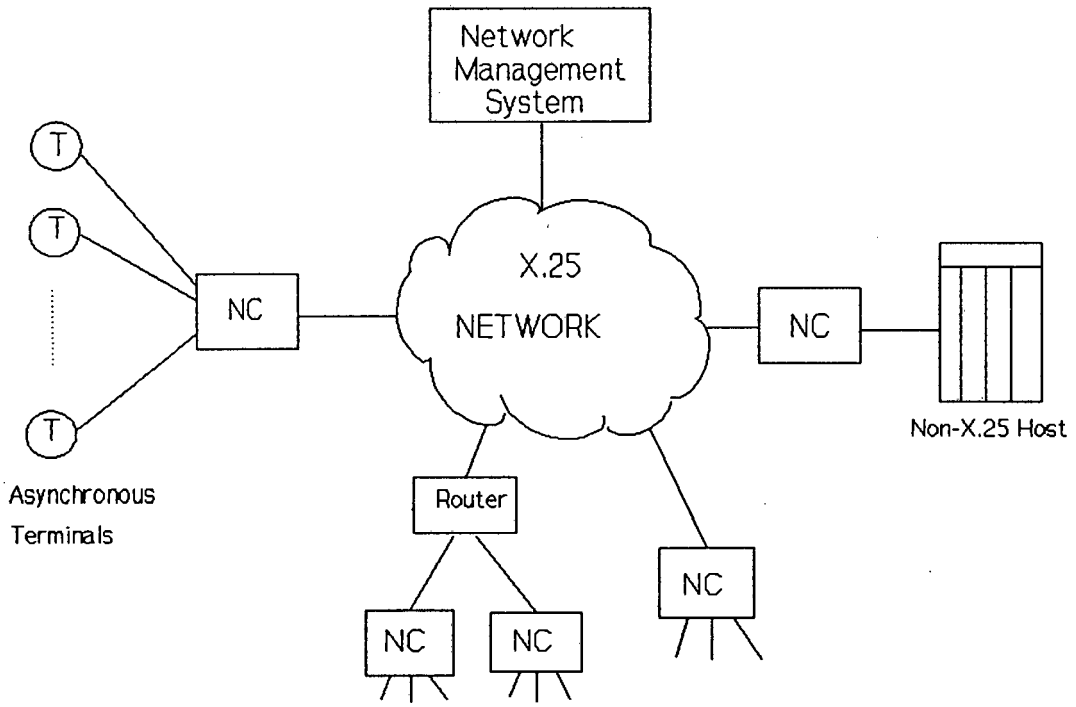


Fig.2-1 Asynchronous X.25 Value Added Network

based mechanism such as FCFS, round robin or priority scheme. Additionally the NC can directly connect to a computer that has X.25 software. One of the X.25 standard parameters

permits a PAD to appear as a DTE or DCE to the connected device. By configuring the NC to present a DCE interface on the link, it looks like a packet network node to the computer, and a large number of terminals can be directly connected to a single computer port. This chapter presents an analysis of the performance of both the ATX NC, and the XTX NC, shown in Fig 2-1 as a router.

2.2 Performance Measures

In the network described above, three performance measures can be identified for the NCs. These are response time between the user terminal and its ATX NC, the response time between a user and another user across the network, and the NC throughput.

2.2.1 Local Response time for the asynchronous terminals

Consider a cluster of asynchronous terminals connected to the ATX NC over say an X.28 connection. Assume that the NC provides several services for the user terminals such as login and other network user identification (NUI) functions. Then for this user, delay exists mainly between the terminal and the NC. If a user had to wait a long time to be allowed to login into the network, that delay would be easily noticeable. However the delay in the X.25 portion of the network is not easily perceived as it would probably involve functions such as large file transfers, which for most users, are not very time-sensitive. The response time is evaluated by considering

- the baud rate over the X.28 connection
- the processing power of the ATX NC and that of the terminal/work-station
- the amount of simultaneous traffic generated by other users connected to the same NC.

2.2.2 User-to-User Response Time

Because a VAN primarily provides a connection between the network users, or a connection between a user and a remote application, a performance measure of interest is the delay that is perceived by a user when trying to access services resident at another host across the network. A related measure is the delay that exists when user and other management information is distributed in NCs as opposed to the case when it is centralised at the higher

level Network Management System. These issues are modeled in Section 2.7.2.

2.2.3 NC Throughput

Assume a cluster of asynchronous terminals sending random messages to other users on the X.25 network via the ATX NC. The throughput, which is a measure of the rate of traffic output, is related to the ratio of messages processed and forwarded to the network by the NC, to the load offered by the asynchronous terminals, or by the ATX NCs in the case of the XTX, in a given time unit. In its simplest sense, throughput would refer to the number of packets transmitted by the BPM in one second. However since the main aim of this section is to investigate the interplay of input and output traffic conditions, it is more instructive to work with a measure that is related to traffic intensity. This is the reason why the throughput is presented in the manner done here. The throughput analysis is limited to that in the direction of the X.25 network side. If symmetry is assumed around the BP in the NC, then the throughput values shall be approximately the same for both directions - on the asynchronous and on the X.25 side. In other words, throughput is considered in the direction of the X.25 network only, and the results obtained therefrom are assumed to hold for the asynchronous side as well.

2.3 System Architecture and Interprocess Communication

The NC's architecture consists essentially of a Basic Processor Module which contains three processors configured as a multiprocessor [D-1]. A functional model of the system is shown in Fig.2-2. All the performance parameters of the NC are based on the BPM. The scenario depicts a situation where the two I/O processors handle the data communication interface of the system. They operate mainly at the character level. The Block Processor handles most of the link and network layer protocols and operating system functions and can be seen as the controller processor. The three processors in the BPM (and the multiple processes that constitute the protocol) are connected together by shared storage. The processes communicate with one another by means of messages called work list entries (WLEs) and associated message buffers. When one process desires to communicate a result to or request a desired action from another process, the source process formats a work list entry containing the fields as shown in the WLE format below, and then issues a POST to the desired target process ID. This WLE is stored in the processing element's Message Queue, and then it can be used by the target process. All module communication between processes is performed by POSTing

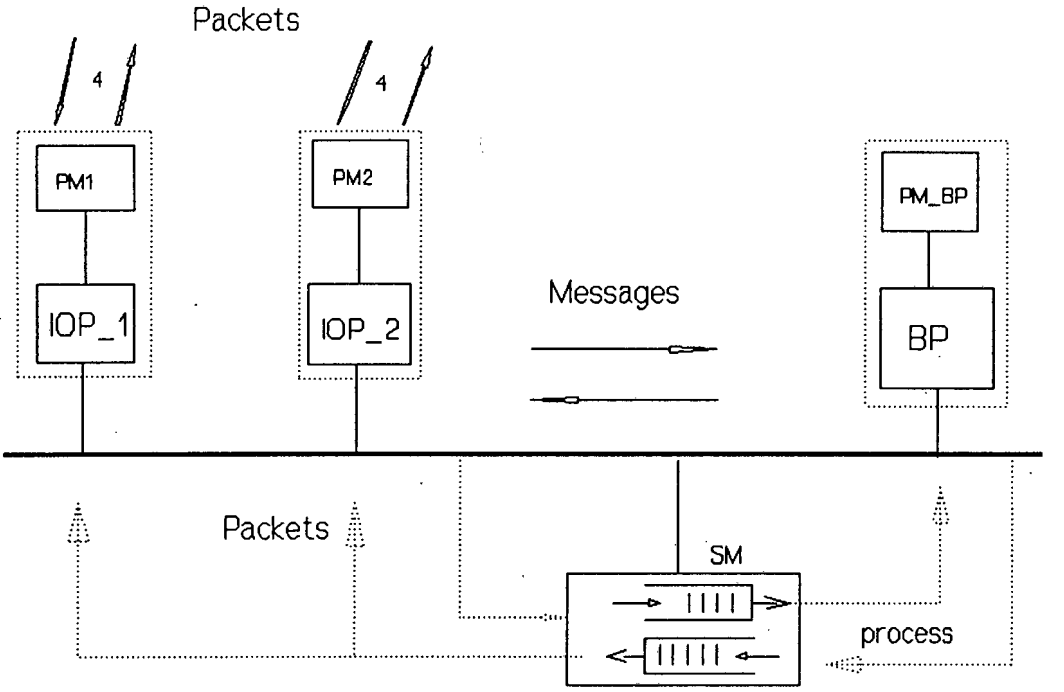


Fig.2-2 A Model of the BPM

a WLE to the target task. The process of posting a WLE is modeled here as if all the three processors were connected together by a global bus. In practice each PE has a link to the shared memory, but because shared memory cannot support multiple, simultaneous access, the concept of a global bus becomes valid. The important assumption made here is that once a PE gains control of the bus, it transmits a complete message before releasing the bus, that is there is no byte interleaved access to the shared memory.

Work List Entry format [D-1]

SENDER ID	RECEIVER ID	BUFFER POINTER	EVENT ID	EVENT DEPENDENT DATA
2	2	4	1	7 bytes

The second most important Interprocess Communication (IPC) vehicle used in the NC is the Message Buffer [D-1]. All data to be sent or received is stored in 70 byte long message buffer segments. If a message is longer than the buffer length, additional message buffers are chained together using the Next Message Buffer pointer. The format of a Message Buffer is as shown below [D-1].

POINTER TO NEXT MSG BUFFER	START OFFSET	MESSAGE LENGTH	MESSAGE DATA AREA
4	1	1	64 bytes

In the queuing model of the BPM above, incoming data packets are stored in the shared memory's incoming queue, and a message (WLE) is sent to the block processor to inform it that a new packet is available in shared memory. In our model the message can be sent directly over the bus or via shared memory, by allocating special files for WLEs for each processor. (In a typical multiprocessing operating system implementation of a Scheduler process would be well represented by passing the messages directly over the common bus) The block processor then processes the information in the WLE and places the packet in the outgoing queue of the Shared Memory, deleting it from the incoming queue. The Block Processor then selects the X.25 port or one of the I/O processors with equal probability of selection to ensure load balancing, to transmit the packet. In practice, in an implementation of the NC, load balancing may not always be performed. Upon transmission of the packet, it is removed from the outgoing queue, and the cycle repeats.

2.4 Traffic Model

(a) Packet Arrival Process

Consider a domain of asynchronous terminals (or non-X.25 host with terminal ports) connected to an ATX NC that provides PAD functions as explained above. The generic term *terminal* here is used to include powerful work-stations (running terminal emulation software) which can participate in file transfers. The asynchronous terminals however typically send a single character on the communication line every time a key on the keyboard is pressed. The assembly of characters into logical blocks is done by the ATX. On the other hand the XTX NC interfaces to ATX NCs and to a WAN or to X.25 hosts, which communicate with the XTX NC in X.25 frame format. Each DTE (terminal or ATX NC) randomly and independently of all other DTEs, generates and transmits a character of 1 byte and a packet (data unit) of 128 bytes to the ATX NC and XTX NC respectively. The packet size is chosen to be 128 bytes because this is most common in X.25 networks. The number of transmissions from a DTE are stationary increments, i.e the distribution of the number of transmissions that occur in any time interval depends only on the length of the time interval. The assumption

of stationary and independent increments is basically equivalent to asserting that at any point in time, the process probabilistically restarts itself, ie the process from any point is independent of all that has occurred (by independent increments) and also has the same distribution as the original process (by stationary increments). In other words the process has no memory and hence exponential interarrival times are to be expected. The analytical basis of this assumption is the exponential distribution which is presented in Appendix B.

(b) Analysis of NC Latency

In the following cases, the various factors that contribute to the latency of the NC are examined. The objective is to characterise the distribution of the latency due to each major component of the NC, so that mean values of the latency seen by each message going through the NC can be computed and fed into the simulator. The analysis distinguishes between the latency of the ATX NC and that of the XTX NC.

Case 1: XTX NC Latencies

(i) XTX NC I/O Processor Latency

For the purposes of analysis, the main functions of the XTX IOP are the following:

- to perform some channel I/O and link level functions of the X.25 protocol

- to communicate with the BP for higher level packet processing

The latency or average delay time of the IOP is measured from the time that a frame is received at the IOP (from the ATX) to the time that a WLE is posted to the BP with respect to this particular packet. We need to characterise this time in order to use it as an input to the simulation program. The components of the delay involved can be broken down as follows.

1. The time to perform link level functions on the received data.

The link level functions themselves consist mainly of error correction and flow controlling the ATX NCs. The error control process involves calculating checksums on the received frames and initiating any retransmissions as necessary. The retransmissions follow the distribution of the errors on the link. The link errors can be described by a Gaussian random

process [B-8] which arises when a large number of independent factors with broad restrictions contribute additively to the end result [B-10]. Let X be the transmission error event. Then the gaussian probability density function f(x) is continuous and is defined by

$$f(x) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

with a cumulative distribution function

$$F(x) = \int_{-\infty}^{\infty} f(x)dx = \frac{1}{\sqrt{2\pi}\,\sigma} \int_{-\infty}^{\infty} e^{-(x-\mu)^2/2\sigma^2} dx$$

where μ and σ^2 are the mean and variance of X respectively. The Gaussian pdf is uniquely and completely described by its mean and variance. It is usual in communication systems to assume a white noise process with a zero mean, single-sided power spectral density.

Further, when the IOP executes a link layer function, it invariably handles several connections from within the same layer and from the upper layer etc, whose data units are likely to vary in length as shown in the protocol interface diagram below due to Halsall [B-2]. In addition there will also be scheduling at the processor of individual tasks or protocol entities at each of the link and packet layers, e.g Send Data and Receive Data primitives. The delay due to process contention time can be estimated if the number of processes in the queue

for the processor is known. This can be achieved using a simple queuing model. It is not easy though to figure out the number of processes in interaction at the processor, and intuitive estimates are used.

2. The bus contention time

The bus or shared memory access contention is resolved by a fixed priority scheme in the BPM. The devices and their priority in descending order are; the BP, DRAM refresh logic,IOP-A, IOP-B, and the Passive Interface Module (PIM). The allocation of priority suggests a geometric distribution of the contention time. Suppose that independent trials, each having a probability p of being a success, are performed until a success occurs. If X is the number of trials required until the first success, then X is said to be a geometric random variable with parameter p [E-4]. In Section 2.5, the average contention time for all the contending devices is computed.

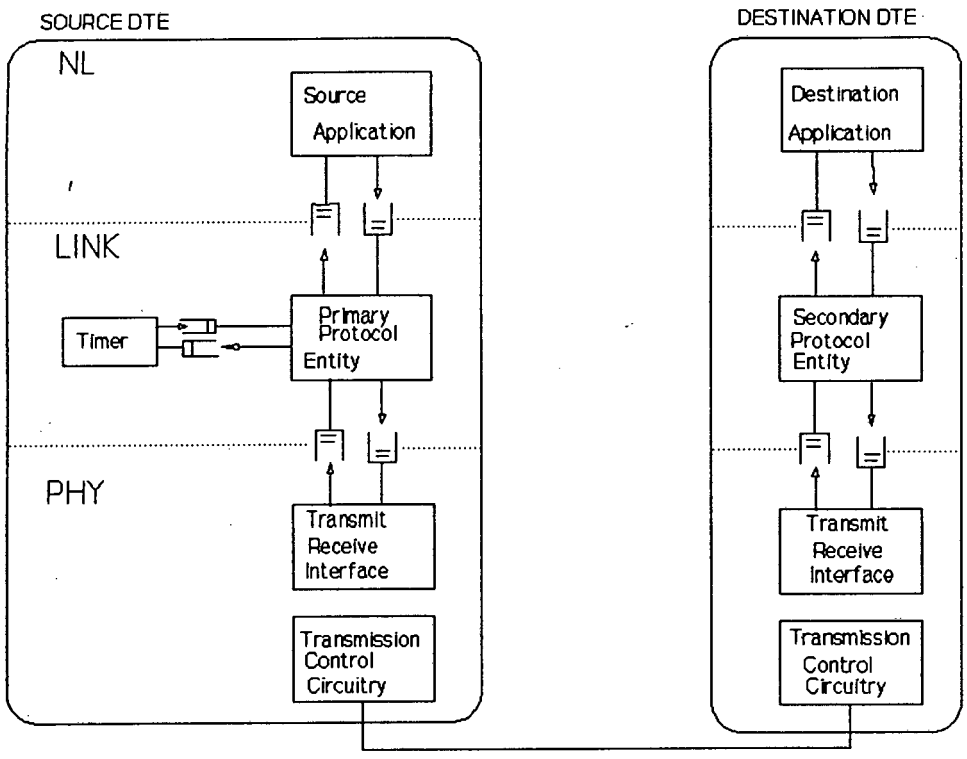


Fig.3-3 Protocol Interface Diagram [B-2]

3. Data transfer time from IOP buffers to Shared Memory

Once the transfer device characteristics have been modeled, based on the average bus contention time, the transfer time can also be estimated. The transfer time is however part of the simulation observation parameters, and is small.

4. The Shared Memory Access Time

Given a processor cycle time and bus cycle pattern, the maximum access time can be determined. The calculation for this parameter is developed later in the discussion of hardware block models.

5. Flow control

Flow control is employed to achieve the following objectives [B-6];

(i) The prevention of throughput degradation and loss of efficiency because of overload of

communication resources.

(ii) Deadlock avoidance. In a sharing environment, if protocol deadlocks occur, a group of communication resources is occupied and none of them can be released before other resources in the same group are released, thus effectively locking up the entire group.

(iii) Fair allocation of resources among competing users.

(iv) Speed matching between the network and its attached users.

There is a natural trade off between allowing a free user access to the network and keeping delay at a level low enough so that retransmission or other inefficiencies do not degrade network performance.[B-3] The flow control mechanism transforms an open queuing network into a closed one, whose solution is difficult. Wu and Chan [B-7] have used Bulk Poisson statistics and entropy maximization techniques to approximate the solution for a packet switching network. For our case, flow control is not considered because it rarely occurs in the NC system [D1], and its effect can thus be ignored.

Concluding remark

From the foregoing it can be said that the latency in the XTX IOP is a complex set of factors that follow different statistics. To characterize the latency time, we use the remarkable phenomenon summarized by a theorem known as the Central Limit Theorem. This theorem can be stated for our purposes as follows; [B-4]

Let $X_1, X_2,\ldots,X_N$ be independent random variables with means $m_1, m_2,\ldots,m_N$ and variances $\sigma_1^2, \sigma_2^2,\ldots,\sigma_N^2$, respectively.

Then the pdf

$$Z \triangleq \sum_{i=1}^{N} X_i$$

approaches a Gaussian pdf as N becomes large with mean

$$\mu = \sum_{i=1}^{N} \mu_i$$

and variance

$$\sigma^2 = \sum_{i=1}^{N} \sigma_i^2$$

provided

$$\lim_{N \to \infty} \frac{\sigma_i}{\sigma} = 0 \ ,$$

for all i.
It is emphasized that the p.d.f's of the component random variables need not be identical.

Another important fact about Gaussian random variables that we draw on is that if X is Gaussian with parameters μ and σ^2, then

$$Y = \alpha X + \beta$$

is Gaussian with parameters $\alpha\mu + \beta$ and $\alpha^2\sigma^2$, [E-4], and the proof is given there.

These results allow an approximation of the IOP latency by a Gaussian distribution.
Later the average values of these parameters shall be calculated.

(ii) XTX Block Processor Latency

The latency time in the BP is measured from the time that a message called a work list entry (WLE) is deposited in the BP's message queue to the time that an outgoing WLE is posted in an outgoing message queue in the shared memory.

In this case the components of the delay are as follows;

- The packet layer processing time
- Data transfer time between the BP and shared memory - is part of the simulation.

From the considerations made in the analysis of the IOP latency, it is assumed that the BP latency also follows a Gaussian distribution. In the case of a non-dedicated router, it is possible to ignore the effect of the flow control of the higher layers as this would affect only the performance of those local higher level processes. The concern then dwells on mainly the

transfer time between the BP and shared memory, which can be estimated from the bus cycle time and from the frame sizes.

For the packet level processing, following factors must be taken into account the :

1) Routing mechanism

It is assumed that the network uses virtual circuits - although the datagram facility is supported, it is not considered here. For a virtual circuit operation, a path or a set of paths must be set up to transport a given packet to its destination. It can be assumed that the BP dynamically maps the routing path into entries in a routing table indicating to which outgoing link a given packet is to be directed. The BP must perform the framing into an X.25 format, which includes assigning network addresses, virtual circuit number/logical channel identifiers etc. If it is assumed that this information resides in the shared memory, then it is possible to estimate the 'route processing' time by read and write cycles. For example, the BP must read shared memory for the source address, destination address, VCI etc and perform the corresponding (negligible) computation and a corresponding write cycle. The routing process will require a path selection process based on some least-cost metric, such as shortest-path; this forms part of the computation time. If the routing tables are located in the shared memory, then the IOPs (for the XTX) can perform the path search. However, effectively the packet will only be transmitted after the routing calculation has been completed. Hence for convenience we assume that it is the BP which performs the routing table computations.

In determining the delay due to routing, there are two main considerations;

- the time to set up a call to the next node

- the time to compute the logical path that the packet will use

The XTX NC has X.25 ports for the ATX NCs and an X.25 port to interface to the main network. In the model presented here, the XTX NC can be seen as a kind of router, in which it isolates local traffic from the traffic destined for the main X.25 network. The time to compute the path depends on the routing algorithm used. It is assumed here that static routing is performed. Then further assume that the routing tables are built at the call setup stage and hence does not arise during the data transmission time window of observation. Then the main

operations relate to frame filtering using the routing directory, and the call set up time corresponds to sending a Call Request frame to the next node, and receiving a Call Accept frame from the remote node's data link layer. The call setup time can be estimated from the knowledge of the link data rate while the path calculation can be estimated from read/write cycles as mentioned before.

2) Process Scheduling

This factor is significant in the BP since it handles several processes for both the network and link layers. Processes go through 3 states as shown below, and the probability of a transition depends on several implementation details including the

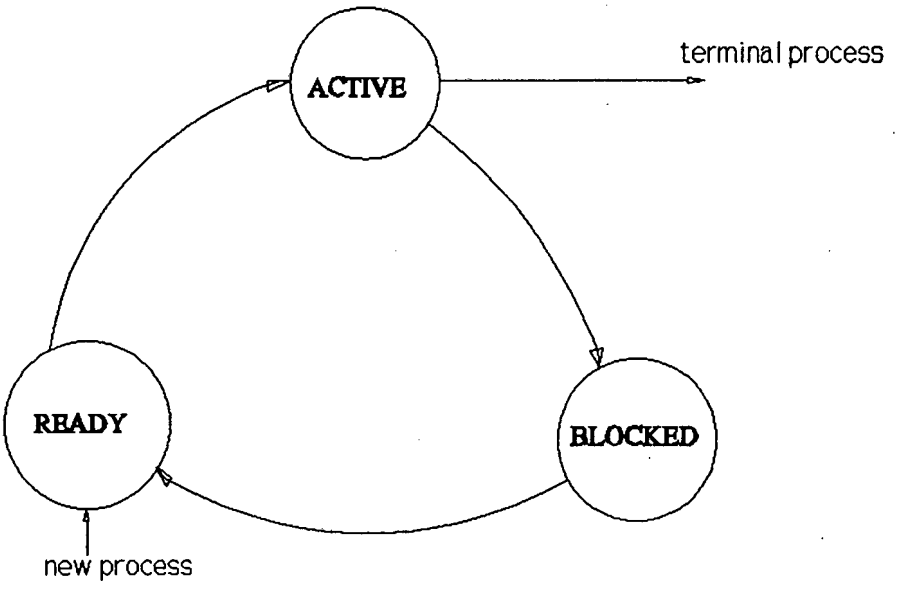


Fig.2-4 Process cycle states

scheduling policy used (preemptive or non-preemptive) and the size of the time slice if a preemptive scheme is used, and the number of concurrently scheduled processes. Assume a non-preemptive scheduling scheme for the BP and that there are no deadlocks in the system.

In general it is possible to construct a queuing model of the BP as in [E-5]. Let n be the average queue length of processes waiting to be served, and W be the average waiting time in the queue, and let λ be the average arrival rate for new processes in the queue. Then

during the time W that a process waits, $\lambda.W$ new processes will arrive in the queue. If the system is in the steady state, then the number of processes leaving the queue must be equal to the number of processes that arrive. Then use Little's formula

$$n = \lambda.W$$

which is true for any scheduling algorithm and arrival distribution.
We can use Little's formula to compute one of the three variables if we know the other two. We are particularly interested in an *estimate* of W, which contributes to the overall latency suffered by the BP in processing a frame.

This treatment conveniently assumes that all processes are run at the same priority. While this may be unrealistic, it gives a reasonable estimate for the older operating systems that do not multitask such as is used in the BPM. This approach has lumped together several unknown effects into an elegant model, and all the interactions with several intra- and interlayer processes are considered if their arrival rate (dependent on the number of processes!) can be correctly estimated.

3) Frame processing time

This quantity can be estimated from the knowledge of the frames that have to be exchanged (WLEs and Message Buffers), and from the bus cycle time. For the Message Buffer, only the control fields within the Message Buffer (6 bytes) are of interest to the BP. In addition to this, token processor cycles will be added.

Case 2 ATX NC Latencies

(i) ATX I/O Processor Latency

The X.28 protocol does not provide for error detection/correction and flow control. (The X-on and X-off signals may be used to enhance the X.28 connection). Therefore from the preceding section, the latency is due to the following

- the IOP buffering

- transfer time between IOP buffers and shared memory of WLEs and Message Buffers
- shared memory access time

All these are part of the simulation observation parameters.

(ii) ATX BP Latency

In the ATX NC the BP is assumed to perform all the link and packet level functions. These functions include packet assembly or framing towards the X.25 network, and disassembly towards the asynchronous terminals. Error correction and flow control are only performed on the X.25 side and not on the asynchronous side. For latency calculation, the factors to consider are

- network layer frame processing time, which includes framing
- process contention
- data transfer time between the BP buffers and shared memory

2.5 Description of the Hardware Blocks

(a) *Processing Elements*

The IOPs are Z-80A processors with a 4 MHz clock rate or 250 ns basic cycle time. The block processor is an Intel 8086 running at 8 MHz, ie basic cycle time of 125 ns.

In the 8086, the bus cycle is divided into 4 times referred to as T-states, where T = 1/clock frequency [E-2]. During each of these T states a distinct sub-operation occurs. In T_1, the address becomes valid and the system is informed of the type of bus cycle (SM, IOP_A, IOP_B, DRAM refresh, PIM), and the address is latched. At the end of T_2, the Ready input is sampled. If it is low, the processor will idle, repeating state T_3 until the ready line is high, allowing the accessing device to synchronize with the processor. In T_3, the read or write cycle commences and in a read cycle or in write cycle output data disappears. The 8086 Bus Timing - Maximum mode is given in Fig.2-5 [E-2].

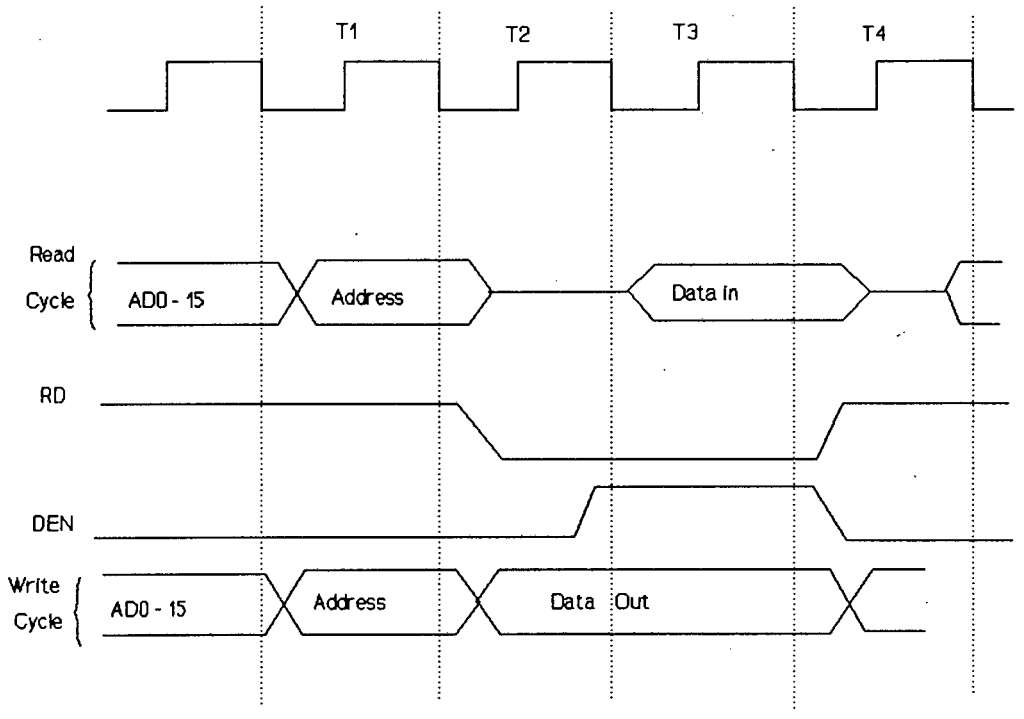


Fig.2-5 8086 Bus Timing - Maximum Mode

Without wait states, no cycle can be longer than four clock periods when the BP accesses its private memory. In the maximum mode configuration, due to the effect of the bus controller, the memory read cycle is committed after the start of T_2, hence a maximum access cycle is less than $3T$ cycles.

For the Z-80 a bus cycle takes 3 clock periods when a PE works with its private memory. The other bus timing considerations are as for the 8086.

When access to shared memory is required, some additional overheads occur. If the access time is defined as the duration from the time the processing element (PE) sends a request signal to the arbitrator logic, until it completes the use of shared memory, then,
Bus cycle = queuing time + memory setting time + processor time

Queuing time is the waiting time from the time the request signal is sent until it receives a grant signal. Memory setting time is equal to one clock period (in practice this time is much

less than a clock period), and allows the memory to set its address. Processor time is the time required by the PE in the memory access process. With no prefetching for both types of these PEs, a machine instruction takes from one to three/four clock periods. Take the upper limits of 3 cycles for the Z-80 and 4 cycles for the 8086.

Next consider the refresh operation. Assume that the DRAM Shared Memory is refreshed every 2 ms, requiring 128 cycles, and the refresh cycles are evenly distributed. Then a refresh cycle occurs every 2 ms/128 = 15 μs. If the refresh cycle is say 350 ns, this corresponds to 350 ns/125 ns = 3 wait states each time, and the time to access SM must be accordingly extended. For convenience the Refresh Logic is represented separately with an equivalent of 3 (BP) cycles. The arbiter will grant the request when a refresh request is made and no memory cycle is occurring or pending. If an access cycle is in progress, the arbiter must inhibit the refresh cycle until the current cycle is completed. The same process occurs if a refresh cycle is in progress and an access is requested. If the requests for access and refresh are made simultaneously, the priority followed is first BP, Refresh Logic, IOP_A, IOP_B and lastly the PIM.

Consider the case when all PEs and the Refresh Logic (RL) request the use of the shared memory simultaneously. The minimum and maximum access times can be computed as follows.

(i) The minimum bus cycle time occurs when a PE sends a request signal to the bus controller and it immediately receives a grant signal. The queuing time becomes zero, and

$T_{bus(min)}$ = 0 + 1 + 3 = 4 clock periods for the IOPs
= 0 + 1 + 4 = 5 clock periods for the BP
= 0 + 1 + 3 = 4 clock periods for the RL

For the 4 MHz IOPs this translates to 1 μs and for the BP this becomes 0.625 μs for the 8 MHz clock signal, and 0.5 μs for the RL which uses the BP clock.

(ii) The maximum bus cycle time occurs when a PE requests the use of the shared memory

immediately after it has just competed using the shared memory, while the other PEs and the RL also request an access to the shared memory. In this case the queuing time becomes 13 clock periods for each of the devices. Then

$T_{bus(max)}$ = 13 + 1 + 3 = 17 clock periods for the IOPs

= 13 + 1 + 4 = 18 clock periods for the BP

= 13 + 1 + 3 = 17 clock periods for the RL

≡ 4.25 μs for the IOPs

≡ 2.25 μs for the BP

≡ 2.125 μs for the RL

≡ 8.625/4 = 2.156 μs average.

If both the extreme cases occur with equal probability, the average bus cycle time when using shared memory is then (4 + 17)/2, that is 11 clock periods for the IOPs and 12 clock periods for the BP, and 11 cycles for the RL. The average bus cycle time then is calculated as follows;

T_{bus_avg} = (11 cycles x IOP_1 cycle time + 11 cycles x IOP_2 cycle time + 12 cycles x BP cycle time + 11 cycles x RL(BP) cycle time) /4

= (2.75 + 2.75 + 1.5 + 1.375)/4

= 2.1 μs

This is designated as the bus cycle time of the time shared bus in the system, as argued in Section 2.3.

(b) *Storage Devices*

The block processor is provided with a static private memory of capacity 64K x 16 bits with

an access time of 251 ns [E-2] to avoid wait states (and accounting for the memory setting time in the bus cycle), and it also has a capability to access a dynamic shared memory of 64K x 8 bits. For the shared memory we set the access time (for all the devices) as a uniformly distributed random variable with an upper bound of 495 ns and a lower bound of 251 ns. The IOPs have each a static private memory of 12K x 8 bits, with an upper limit access time of 720 ns, and they also have access to the shared memory.

(c) *Transfer Devices*

Each of the processing elements IOP_A, IOP_B and BP is connected to its private memory by a private bus Bus 1, Bus 2, Bus 3 respectively. Buses 1 and 2 can transfer 8 bits per cycle and have each a bus cycle time of 0.75 μs. Bus 3 can transfer 16 bits in a bus cycle time of 0.5 μs from the considerations already made above.

The time shared bus interconnects all the processing elements and the shared memory. It can transmit a maximum of 8 bits in a cycle time of 2.1 μs. All the Transfer Devices (TDs) use a First Come First Served scheduling discipline.

2.6 Mean Data Values

In this section, the analysis presented in Sections 2.4 and 2.5 is used to compute the means that will be used to describe the processes running in the BPM during the simulation.

2.6.1 XTX NC Mean Values

Each IOP has two serial I/O controllers which together provide four configurable serial I/O channels per IOP. Each of these channels supports baud rates ranging from 50 b/s to 19.2 kb/s [D-1]. For our purposes this range is limited to [300 b/s - 19.2 kb/s]. The calculations that follow are for Model II as given at the end of this document (see Section 2.7 later).

(a) Packet Arrival Rate for the X.25 links

The line rate can be taken to represent the transmission capacity of the link. However if it is assumed that message or frame arrival rate equals the baud rate then results for the limiting

case can be obtained. Thus baud rates of [300 b/s - 19.2 kb/s] represent arrival rates of [0.3 - 18.7 packets/second] corresponding to interarrival rates of [3.3 sec - 53.33 ms] between frames on each X.25 link. Assume a line capacity of 19200 b/s, and an operating point with a mean baud rate of 9600 b/s which corresponds to a minimum interval of 106.67 ms between frames per link. Take the frame length to be 128 bytes and assume a constant bit stream. Since there are 4 full duplex links per IOP then

Mean interarrival time = 1/(4*9600 ÷ (128*8*1000)) = 26.67 ms
Upper bound interarrival time = 1/(4*300 ÷ (128*8*1000)) = 833.33 ms
Lower bound interarrival time = 1/(4*19200 ÷ (128*8*1000)) = 13.33 ms

at each IOP.

(b) IOP Delay Time

(i) Delay due to error control

Assume a probability of frame error of 10^{-6}, and a frame length of 128 byte. Assuming a Stop-and-Wait protocol, the number of retransmissions required for correct delivery of the frame N_r is given by

$$N_r = 1/(1 - P_f)$$

This result is derived in the Section 3.6 where the NA is analyzed.
Hence for this $P_f = 10^{-6}$, $N_r \doteq 1$. A single reception means a delay equivalent to sending a NACK (4 bytes) to the terminal and for the corrupted frame to be retransmitted.

For a line rate of 9600 b/s,

Time to transmit a nack = 32/9600 = 3.3 ms

Propagation delay = 20 km (say) /2 x 10^8 ms = 0.1 ms

Frame transmission time = 1024/9600 = 107 ms

T_{out} (for this case) = T_{ack} = 3.3 ms

Processing time at IOP = 100 clock cycles (say) x 250 ns = 25 mic

Assume a send window of 1, that is Stop and Wait. Then average time for a correct

reception = 110.1867/(1 - 10^{-6}) (see Section 3.6)
= 110.1868 ms

Without errors the average time would have been given by

Frame transmission time + Propagation time + processing time

= 1024/9600 s + 20 km/2 x 10^{8} s + 25 mic
= 106.7867 ms

Hence mean delay = 3.4 ms

(ii) Data transfer time between the IOP and Shared Memory

The IOP captures frames directly to shared memory. The amount of data to be transferred consists of a 16 byte WLE and two 70 byte Message Buffers written to shared memory.

Bus cycle time = 2.1 μs per byte

SM access time by IOPs = 720 ns per byte

Hence transfer time = (16 + 140) x (2.1 + .72) microseconds

Mean data transfer time = 439.92 μs

(iii) Pure processing delay

Assume 1000 processor cycles to process the information (eg error checking, formatting wle, Msg Buff etc) for valid frame only.

Hence

Mean processing delay = 1000 x .25 = 250 μs

(iv) Delay due to process scheduling by the IOP dispatcher

The processes contending for the processor could be due both to the incoming packets and internal processes, all with Poisson arrival statistics. Hence the arrival rates sum linearly. If say there are 10 processes in the IOP queue awaiting execution at any one time, and in one second we have say 2000 process arrivals, we can use Little's formula from previous to obtain the average waiting time per process

$W = n/\lambda = 10/2000 = 5$ ms

The practical difficulty is making a good estimate for arrival rate of processes, and this will give the largest margin of error.

Not withstanding the limitations mentioned above, obtain a mean of the IOP latency per frame as

Total IOP Latency = 9.1 ms

(c) BP Delay Time

(i) Routing time delay

1. Directory lookup time

The lookup time consists of reading the contents of a routing table which are

- the source address
- the NEXT destination address
- the virtual circuit id : say 3 bytes, ie 3 * (2.1 μs bus cycle + .72 mic SM access time) = 8.5 μs.
- post 16 byte WLE to IOP

Transfer time = 16 bytes * 2.82 μs = 45.1 μs

Total directory lookup time = 53.6 mic

2. Call setup. X.25 is a connection oriented protocol, and as such call setup is a significant part of its overhead. Hence the need to capture and be able to account for the call setup latency in the simulation. Call setup (via IOP) (assume full duplex links and no errors) involves

WLE SM read + send 4 byte call request on X.25 and await Call Accept

45.1 μs + 32 bits (serial) x 104.1667 mic (X.25 link cycle time) + Tout

= 3.378 ms + $2T_{prop}$ + T_{ack} + T_{proc}

= 3.378 ms + 0.2 ms + 3.3 ms + 20 mic

Call setup time = 6.890 ms

(ii) Process contention time

If assume an arrival rate of 5000 processes/s (assuming an execution time of about 200 cycles per process) and a queue of 10 processes at any instant, then proceeding as before

Delay = 10/5000 = 2 ms

(iii) General frame processing time

We approximate the time to process a frame as the time to read a wle, process the 6 bytes (read + write + say 50 cycles overhead) of the Msg Buff, and write a WLE back to SM.

Processing time = 16 bytes x 2.82 μs + 2 x (6 bytes x 2.82 μs) + 50 cycles x .125 mic + 16 bytes x 2.82 μs.

= 113.4 microseconds

Total BP latency ≈ 9.00 ms

2.6.2 ATX NC Mean Values

In the ATX NC, each IOP has 4 ports which interface to the asynchronous terminals. A terminal user can transmit either interactive traffic which has a low rate of character generation, or send large files in which characters are passed continuously along the link, or a mixture of both type of traffic. In this exercise, the generation rate of interactive traffic from each terminal is considered to be 10% of the baud rate, with a mean traffic generation rate of 75% of the baud rate and a maximum possible generation rate of 100% of the baud rate, corresponding to large file transfers.

As an example calculation, consider an asynchronous terminal connected to the ATX NC via a 300 b/s X.28 link.

Interactive traffic generation rate = 0.1 x 300

= 30 b/s

Each Message Buffer is 70 bytes long. Time to build a Msg Buff at the NC is thus

= 70/(30/8)

= 4.7 seconds per link

Since there are 4 ports per IOP, time to forward a Msg Buff is then equal to

= 1.175 seconds

The mean traffic generation rate of each terminal is 0.75 x 300 b/s

= 225 b/s

Time to forward a Msg Buff = 70 x 8/(4 x 225)

= 0.62 seconds

At the maximum possible arrival rate of 100% baud rate, the time to forward a Msg Buffer is 0.47 seconds.

The calculation is performed for the other baud rates of 2400 b/s, 4800 b/s, 9600 b/s, 14400 b/s and 19200 b/s.

The objective of this calculation is to obtain a measure of the mean message arrival rate at the shared memory, the lower bound and upper bound of the arrival rate seen by the Block processor. In practice, these times are lower due the fact that, of the 70 bytes in the Message Buffer, 6 are composed of overhead to manage the 64 byte data area, as illustrated in Section 2.3.

2.7 Simulation Exercises

In this section, more details are provided concerning the measurement of the performance indices given in Section 2.2, at the beginning of this chapter.

2.7.1 ATX NC Response time

To evaluate the response time of the NC, simple models of a BPM serving [1 - 8] terminals are created. The terminals will independently generate requests and send them to the NC, which must process the request and send back a response. From this sequence of events it is possible to build a process flow with a Generate Process that will generate all the messages, an I/O Process that will be responsible for formatting the messages and transmitting them over the X.28 connection, and a Consume Process which will receive responses from the NC, at the terminal. At the BPM there will be corresponding processes of an I/O Process and a Server Process. Each of these processes can be executed on either of the IOP A or IOP B. Let the transmission time of a frame be the time taken by the transmitter to output the bits onto the channel, while the delivery time is the time that it takes for the last bit of the frame to reach its destination. By measuring the delivery time T_d and transmission time T_x, and computing the ratio T_d/T_x, it will be found that smaller messages of up to 6 bytes, suffer a heavier penalty than larger files of about 100 bytes, irrespective of routing strategy. Hence it is vital to optimise the response time for small messages. The time-critical response time is for small messages typically up to 6 bytes. Larger frames of up to 128 bytes are not considered for the analysis in this subsection.

This simple model is adequate for testing the response time of the NC in a variety of designs.

Some of the model parameters deal with

- The baud rate of the X.28 connection within the range 300 b/s to 19.2 kb/s.

- A request generation rate of between 1 to 10 messages per second. The message generation rate follows a Poisson distribution.

- The processing power of the NC is modeled by the latencies of the BPM - a small latency reflects higher speed processors and vice versa.

- The number of terminals being served by a single NC. This number is varied from 1 to 8.

- The terminals are modeled by a single processor operating at speeds between 2 - 8 MHz, an I/O port that can support baud rates of up to 19.2 kb/s, and local storage. A process running on the terminal wishing to communicate with the NC will deposit the message in the main memory, from which an I/O process will format and transmit it.

Observations

1. The effect of the baud rate of the transmission line on the response time of the NC as seen by the user is dramatic. Table 2-1 shows this relationship for typical baud rates, although speeds above 2400 b/s would be unusual.

Table 2-1 Records Response Time vs Baud Rate of X.28 connection

Baud Rate bits/s	300	1200	9600	19200
Response Time	613.1 ms	173.1 ms	44.8 ms	35.6 ms

2. If an attempt is made to increase the processing power of the terminal, minimal benefit is realized, as Table 2-2 shows. This is because messages are I/O intensive and do not consume much processing cycles. Of course full benefit occurs for local application processing functions.

Table 2-2 Records processing power of terminal vs NC response time

Cycle time of Terminal	1 μs	0.5 μs	0.2 μs	0.1 μs
Response Time	173.21 ms	173.11 ms	173.05 ms	173.03 ms

3. To illustrate the effect of increasing the processing power of the NC, the IOP and BP latencies can be reduced as shown in Table 2-3 and Table 2-4. The figures therein refer to a baud rate of 1200 b/s. The workload used in the simulation is biased towards IO functions, hence the effect of more powerful IOPs is more pronounced because they handle more I/O traffic than the BP.

Table 2-3 (a) Effect of reduced NC Latency

Percent of IOP Latency	75 %	50 %	20 %	5 %
Response Time	168.7 ms	164.5 ms	159.5 ms	157.0 ms

Table 2-4 (b) Effect of reduced NC Latency

Percent of BP Latency	75 %	50 %	20 %	5 %
Response Time	170.9 ms	168.7 ms	166.1 ms	164.7 ms

4. Also of great influence is the effect of increasing the number of users that the NC serves, see Table 2-5. The delay is not linear with the number of additional users because there are two I/O processors, as well as because of conflicts for IOP time by the multiple user processes. This is for the case when all users are communicating through the NC.

Table 2-5 Shows the deterioration of response with additional users

Number of users	1	2	3	4	5	6	7	8
Response Time	0.613 sec	1.039 sec	1.466 sec	1.839 sec	2.319 sec	2.746 sec	3.173 sec	3.599 sec

2.7.2 Investigating User-to-User Delay

This section investigates the issues affecting the delay between two users communicating over the X.25 network. Consider a simple subnetwork consisting of 9 NC nodes over which two particular users A and B are connected as shown in Fig.2-6. Each NC node is modeled as a

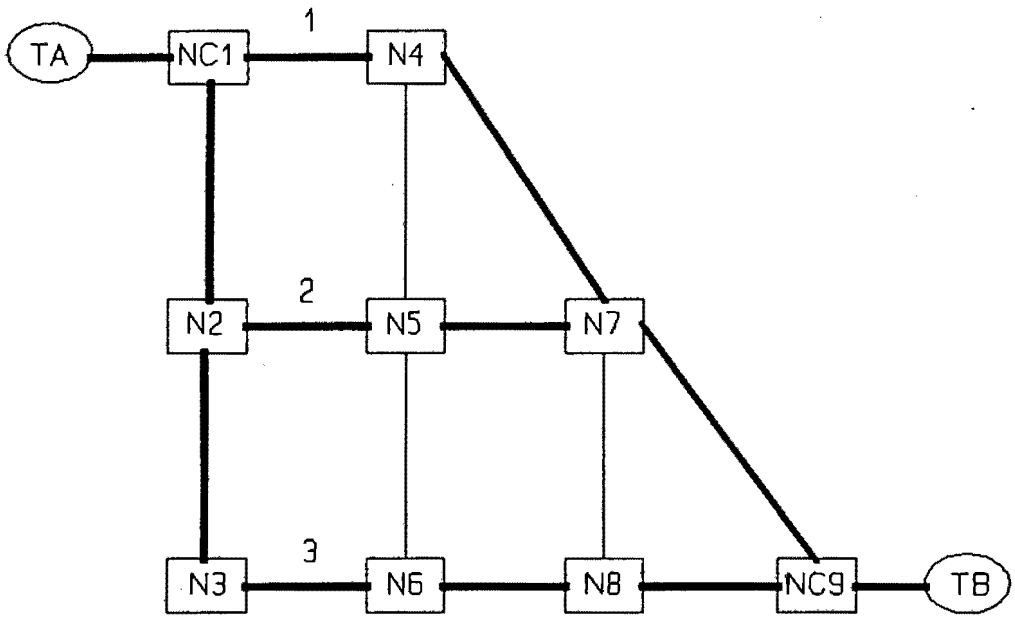


Fig.2-6 A simple network to investigate user-to-user delay

processing element where Node 1 and Node 9 (the edge nodes) each have a cycle time of 20 ms. This figure is obtained from adding the IOP and the BP latencies, adding the cycle times of the BP and the IOP, and then making an allowance for the NC I/O setup time. The other nodes each have a cycle time of 2 ms, and the user terminals 1 μs. All transfer devices support a bit rate of 9600 b/s, and the mechanism of data transfer is by means of message passing. Essentially, User A generates requests to User B in an exponentially distributed manner, with a mean of 1 message per second. User B replies to User A after a normally distributed 'access delay' of about half a second. The messages from A to B are typical queries of size 256 bits whereas the replies from B to A are of size 128 bytes. The extraneous messages are all 64 bytes each.

Two scenarios can be modeled based on Fig.2-6;

(a) Delay due to interference traffic

Three main routes are chosen for forwarding traffic between User A and User B. The response time is evaluated by varying the probability of selecting any of the paths

- **A** - Node 1 - Node 4 - Node 7 - Node 9 - **B**
- **A** - Node 1 - Node 2 - Node 5 - Node 7 - Node 9 - **B**
- **A** - Node 1 - Node 2 - Node 3 - Node 6 - Node 8 - Node 9 - **B**

Since the probabilities of route selection are predefined, this exercise uses static routing. Further it can be seen that the delay will correspond linearly to the number of links traversed in the path. In this exercise the aim is to investigate the effect of excess processing load on one of the nodes on each prescribed route. This aspect is modeled in the following way.

- on the first path nodes 4 and 5 are exchanging messages
- on the second path nodes 5 and 6 are exchanging messages
- on the third path nodes 8 and 7 are exchanging messages

In a variation of this scenario, investigate the case when this extra traffic uses the main links 1,2 or 3. The offending stations send messages to each other in a random manner, with the distribution of inter-generation times following Poisson statistics. The intensity of the traffic is varied from 1 message per second to 5 messages per second. These messages are pegged at the same priority level as the messages between A and B so that realistic interference effects can be studied. The size of the frames transferred between these stations is fixed at 64 bytes.

(c) Location of user information

The second scenario investigates the delay that arises if the NC Node 1 does cache some of the user information as opposed to the case where all user information is centralised at one node, say User B. Here let B be some higher level NMS. It is possible to investigate this aspect by considering arbitrary 'hit ratios' for the user NC Node 1. If a miss occurs, then User B must be searched, otherwise the user retrieves the required information from Node 1 immediately. This scenario simulates the user identification (NUI) validation; if not contained in the cache of the NC to which the user is attached, a search must be extended to

the NMS. One is thus measuring the response time for validation of NUIs.

Observations

1. Table 2-6 shows that for moderately low extra traffic in the network, the average response time for this network configuration is about 1.2 seconds. Because each node has a finite latency, it is clear that this response time is also position dependant in the network, and depends on the number of links traversed between the two users. Further the access time

Table 2-6 General Response Times for User A

Extra traffic off main links		Extra traffic on main links	
Interference Interval (s)	Response Time (μs)	Interference Interval (s)	Response Time (μs)
1	1 248 813	1	1 260 652
10	1 247 685	10	1 259 521
20	1 246 256	20	1 258 090
50	1 239 996	50	1 249 662
100	1 239 996	100	1 249 662

of the information located at B affects the response time directly. This network configuration has an average of 4 links/hops in the subnetwork.

2. In all the cases the utilization levels of the links are all below 20%, while the most heavily utilised nodes NC 1 and NC 9 are loaded at about 21%.

3. Table 2-7 shows the average response times for a network user given a certain chance of finding its user information in its edge node NC. If the chance of finding the user information in the edge node NC is very high, then the response time can be compared to that obtained in Section 2.7.1 above. As the chance of finding the user information at the higher level DM increases, the response time tends to that of the first case given in Table 2-6. These are the two extremes. Again the constraints are as discussed previously, including the line rates and the separation of the users. Depending on the delay sensitivity of the information, it may not matter whether the network user information is cached in the user's NC or centrally located in a remote location such as the NA - given a maximum response time of about 1.2 seconds.

Table 2-7 Location of User Information

'Hit Rate' at NC 1	Response Time (μs)	'Hit Rate' at NC 1	Response Time (μs)
95%	112 888	40%	664 774
90%	153 658	20%	919 547
85%	156 933	10%	1 055 250
75%	301 646	5%	1 094 827
60%	431 374	1%	1 231 548

4. For the case where there is no spurious traffic, the response time becomes about 1 second for this network.

This simple model reveals a number of serious considerations that usually need to be made in network planning and design, among them the bit rates of the links, the latency of each node and terminal, the sizes of the messages to be used, the location of network user information (i.e cached or not), construction of static routing tables (or allowing for dynamic routing), as well as how other network users' traffic affects the loading and response time of the overall network.

2.7.3 NC Throughput

To investigate throughput performance of the NC two basic types of models were created.

The first is a rather simplified model that is nevertheless powerful because the level of detail is sufficiently low to render the model flexible enough to tune, in order to observe the effects of interactions of various parameters that are at work within the system. It follows the same basic architecture as the real BPM, but is free of bottlenecks that would be considered in the real case such as error performance, flow control and other inter/intra layer process conflicts; i.e the latency of the NC is unrealistically small.

The second class consists of a more rigorous and more elaborate model of the XTX NC and the ATX. Because this model is built on real values, it is not easy to modify because a single parameter can be linked to numerous other values that are used in the system. It is used mainly as a measurement tool of the network parameters and the results obtained can therefore be related to the practical case within a small margin of error.

In both types of models the performance indicator of interest is the system throughput, which represents the number of packets served or processed by the block processor and retransmitted, modelled by the number of Message Buffers exchanged between the IOPs and the BPM in the simulation time. The Message Buffer and WLE are the basic data units common throughout the system. Normally the throughput would have been given as the number of packets processed per second. The graphs of results for the throughput that follow however are normalised values where the ratio of output to input messages for the BPM gives the same information over a suitable simulation period. This also allows some scalability in the plots.

We have deliberately made a direct relationship between the input data rate and the arrival rate (see Section 2.5-a), hence the throughput in terms of the actual number of packets transmitted in unit time for a given input data rate can be inferred if so desired. Assume that the input data rate corresponds to the output bit rate.

Process Workload

In order to construct the workload, a number of typical operations in the NC multiplexing environment were identified. These operations were grouped into functions, and simple process flows developed by building dependent relationships among the processes to form Network Software Modules. The general sequence of actions is as follows;

- An event is generated in the IOP, such as a message arriving from a terminal. Note that once a message arrives at the IOP, it does not matter to the BP whether it was 'generated' in the IOP or received from some DTE.

- The packet is deposited as a Message Buffer in shared memory, and a WLE is posted to the BP. While it is possible to deposit the WLE in shared memory, we use Network II.5's message passing feature to send the message directly to the BP, and make up for the shared memory access process.

- BP reads and processes the message in its buffer in shared memory and sends a message to either of the IOP or to a specific IOP, or to the X.25 port.

• The IOP will then read the Message Buffer and transmit the message as necessary.

The cycle is repeated for each type of I/O event. In this way it is possible to create a generic quantity of PACKET for any type of I/O event from the terminal, whose actual content is irrelevant. To balance out the BPM processes, some CPU intensive tasks are created and given an arbitrary number of processing cycles. The message sizes are 2 x 70-byte Message Buffers between the IOP and the BP, except in Model III where each message is strictly made equal to one 70-byte Message Buffer.

(a) Model I

Objectives

(i) In this model the aim is to observe the performance limits of the system in terms of throughput by using different traffic input rates. This relates to the number of I/O lines that are serviced by a single (central) BPM. Other relevant parameters are the utilization levels of the IOPs, the BP, common bus and shared memory.

(ii) Also to show effects of relative processor cycle times of the BP and IOPs on the throughput of the system in terms of packets handled by the system in the (steady-state, constant) simulation time. This will enable us to determine if the BP is a bottleneck in the system.

(iii) The BP latency time is also studied. It is related to the BP cycle time but represents the other system effects such as the error conditions, protocol delays etc. The aim is to find out the balance between the rate at which messages arrive at the shared memory and the rate at which the BP processes and forwards them.

The data used to drive the model has been given in sections 2.5 - 2.6, save that the messages used are small (70 bytes) to allow a fine tuning advantage.

The model is based on the following assumptions;

• errors are negligible in the system

- process contention has been ignored
- the presence of the refresh logic has been ignored

Observations

1. At the prescribed operating point as an 8-port Single Board Unit, operating at 9600 b/s or arrival rate of 38 packets/s, the IOPs are capable of servicing extra I/O lines. The throughput at this point is 100%, and will start to drop if the number of lines is increased to an equivalent of 3 FBUs as the following graphs illustrate. The loading level can be increased by either increasing the number of lines in terms of SBUs or by increasing the offered load on each line. In theory the Z-80A SIO can operate at data rates of between 0 and 800 kb/s.

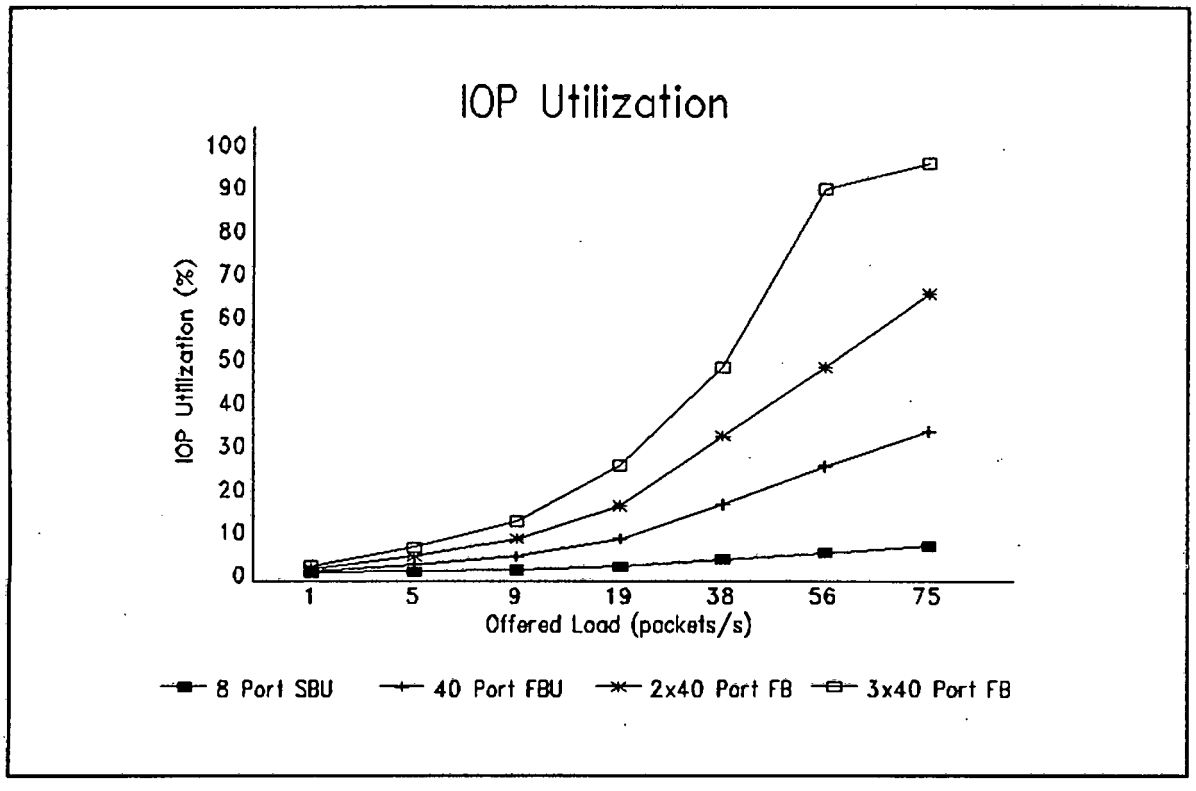


Fig.2-7 Shows IOP Utilization for Model I

Also, the design of the NC allows for a replication and extension of the I/O port capabilities of the BPM by use of the Passive PNAC Module (PPM). The hardware logic of the PPM is functionally equivalent to the BPM's I/O logic, and each PPM which is

attached to the BPM provides an additional eight ports of I/O. A maximum of four PPMs can be attached to a BPM, which gives a total of 40 I/O ports in the real system [D-1]. Practical considerations of modem baud rates will also limit the number of lines and baud rates that can actually be used.

2. The workload of the system is designed to be I/O inclined to simulate typical activity during normal operation. The graphs show the utilization figures of the main hardware elements, which at operating point are quite low, below 20%. The next case of interest is the FBU as it was originally provided for in the design, and therefore can be easily implemented and tested.

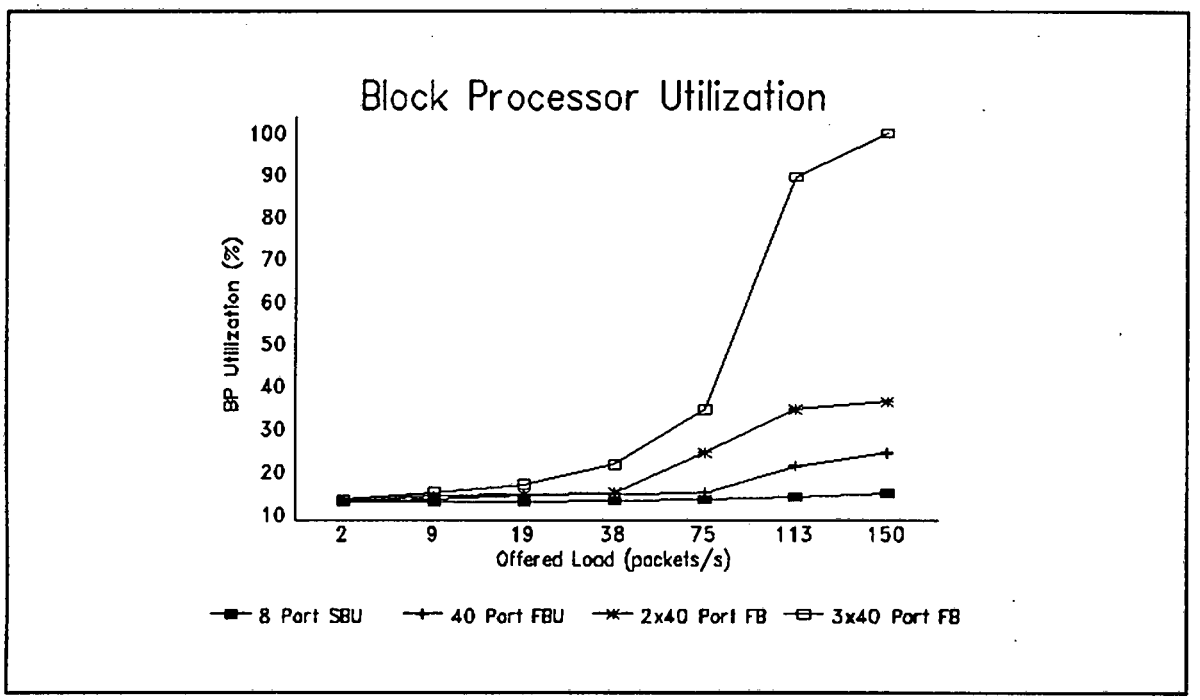


Fig.2-8 Shows BP Utilization for Model I

3. For the sake of completeness the processing power of the IOPs was increased, in order to observe the saturation point of the BP, since at the operating point there is no backlog at the Block Processor. The limit of a cycle time of the IOPs was chosen for practical reasons to be 0.02 μs (50 MHz), which is close to the current limit of the general purpose processors. With an IOP of clock rate of 33 MHz (a typical 80386 standard), the number of packets handled by the BP doubles but the throughput also falls accordingly.

The BP is powerful enough for application in an SBU configuration with no overheads.

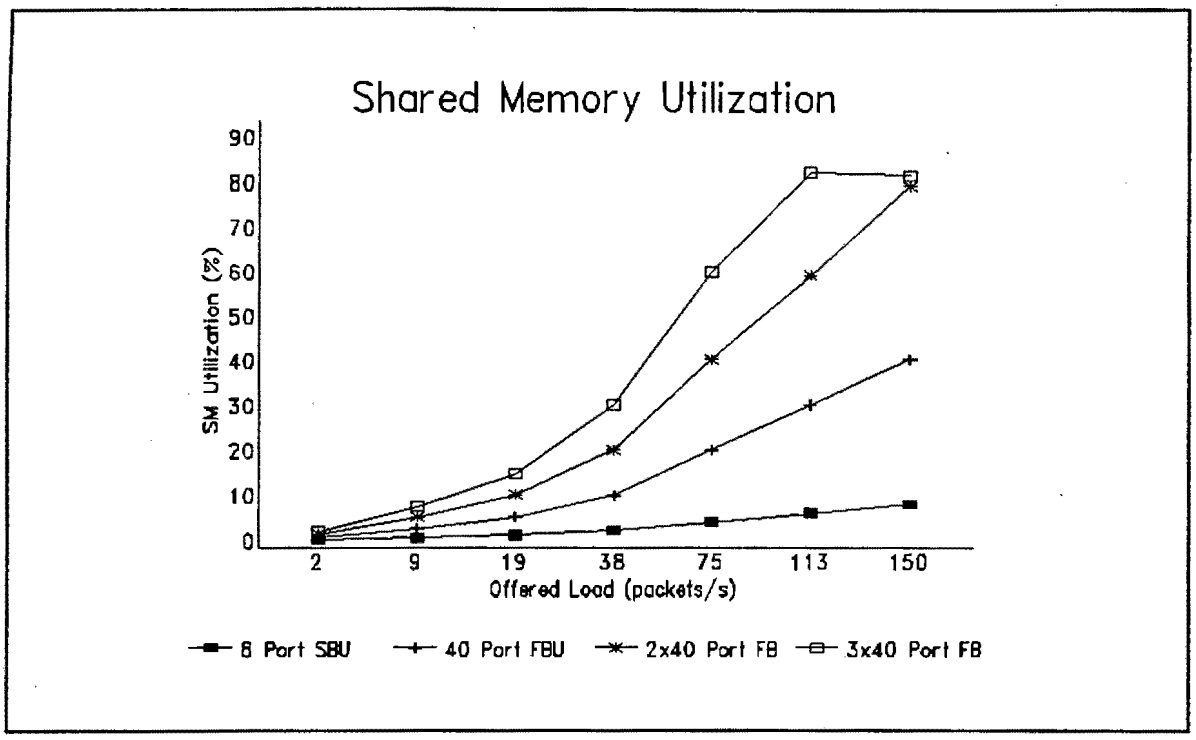


Fig.2-9 Shows Shared Memory Utilization for Model I

Model II

Objectives

The aims of the exercises in this model are to measure more accurately some of the generalisations observed by use of the simplified Model I. The current aim is to measure the actual utilization levels, the throughput, bottlenecks etc. The data values derived in section 2.6 are used to drive the model; and as many factors as possible are taken into account.

Observations

1. The graphs below show the performance of the NC at different loading levels, in terms of throughput, and hardware utilization figures. The same workload that was used for Model I is used to drive this model.

2. At the operating point {SBU, 75 p/s} the system achieves a maximum throughput of about

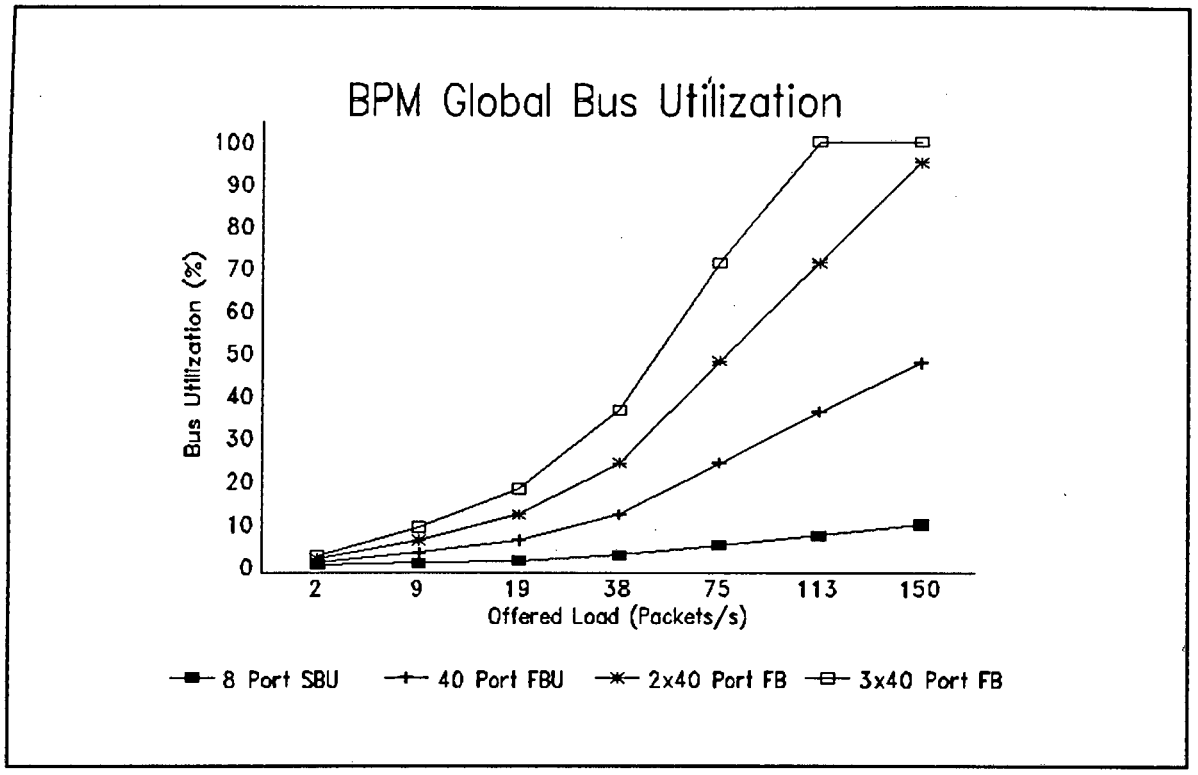


Fig.2-10 Shows Bus Utilization for Model I

75%. The graphs give the system response for various traffic intensities represented by the parameter of arrival rates. The arrival rate translates to a physical configuration in 8-port Single Board Units.

3. The bus and shared memory utilization figures are underrated because much of the I/O activity in the model has been characterised as DELAY, and represented as a processing overhead in processing cycles, and is reflected in the IOP and BP utilization figures. Note however that the throughput figures, which are the main target results are not affected by this approach. In Network II.5, processing instructions take about 25% of the time required by I/O instructions to execute [C-1]; thus if the target result was execution time, the delay components would have to be represented explicitly as such.

4. The BP service time can be varied to find the point at which 100% throughput can be achieved as shown. It is found that if the BP service time falls to about half of the IOP latency, then a throughput of 100% can be achieved. However it is not clear if this cost can be justified, as it requires a more powerful CPU for the BP or a shifting of tasks away from the BP or using a different protocol stack or IPC mechanism.

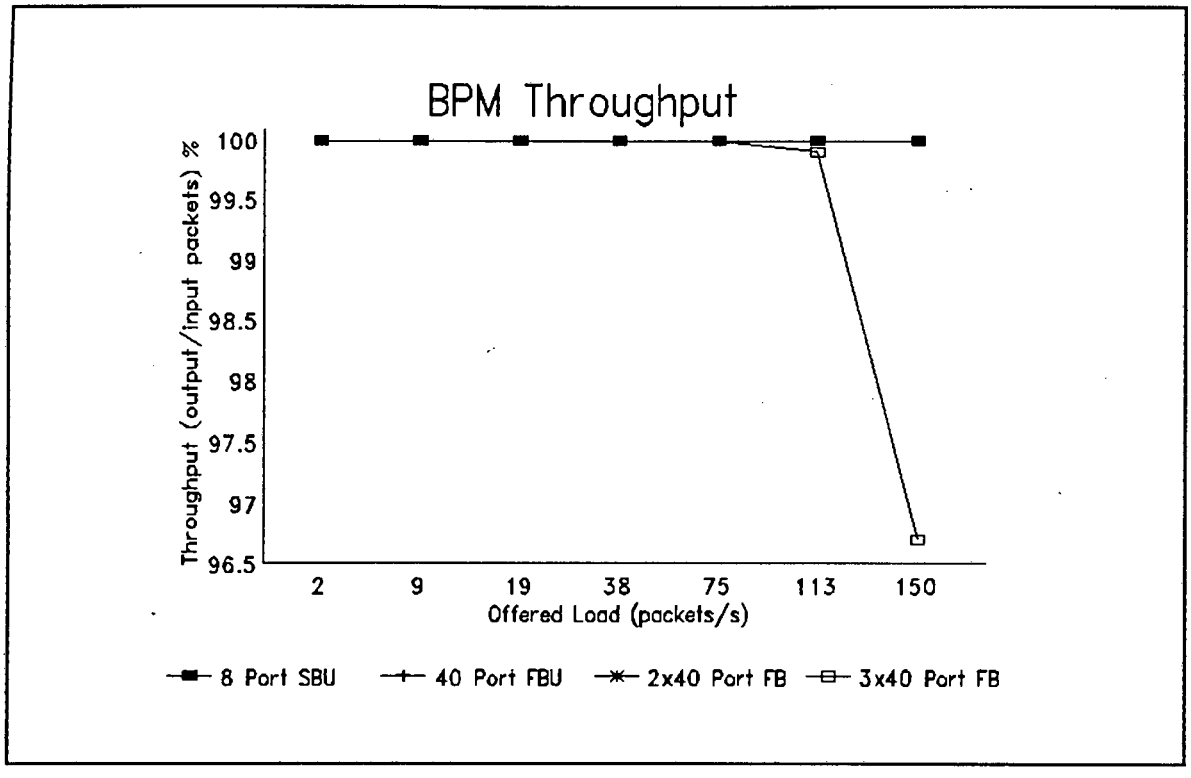


Fig.2-11 Shows the theoretical BPM throughput values - Model I

5. The utilization figures for the BPM are typically below 50% for an 8-port BPM and less than 75% for an FBU at an arrival rate of 75 p/s for the input lines, while the utilization of the shared memory is below 10%.

Model III

The objectives in Model III are to measure the hardware utilizations as in Model II, as well as to measure the throughput of the ATX NC. The computed values are used to derive realistic performance values for the NC.

Observations

1. The utilization results indicate the level of loading for both the standard 8-port Single Board Unit and a Two Board Unit NC which provides 16 I/O ports. It has been assumed that the design of the BPM allows the added SBU to approximately double the load to the NC system, with the effect that for the IOPs, the traffic generation rate has been doubled on that of a Single Board Unit NC.

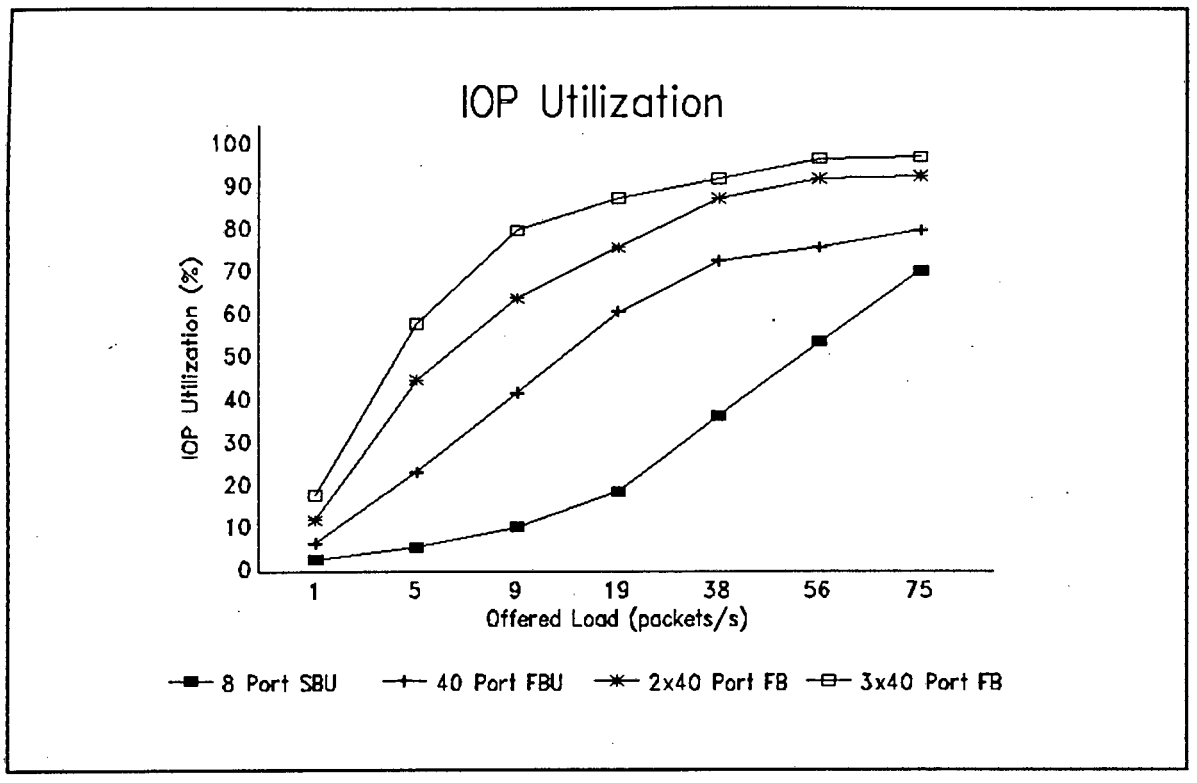


Fig.2-12 Shows the IOP Utilization for Model II

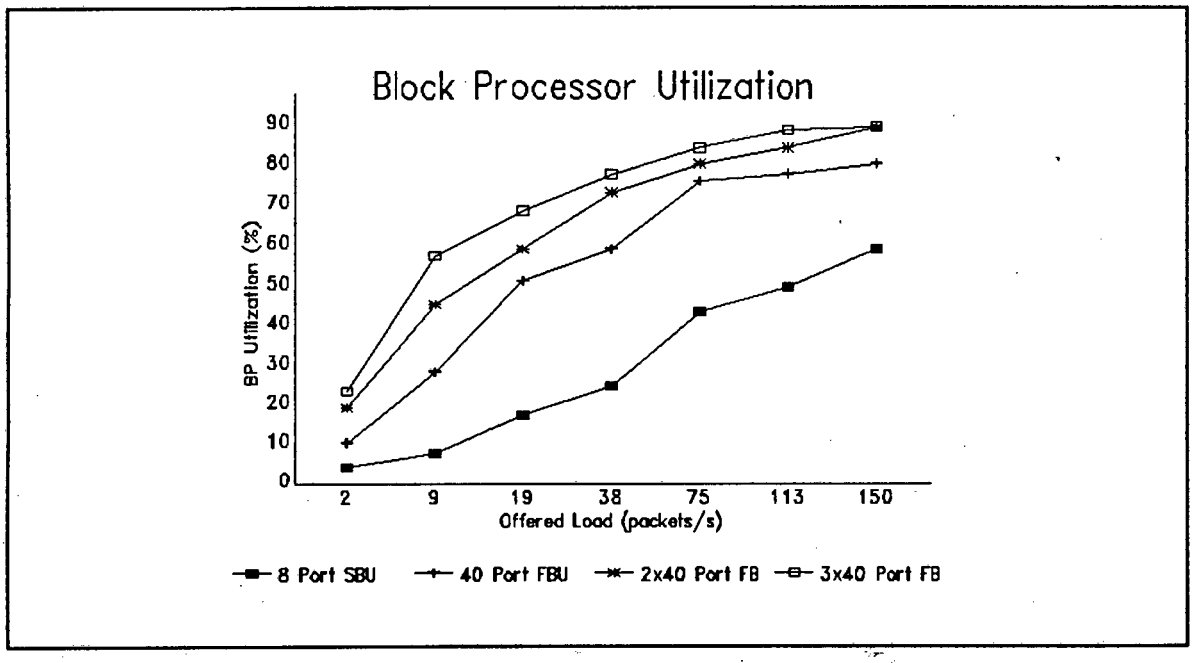


Fig.2-13 Shows the BP Utilization for Model II

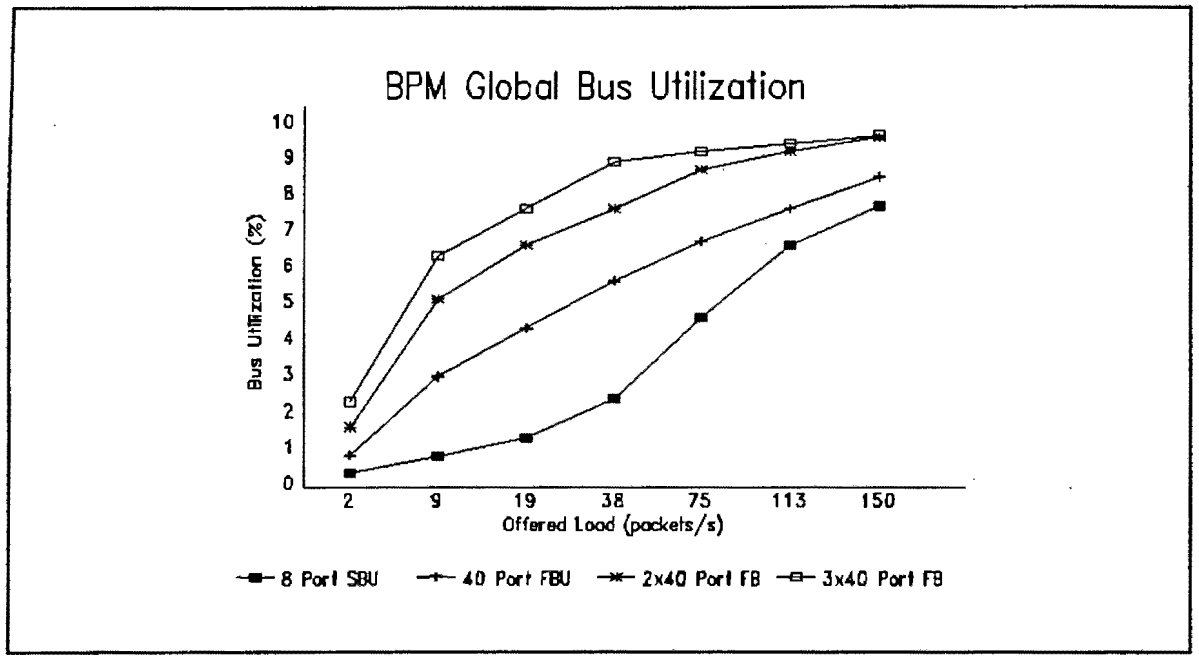


Fig.2-14 Shows Bus Utilization for Model II

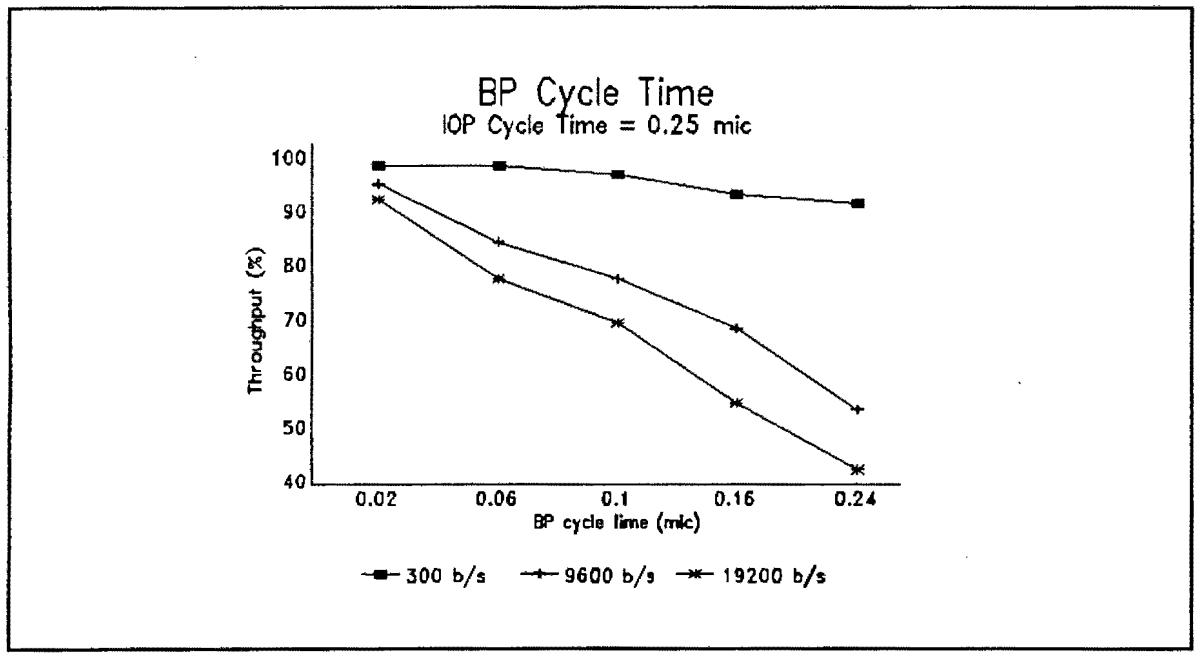


Fig.2-15 Shows the effect of changing the speed of the BP Model II

2. From the graphs Fig2-13 to Fig 2-21, it can be seen that utilization levels are low for the BP, IOP and Shared Memory for a Single Board Unit NC. These results confirm the observations made in Model I, namely that the utilization levels for the BP become significant for baud rates of over 9600 b/s for an SBU NC, and at around 7200 b/s for

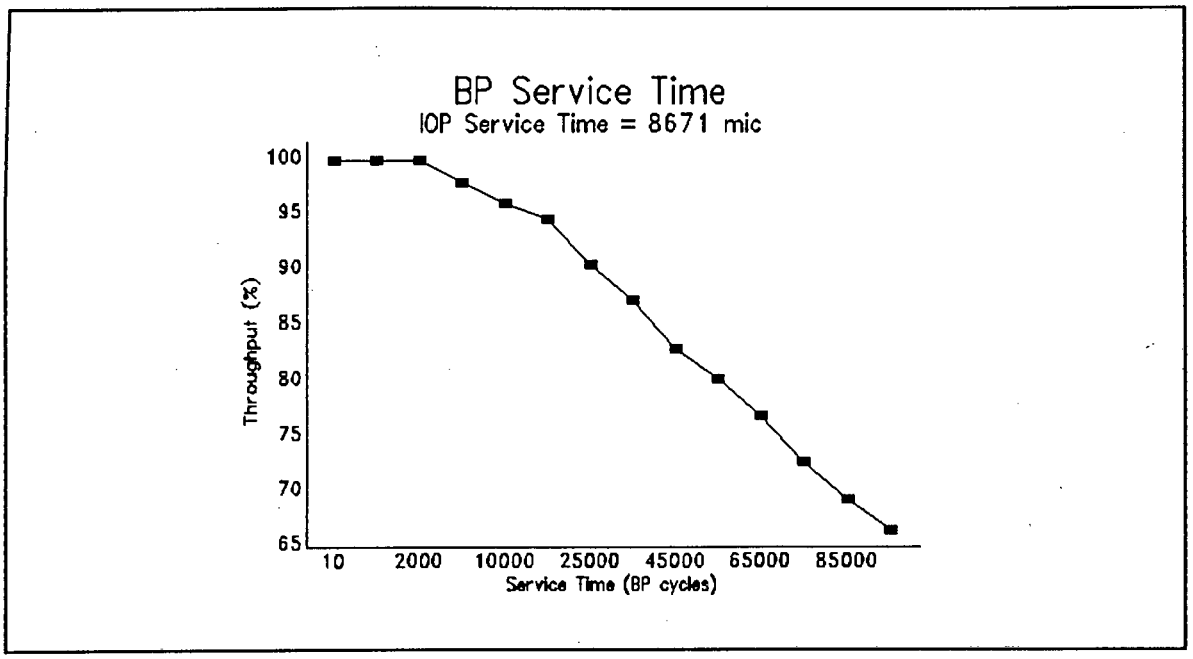


Fig.2-16 Shows the effect of the BP Latency on throughput Model II

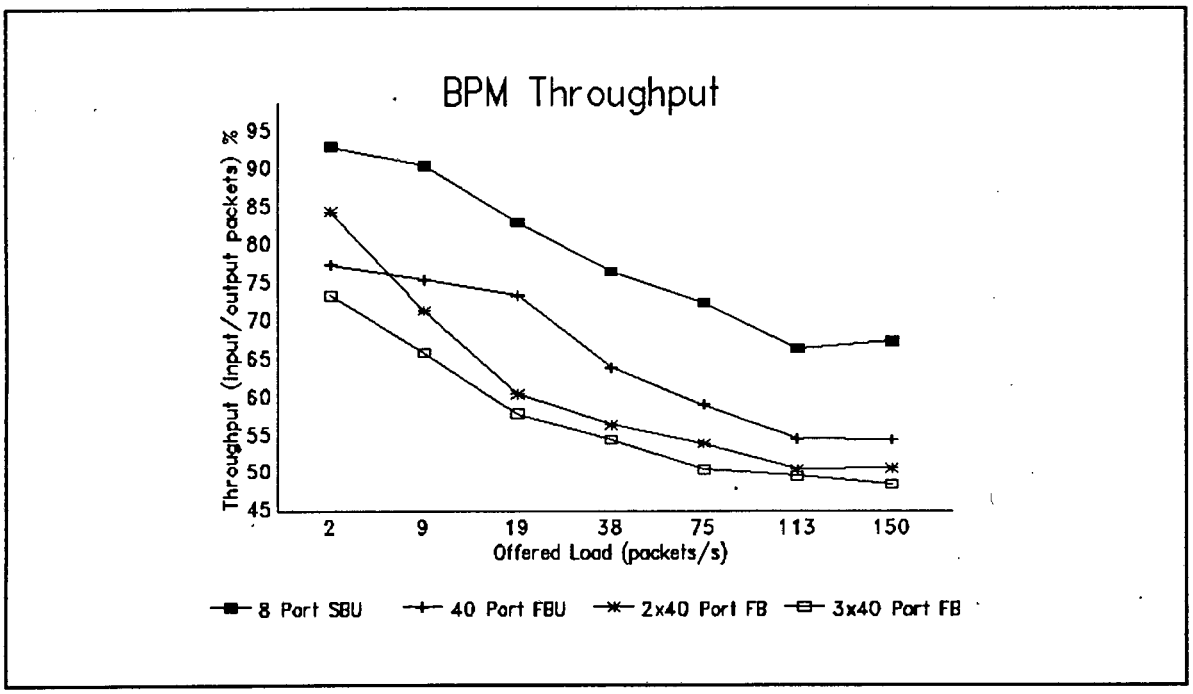


Fig.2-17 BPM throughput - Model II

a 2-Board Unit NC.

3. The IOP and shared memory utilizations are also below 15% for an SBU NC for baud

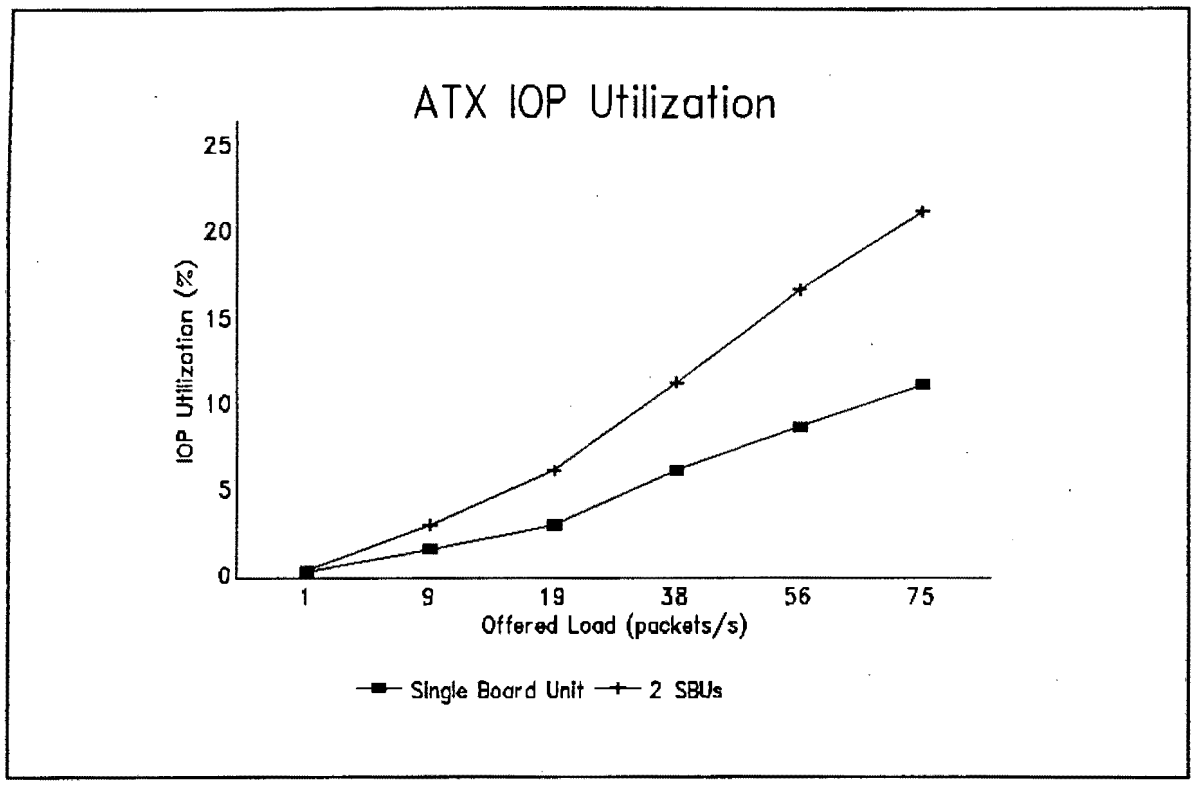


Fig.2-18 IOP Utilization Model III

rates of up to 19200 b/s. For a 2-Board Unit NC, the utilization of the IOPs is slightly less than twice that for an SBU at the corresponding baud rate. It is significant that there is no congestion in the Shared Memory because the Shared Memory is a potential bottleneck in the IPC between the IOPs and the BP, and could contribute to the degradation of throughput as it has been defined previously in this document.

4. For an SBU ATX NC, all messages forwarded by the terminals are retransmitted by the BP without congestion, i.e if the BPM receives 1000 characters per second, all the 1000 characters are processed and forwarded in one second. The BPM can sustain a throughput of 75% at 19.2 kb/s or at an offered load of 150 packets/s. If a second SBU is added, the NC can sustain a throughput of 75 p/s without congestion. The throughput starts dropping after 75 p/s, and at 150 p/s the throughput achieved is just over 70% of that achievable with a single board unit NC.

In summary, it can be said that when configured as an 8-port SBU NC, the ATX does not

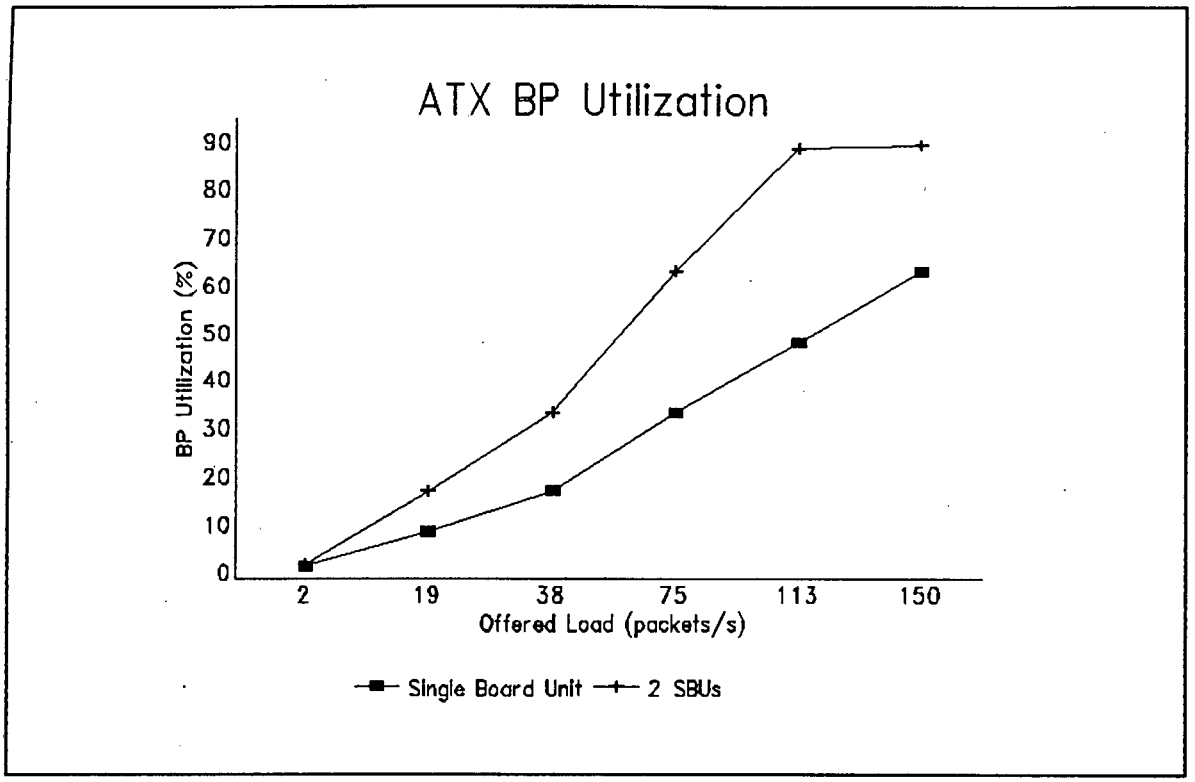


Fig.2-19 BP Utilization Model III

suffer from any traffic congestion. By including a second SBU, it has been shown that the throughout can degrade as more than 8 ports are added.

2.8 General Conclusion

In Section 2.7.1 some simple exercises to show how we can measure the response time of the NC with respect to the terminal users were done. It has been shown that for the purposes of exchanging information with other users over the network, it is not necessary to have expensive (fast) terminals as the gain in response time performance is nominal. A similar observation was made regarding increasing the processing power of the NC. However, bearing in mind the advent of high performance distributed applications, for the sake of local (protocol) processing, a high performance workstation is beneficial. It is also clear from there that the biggest bottleneck could be the communication line itself. Since the NC possesses capability to operate at high data rates, the user can use high speed modems and thereby realise the desired response time. It has also been shown that the response time varied

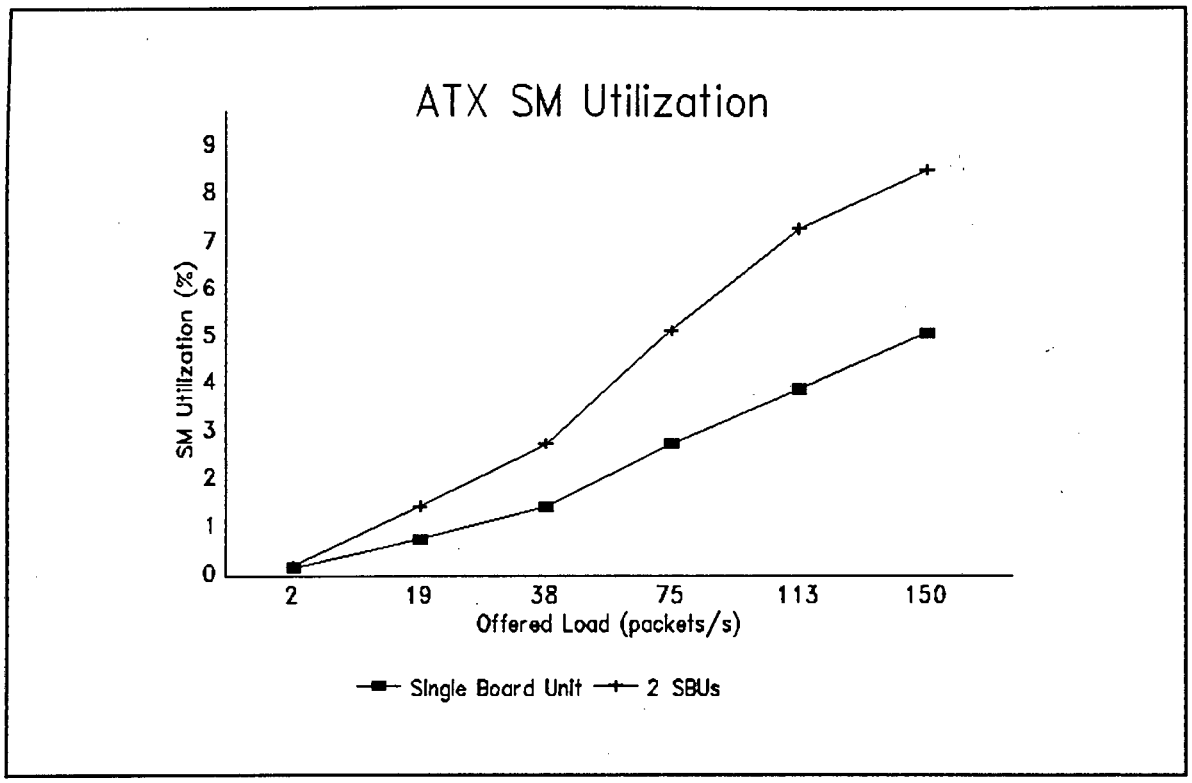


Fig.2-20 Shared Memory Utilization Model III

tremendously from the simplest case where a single User A connected to a local edge NC gets immediate response within about 50 ms using a 9.6 kb/s connection, to the other extreme case where 8 terminals are connected to a local NC, and one or more of these terminals communicates with a remote application across 4 or 5 nodes over a 9.6 kb/s link, in which a response is obtained in about 6 seconds. In general it has been found that the response time will depend on several factors including the bit rates of the links, the amount of traffic due to other users connected to the same network, the number of links traversed (and hence the processing power of the nodes on the path) as well as the size of query/response messages and the access time at the remote server.

In Section 2.7.2, a throughput performance evaluation for the NC was carried out . From these exercises it was shown that the NC design has no bottlenecks when operating as an 8-port Single Board Unit (SBU). The main contribution to the throughput bottleneck at higher loading levels is due mostly to what can be called protocol processing overhead within the BPM. It has been shown that little benefit is derived by increasing the sheer processing power

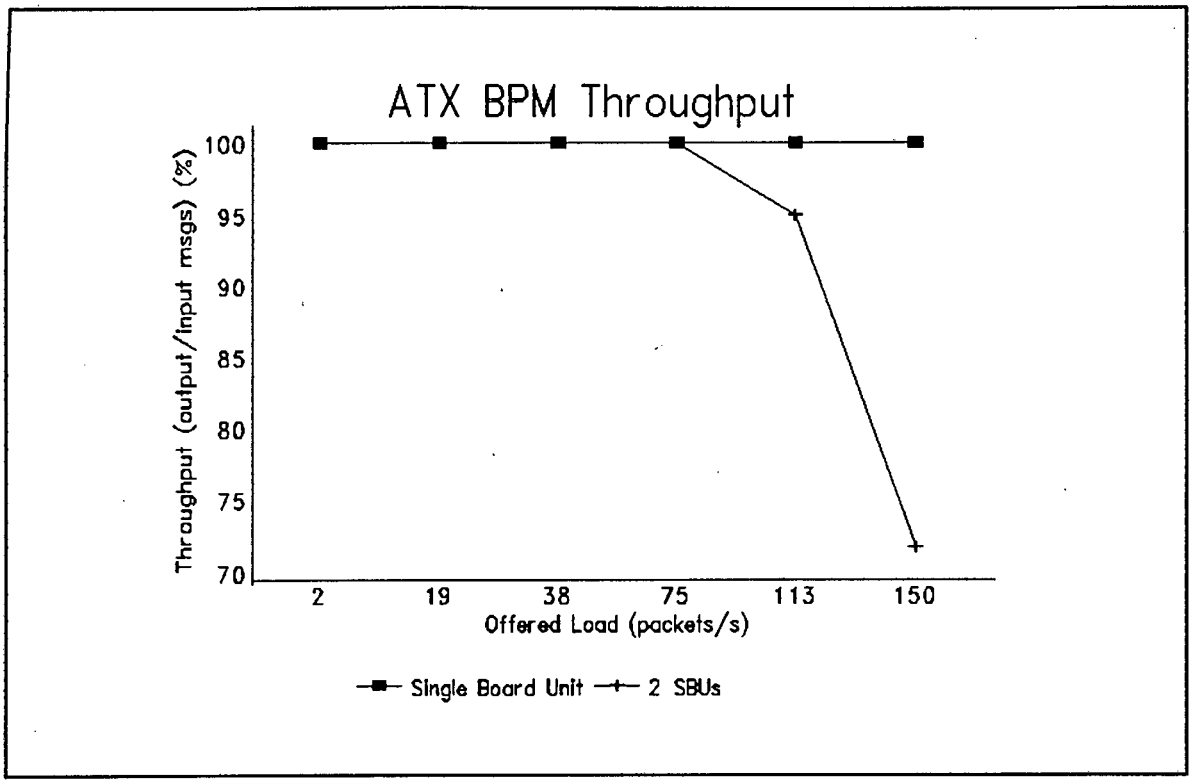


Fig.2-21 ATX NC Throughput

of the BPM to decrease this overhead. In other words it can be said that there is a software bottleneck. It is known that in communication systems, the protocol overhead can be quite heavy; for such functions as error and flow control, interprocess communication and other control mechanisms. All these events, and others like process contention for the processor etc, lower the throughput. In Model I we showed that the system is free of bottlenecks if these considerations are left out. Yet for a normal application a throughput of above 70% may correspond to a fair grade of service in a multiplexing environment such as this. Further in Model II we made excessive allowances for error levels given the current transmission technologies in coding and digital transmission using FEC techniques and fibre based media. Again the error margin in the value for the delay due to process contention and interlayer process interference is unknown, and the system performance is better or worse by this same uncertainty. If the Block Processor Operating System has latencies in its system primitives, this will again increase the uncertainty in estimating and characterising the delay due to the system processes, which from Section 2.5 can be much larger than delay due to the hardware components.

Chapter Three

Model of a Network Administrator

3.1 General System Description

The Network Administrator is used to load its associated NCs with the necessary operating software and to configure the NCs that perform Terminal Packet Assembly and Disassembly (TPADs) and Host Packet Assembly Disassembly (HPADs) on the network. It is also used to provide centralised network control, and to provide other support services to the NCs and the users such as accumulating statistics information about the entire network of distributed NCs in one centralised location. Management information is accumulated locally at the NC. The NCs can be of either the ATX or XTX type. In addition the NA supports network operator functions and offline functions. Operator functions include login control, configuration of NAs and NCs in domain, dump analysis, file transfer etc. Offline (and operator) functions are not considered for the purposes of this simulation.

Some of these functions execute periodically as a result of system timeouts (eg updating of system clock). Other functions are driven by network related events (eg reestablishment of contact with the NCs, loading and dumping of failed NCs). The NA has four, full duplex trunks which can be connected directly to the NCs or to an X.25 network. All NCs must be connected to the NA, either directly cascaded, or through the X.25 network. The relay or cascading of NCs allows the NA to access another NC through the services of the relay in between. In any case, one NA can control up to 80 NCs. A maximum of four separate networks may be serviced by an NA. Each I/O port may service a link with a maximum speed of 19.2 b/s, but the total I/O bandwidth cannot exceed 76.8 kb/s. In addition to the four communication links, the NA also provides for four local ports to service a status or error message report printer, a debug console, a network operator console and secondary storage.

Each ATX NC is connected either to a cluster of asynchronous terminals or to a host computer, as described in the treatment of the NC in Section 2.1., but this does not affect our present model and will thus not be considered further.

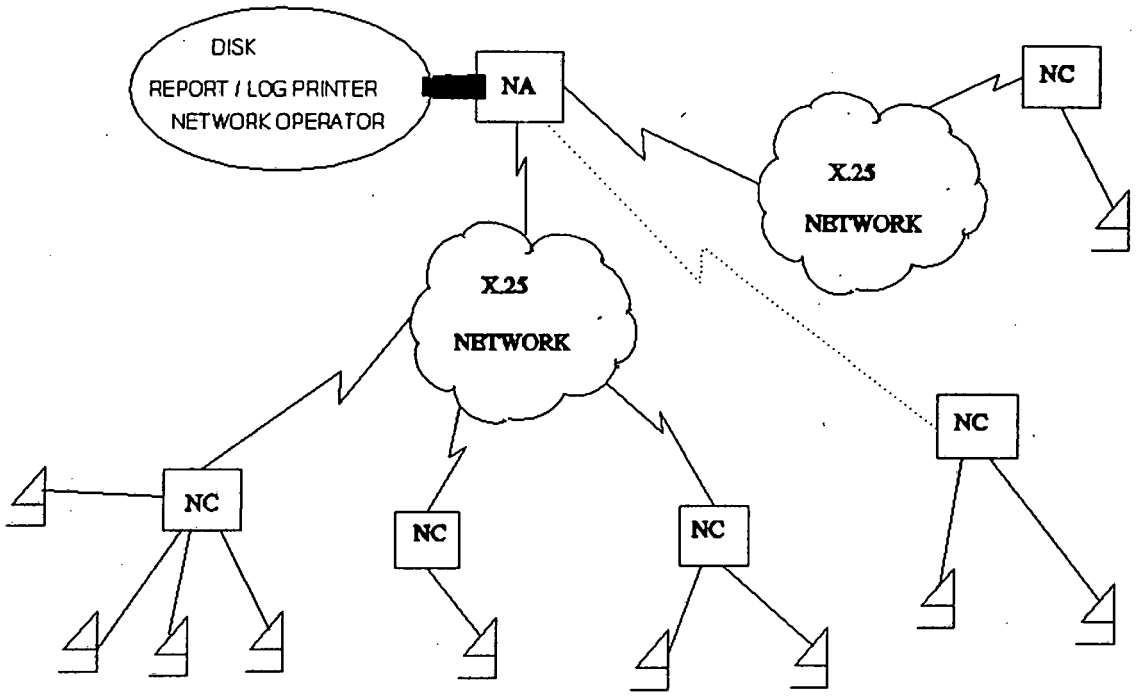


Fig.3-1 Network Configuration showing NA

3.2 The NA/NC Interface

NA applications like network control functions and other utilities are called subtasks, and run under the control of the Subtask Manager [D-1]. This is the primary interface for communication with NC processes. NA subtasks communicate with NC processes via support layers in each processor and through the X.25 network. The primary interface for an NC is the PNAC Manager. This module is the NC Transport Layer for NA/NC communications. In modelling the NA however we are only interested in the interaction of the NA and the NC at the link and the physical levels, and we restrict ourselves accordingly. The other higher level processes can be conveniently lumped and given an appropriate overhead at the link level.

The basic unit of communication between an NA and NC is a message. Messages from the NA to the NC are called commands, while messages in the opposite direction are called responses. X.25 data packets are used to transfer information to and from the NA and NCs.

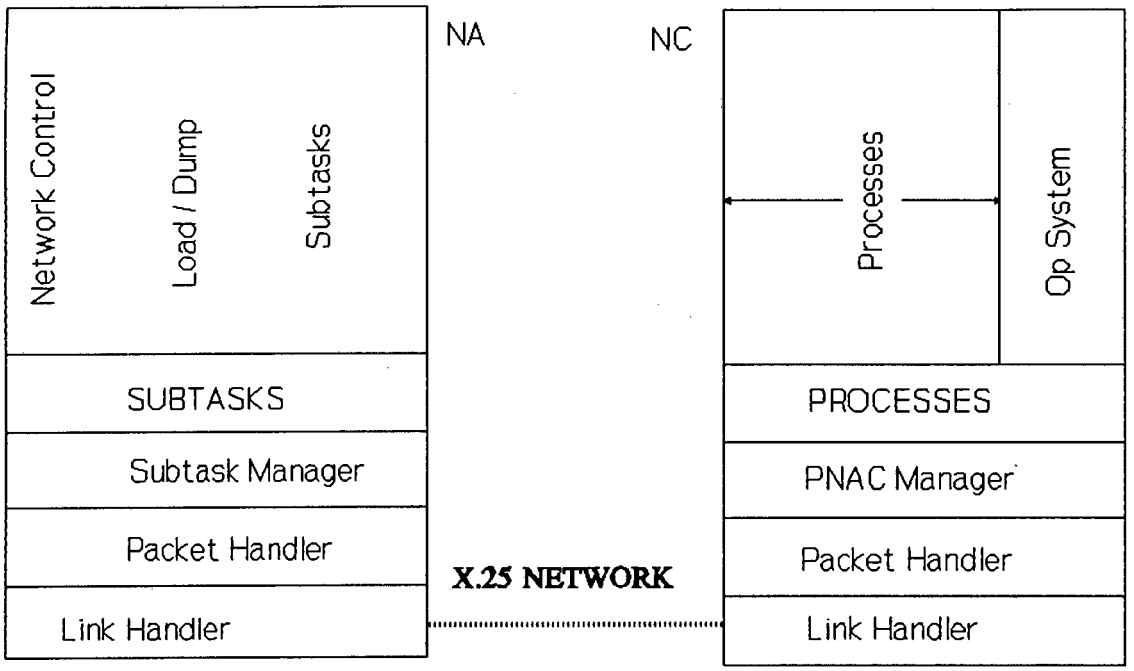


Fig.7-2 The NA/NC Interface

The user data area of the packets contain commands and responses. All commands and responses exist in system buffers in the BPM shared memory. They are the main vehicle in which data is transferred between processes in an NA or NC.

Fig.3-2 illustrates the logical configuration of the NA and the NC.This model considers mainly the link level and the physical layer; the upper layer processes are modeled by a suitable overhead at the link level.

3.3 Description of the Hardware Blocks

Processing Elements

The NA consists of a BPM that is configured as a multiprocessor system of three processors - the Block Processor, and two I/O processors. The physical attributes of each processing element and the BPM corresponds to that already described for the eight-port Network Concentrator. The operator console can be characterised as a processing element, modelled mainly by its 9600 b/s transfer device.
Two main scenarios are studied; NCs connected to the NA via an underlying X.25 network,

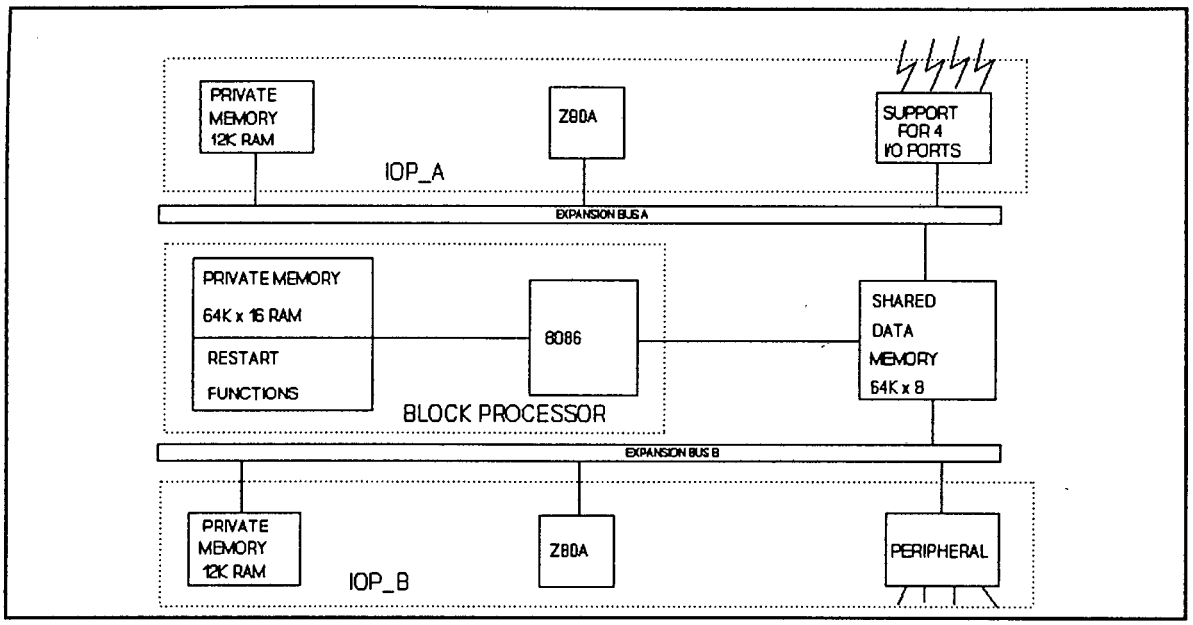


Fig.3-3 BPM Block Diagram

and NCs connected directly to the NA but strictly adhering to the X.25 protocol. At any given time a single NA can load a maximum of 8 NCs via the X.25 network and 2 NCs in a back-to-back configuration. For the network, if say a number of NCs distributed on each of the networks connecting to the NA had to be loaded, the NA performs load balancing and would load 2 NCs per network, and the bit rate per link would then be somewhat less than the maximum possible, or equivalently, it takes longer to load a given NC.

Transfer Devices

In addition to the 3 local buses between each processor and its private memory, there is the shared bus interconnecting all the processors and the shared memory. Also, four local buses are supported that provide service to a status printer, a network operator console, a message logging printer and secondary storage, all connected to the Block Processor. Connecting each IOP to an X.25 network are two full duplex links each capable of supporting a line rate of between 300 b/s to 19.2 kb/s, with the constraint that for full duplex operation the total link bandwidth cannot exceed 76.8 kb/s which effectively limits the line rate to 9600 b/s for the X.25 link during the dump/load process. Even in the case where the NA is connected directly to the NC, these same bit rates will be assumed. The baud rate of the debug console is always set to 9600 b/s.

Storage Devices

In addition to the three private memories associated with each processor, there is a hard disk secondary storage of capacity 40 Mbyte and a floppy disk drive for a 360 kb disk. The hard disk has an access time of 40 ms and the floppy disk can be modelled to have an access time of about 500 ms. The shared memory is configured as in the NC, and the access priority in ascending order is the Block Processor, the DRAM refresh logic, IOP-A, IOP-B. The printer devices connected to the BPM can be modelled as storage devices, accessed bit serially, and for which only write instructions apply of course. Since we are considering only the load/dump processes (Section 3.8 and Section 3.9), devices such as printers, operator consoles, and mass storage have no direct effect on the model.

3.4 Performance measures

The presence of an NA is not a prerequisite during normal operation of its subject NCs. Once the NC is loaded and running, its functionality is not affected by the presence or absence of an NA, although of course any information forwarded to the offline NA during this time will be lost. During normal operation the NA acts as a central recording site for statistics, billing and status information, which is not a time critical event. However if response times for these routine operations were to be required, then the results presented in Section 2.6 would be relevant, given that the architecture of the NA is similar to that of the NC. During abnormal operation, the main function of the NA is to return the failed NC to service without operator assistance with the correct software required for the NC in an expeditious manner. For this reason emphasis is placed on the factors that govern the operation of the NA only during the NC loading/dumping process.

3.5 Transmission Protocol

The error control scheme applied in the transmitter at the NA and NC is the Go-Back-N method commonly used for X.25 networks to improve throughput. Go-Back-N is a form of continuous RQ where the transmitter sends information frames continuously without waiting for an acknowledgment on a frame by frame basis. The common implementation though sets some value K (the send window) of the number of frames that can have outstanding acknowledgment. The scheme is illustrated in Fig.3-4 and Fig.3-5. For each of the network connections to the NC's under its domain, Go-Back-N error control is applied, at the network

layer level.

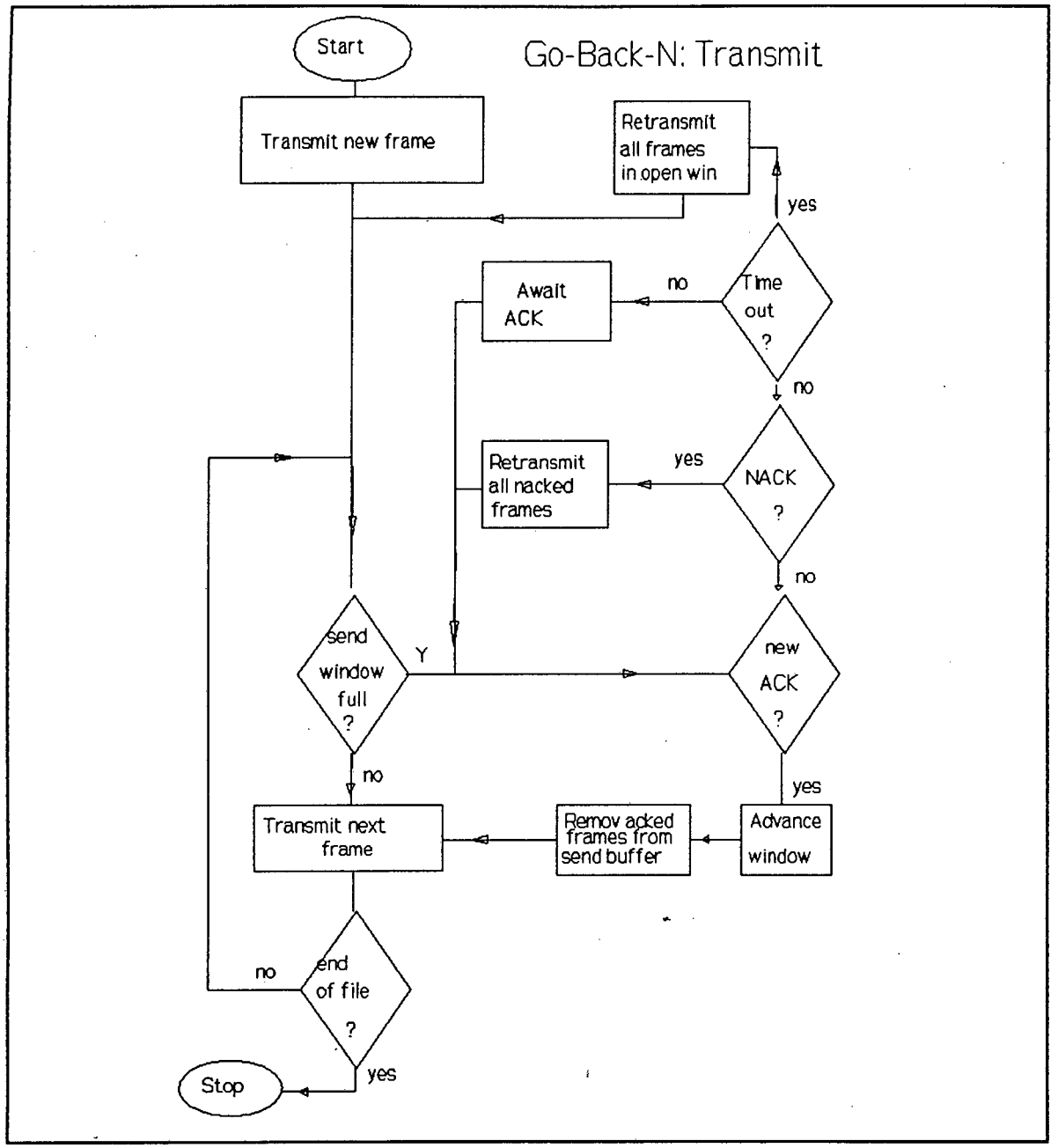


Fig.3-4 Go-Back-N Transmission Protocol

3.6 Traffic pattern

In modelling the traffic pattern it will be noticed that during the NC loading process frames are transmitted continuously one after another without waiting for an acknowledgment. In this

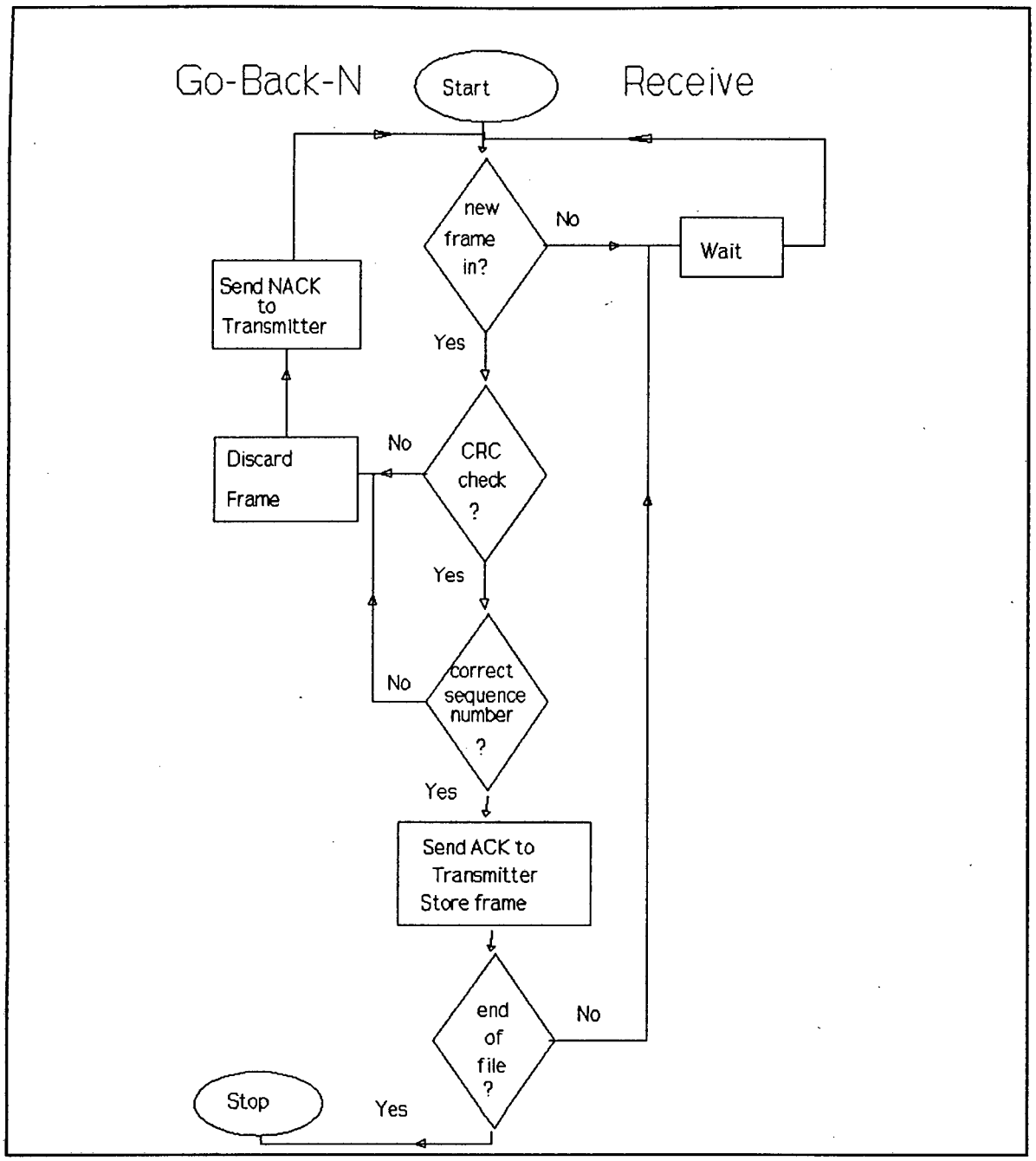


Fig.3-5 Go-Back-N Receive Protocol

case the Poisson process assumed for the NC network no longer holds. Factors that influence the distribution of the arrival of frames at the NC receiver include line noise, the effects of acknowledgement, the error control protocol, and the flow control scheme (fixed or variable window if used). Thus the service time of a packet (or a frame) will follow the distribution of the packet retransmission. Then if a packet must be retransmitted an average of N_r times,

and the distribution of N_r is known, this is also the distribution of the service time of a single packet. If the errors occur as bursts then we can approximate the distribution by the Rician model. [B-12] Alternatively, it may not be necessary to know the distribution of N_r, but to set up a range of error conditions and observe the throughput of the network for each error level. This is the procedure followed in Appendix E.

3.7 Mean Data Values

In this section the parameters that contribute to the delay associated with each packet of data that is handled in the BPM and in the overall X.25 network are identified. The approach used here follows that already developed in Section 2.5 for the ATX. In the main text, a baud rate of 9600 b/s is assumed. The actual simulation however makes measurements for both 9600 b/s and 19200 b/s, and results for both cases are presented.

In the BPM, the delay consists of the IOP and BP latency.

(a) IOP Latency

1. The processing delay includes the time for the exchange of WLEs between the IOP and BP, and some time for the BP to manipulate the Message Buffer (just the 6 bytes of control fields) which can be approximated by reading and writing the 6 bytes in SM plus some token processing cycles.

T_{proc} = 2x16(wles)x[2.1(bus cycle time) + 0.72(memory access time)] + 140 (Msg Buff) x 2.82 + 100 (cycles)x0.25

T_{proc} = 510 mic

2. Process contention delay can be taken as in Chapter 2 to be about 7 ms.

Then if delays due to error control functions are left out,

IOP Latency = 7510 mic

(b) BP Latency

During the load/dump process, there is no delay due to routing functions or flow control.

1. Processing delay is due to reading and writing two wles, reading and writing two 6-byte Msg Buff control fields and token processing cycles.

T_{proc} = 2.82 x 16 + 2 x 6 x 2.82 + 100 x 0.125

= 100 mic

2. Process contention delay can be taken to be about 7 ms as well.

Then the BP service time can be approximated to

BP Latency = 7100 mic

In the X.25 network the delay can be characterised as follows;

(i) Acknowledgement transmission time

The time taken to transmit an explicit acknowledgment (RR or RNR) T_{ack} is

T_{ack} = 6 bytes/9600 b/s = 0.005 sec.

(ii) Processing delay

For the purposes of error recovery the processing time consists of the time to perform error checking, which includes manipulating the 6 'supervisory' bytes of the X.25 frame. We can assume that the frame is processed entirely in the IOP and its private memory, the frame processing can be made up of cycles to read and write 6 bytes to private memory, i.e 334 μs.

(iii) Propagation Delay

Propagation delay = path distance/signal velocity

Using typical radii for a data network, we can have the values below.

Distance (km)	Tprop (microsec)
2	10
5	25
10	50
20	100
50	250
100	500
200	1000

(iv) Frame transmission time

T_I = frame size /baud rate

= 1080/9600

= 112.5 ms

Effective frame transmission time

Then the effective time required to transmit a frame becomes

$$T_T = KT_I + T_{out}$$

Appendix C gives the values of T_{avg} for various P_f and a K of 7, and d = [2 - 200] km. T_{avg} translates to an **effective frame size**. For example suppose there are transmissions errors and T_{avg} = 113.296 ms. Without errors the time required to transmit a frame is 112.5 ms. Hence under error the effective frame size becomes [(113.296 - 112.5)/112.5]*1080 = 8 bits extra, or a frame size of 1088 bits.

3.8 The Dump Process

In the event of a fault the software required for the correct operation of the NC may be lost and it is the function of the NA to reload the NC. In general an NC will be reloaded when any of the following hold.

(a) as a result of a reconfiguration or a software level change

(b) when the NC is force failed from the NA operator console

(c) on a power failure at the NC site

(d) an abnormal condition is detected in the NC code and the NC halting itself for debugging purposes.

When powered up, force-failed, or abnormally halted, an NC runs diagnostic tests, then enters a ROM state. The diagnostic tests last approx 45 seconds, and then the NC remains in this ROM state awaiting a call from its primary or secondary NA. Our simulations start after this point. When a call comes from the NA, the NC accepts the call and then responds with a Load Request message to the NA. On receipt of the Load Request, the NA will send a Dump Request to the NC, soliciting a dump of some NC control information. The 16 bytes of control information returned contains among other things, the diagnostics tests results, the error code at fail time, the NC ID number etc. Based on the contents of this control information, the NA decides whether to dump all data from the NC or whether to continue reloading the NC immediately. After dumping the NC Control Block the NA requests to dump further data. Code is not dumped, only the process data segments, operating system data, shared memory buffers, stack and IOP data is retrieved.

If the NA chooses to dump the NC, it determines which data was originally loaded, its location etc. For the dump process the NA creates a dump file whose size depends on the configuration of the NC in question. The Service Offer, Service Offer Accept, and Load Request and Dump Request and Response messages are passed as X.25 data on the now established management virtual circuit between the NA and the ROM-state NC.

The sequence diagram of Fig.3-6 forms a characterization of the workload constructed for the simulation model. We assume that the communication between the NA and the NC occurs via port 1A only although it is possible to dump/load via either 1A or 1B, or both. The reason is that we are more interested in obtaining the upper bound for the loading time.

3.9 Loading Process

After the dump process, the NA proceeds to reload the NC, (only) as a result of a Load Request from the ROM state NC. When starting a load, the NA consults the build files

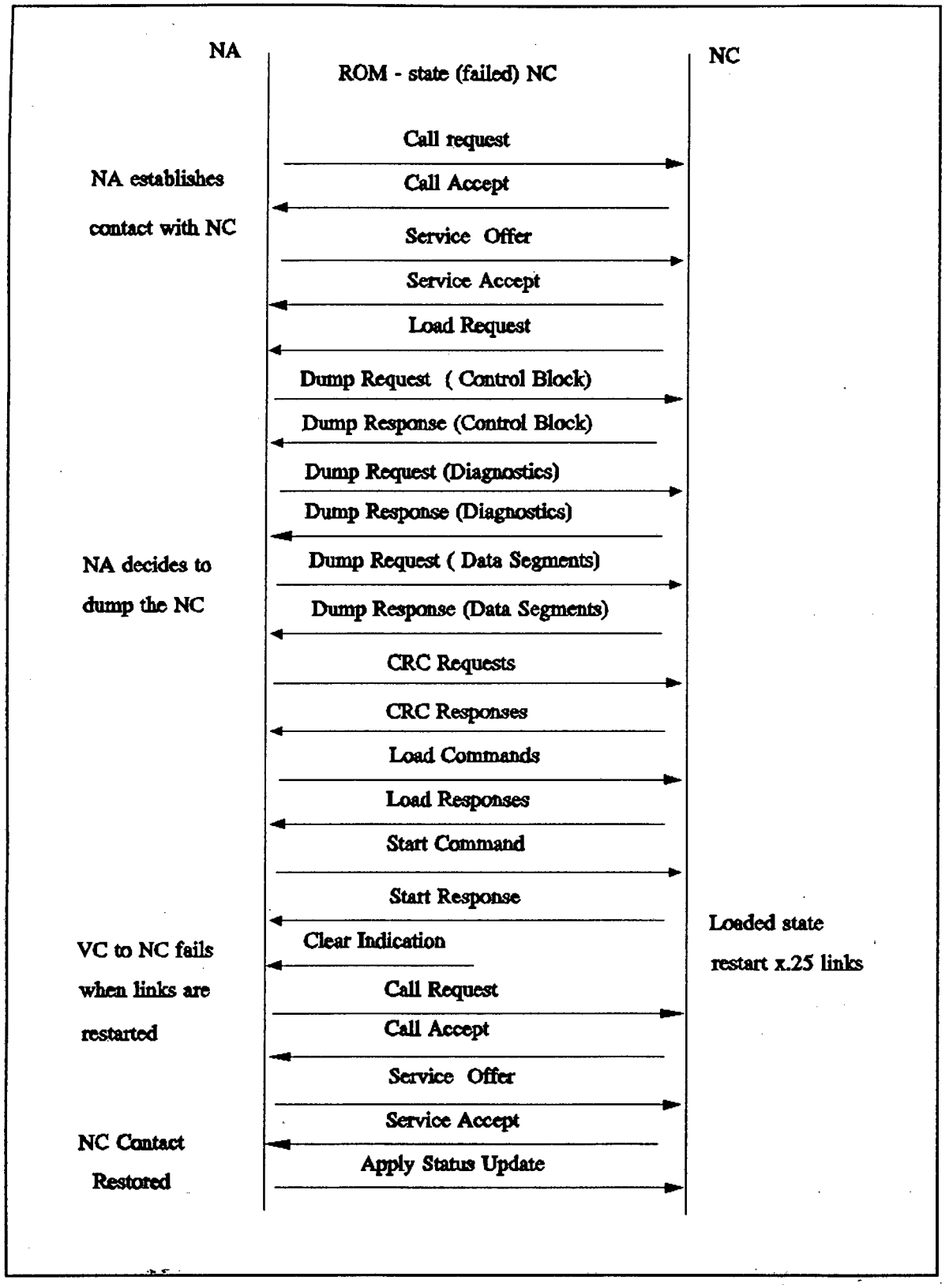


Fig.3-6 NC Service Restoration Sequence Diagram

corresponding to the build ID installed for the NC to determine which segments, and where, need to be loaded to the NC, from which it builds an initial control block. This frame is loaded by sending a Load Command to the ROM-state NC. The NC responds with a Load Response to each load command. The build file is used to continue the loading process as necessary. Checksum information is always exchanged between the NA and the NC during the Load of the BP Operating System on the existing information located at the segment target address. CRC Command (64 bytes) is issued by NA to determine the integrity of the NC BPOS code from the NC's CRC response (80 bytes). This is another source of variable loading time; if the CRCs match the segment is not loaded, decreasing the time accordingly. Data segments are always reloaded. We assume that the maximum amount of code to be loaded is equal to the total of BP, IOP-A and IOP-B private RAM memories and shared memory, ie an absolute maximum 203.5 kbytes. Since IOP code is destroyed by the diagnostic tests, IOP code is always reloaded - no CRC commands are issued before the load orders.

When all the information has been loaded successfully, the NA issues a Start Command (6 bytes) to cause the NC to switch to live RAM code. When a Start Response (also 6 bytes) is received from the NC, assume that the load process will have been completed, and the simulation is stopped.

Because of memory limitations the NA cannot load more than 8 NCs at any one time.
The case when the NC cannot be successfully loaded is not considered here.

3.10 Simulation Exercises

In the following, the various conditions under which the NC loading time was tested are described. Based on the sequence diagram of Fig.3-6, a process flow from which a workload was derived was developed. For example, a ROM-state NC will send a LOAD REQUEST to the NA after it has received and acknowledged a CALL REQUEST and a SERVICE OFFER. Each of the events in the sequence diagram is encapsulated into a software module with all the right attributes of priority, synchronization requirements, number of executable instructions etc. The modules taken together constitute the workload of the system. The two common cases of NA configuration are tested, namely the back-to-back arrangement which is convenient for testing purposes and exploring several scenarios as done below, as well as the normal operation of the NA over an X.25 network. This latter configuration is necessary

in order to explore the subtle effects of extraneous network traffic on the overall communication efficiency between the NA and the ROM-state NC.

3.10.1 NA and NC in a Back-to-Back Configuration

In this case there is a point to point X.25 link between the NA and the NC and the whole bandwidth is available to the loading process. In this configuration, a maximum of four NCs can be connected to the NA. The NA and the NCs are not aware of the nature of the network between them, as long as the X.25 interface protocol is adhered to, and clocking options are matched. This corresponds then to the simplest case from a modelling point of view, and yields the most optimistic results.

(a) The Send Window K

In the Go-Back-N protocol, if an error occurs and a NACK is returned by the receiver before the send window is full, the transmitter will continue to send frames until the send window is full and then retransmits. This has the effect of increasing the effective packet transmission time with respect to the receiver, and contributes to the delay in error conditions. The K values used are [7, 3, 1], and the simulation results for P_f values in [10^{-7}, 10^{-1}], corresponding to low/moderate to high BER [10^{-10} to 10^{-3}] are presented. The tables below are a representative case for a distance of 50 km between the NA and the NC in a back-to-back configuration, and the load time is in seconds. For conditions of high BER, the lower the K value, the faster the loading because in that case, efficiency is higher. The opposite is true for conditions of low bit error rate.

Table 3-1 Send Window Value vs Loading Time: 9600 b/s

K = 7		K = 3		K = 1	
Pf	Load Time	Pf	Load Time	Pf	Load Time
1E-1	335.82 s	1E-1	254.30 s	1E-1	213.56 s
1E-2	205.07 s	1E-2	197.78 s	1E-2	194.03 s
1E-3	193.36 s	1E-3	192.67 s	1E-3	192.34 s
1E-4	192.17 s	1E-4	192.17 s	1E-4	192.17 s
1E-5	192.00 s	1E-5	192.00 s	1E-5	191.00 s
1E-6	192.00 s	1E-6	192.00 s	1E-6	191.00 s
1E-7	192.00 s	1E-7	192.00 s	1E-7	191.00 s

Table 3-2 Send Window Value vs Loading Time: 19200 b/s

K = 7		K = 3		K = 1	
Pf	Load Time	Pf	Load Time	Pf	Load Time
1E-1	168.26 s	1E-1	127.40 s	1E-1	107.02 s
1E-2	102.60 s	1E-2	98.86 s	1E-2	97.00 s
1E-3	96.88 s	1E-3	96.41 s	1E-3	96.38 s
1E-4	96.30 s	1E-4	96.30 s	1E-4	96.29 s
1E-5	96.22 s	1E-5	96.22 s	1E-5	95.70 s
1E-6	96.22 s	1E-6	96.22 s	1E-6	95.70 s
1E-7	96.22 s	1E-7	96.22 s	1E-7	95.70 s

(b) Separation of NC and NA

A typical value added network would encompass a radius of up to 200 km or more if a wide area network is used. The effect of the propagation delay is accounted for in the expression for T_{out} as given previously, assuming a propagation speed of 2 x 10^8 m/s.

As the table below shows, there is a minimal effect on the ultimate loading time due to *d*. The table is a representative case for $K = 7$. Again the loading time is given in seconds.

Table 3-3 NA - NC Separation Distance vs Loading Time

d = 10 km		d = 50 km		d = 200 km	
Pf	Load Time	Pf	Load Time	Pf	Load Time
1E-1	335.65 s	1E-1	335.82 s	1E-1	335.99 s
1E-2	205.07 s	1E-2	205.07 s	1E-2	205.24 s
1E-3	193.36 s	1E-3	193.36 s	1E-3	193.36 s
1E-4	192.17 s	1E-4	192.17 s	1E-4	192.17 s
1E-5	192.00 s	1E-5	192.00 s	1E-5	192.00 s
1E-6	192.00 s	1E-6	192.00 s	1E-6	192.00 s
1E-7	192.00 s	1E-7	192.00 s	1E-7	192.00 s

(c) Size of the loadable memory segments

When an NC fails, depending on the mode of failure, and ultimately the analysis of the dumped control blocks and the CRC checks that are done prior to each non-IOP code segment load, it may not be necessary to load the entire SM, BP and IOP memory. For the moderate

error condition for the point-to-point link, the table below shows the loading times obtained.

Table 3-4 No. of Memory vs Loading Time

No of Frames	Load Time	No of Frames	Load Time	No of Frames	Load Time	No of Frames	Load Time
1628	192.00	1180	139.34	732	86.67	284	33.91
1564	184.46	1116	131.82	668	79.16	220	26.38
1500	176.95	1052	124.32	604	71.66	156	18.85
1436	169.42	988	116.79	540	64.13	92	11.35
1372	161.90	924	109.26	476	56.60	28	3.83
1308	154.38	860	101.73	412	48.98		
1244	146.86	796	94.20	348	41.44		

In the above table assume a modulo (8) X.25 network with frame size of 128 bytes, and the amount of data or memory blocks can thus be computed directly. The load time is in seconds, and we assume negligible line errors.

3.10.2 Loading over an X.25 Network

In this case the bandwidth of the X.25 link must be shared and the effective bit rate of the logical channel will be lower than 9600 b/s. Consider an X.25 link where there are several users (NCs) on the same domain communicating over the same network, then we obtain the table below which gives the relationship between the volume of "extraneous user" traffic and the loading time. For the other users assume a Poisson distributed traffic. To reduce the labour of model preparation, instead of increasing the number of users on the network, we simply increase the volume of traffic (arrival rate) from a fixed number of users and obtain exactly the same congestion effect. In this case, make the same assumptions as in Case I, in respect of K values, error levels, and the error control scheme. The amount of memory to be loaded is arbitrarily fixed at 203.5 kbytes which corresponds to 1628 frames of data to be transmitted for each NC. The Table 3.5 below is for a case of two NCs, hence the actual extra traffic on the link is double the rate show there.

In Table 3-4, the traffic intensity corresponds to the time between transmissions per NC node, for example if the two NCs each independently transmit packets at an average rate of 1 every minute, then the loading time for a single NC using the same 9600 b/s X.25 link is found to be about 243 seconds.

Table 3-5 Loading Time vs Extraneous Network Traffic

Traffic Intensity (int_arr_tim /nc node)	Loading Time (sec)	Traffic Intensity	Loading Time (sec)
10 ms	892.544	2 sec	214.822
50 ms	891.050	5 sec	200.530
100 ms	737.458	10 sec	196.050
200 ms	716.550	60 sec	192.641
500 ms	334.487	120 sec	192.214
1000 ms	243.831	240 sec	192.001

Information and Supervisory frames

As in the case for the NC, all interprocess communication within the BPM is by means of work list entries of 16 bytes and message buffers of 70 bytes. In the X.25 protocol, three types of frames are defined to handle information flow, supervisory and control signals, and responses to all of these: Information transfer format, Supervisory format, and Unnumbered format. The U format is used in the link connect/disconnect phase of the protocol as well as to provide extended sequence numbering if desired, and is used here only for Call Request and Call Accept frames [D-1]. On the X.25 link it is assumed that all I-frame messages are

Table 3-6 NA - NC Message Frame Sizes

Frame	Size (bytes)	Frame	Size
Call Request	3	CRC Request	8
Call Accept	3	CRC Response	10
Service Offer	6	Load Command	135
Service Accept	6	Load Response	6
Load Request	6	Start Command	6
Dump Request	8	Start Response	6
Dump Response	135	Apply Status Update	6
		Clear Indication	3

a standard 135 byte frame which comprises a 128 byte packet and a 7 byte framing overhead at the link and packet levels. This frame carries all the data to be loaded and the dump responses from the NC. Other control messages exchanged between the NA and the NC are assumed to be as specified by the X.25 protocol and according to the table above.

The High Level Data Link Control (HDLC) protocol [B-9] provides both a piggy-back feature with the acknowledgement function imbedded in an I-frame transmitted in the reverse direction (in the *N(R)* field), and a separate acknowledgment, the Receive Ready (RR) frame, that can be used to signal a positive ACK in the absence of an I-frame or to expedite the delivery of an ack. Hence in the table above, with the exception of the Dump Response frames, all the responses are either RR or UA (Call Accept, Service Accept). The above table must be understood in this context.

3.11 Conclusion and General Remarks

In this chapter the factors that influence the time required to load a failed NC have been examined and some values of the loading time estimated based on certain assumptions of error levels represented by the probability of frame error P_f, send window value K, network traffic as well as internal node latencies etc. From the exercises performed here it is seen that two factors have the most significant impact on the time taken to load a failed NC. The first is the extra network traffic due to other users during the load/dump process. The larger the number of other active users on the network, the more the time required to fully load the NC. The second factor is the speed of the X.25 link interconnecting the NA and the NC. If the baud rate is doubled from 9600 b/s to 19200 b/s in a back-to-back configuration, the loading time becomes only slightly more than half (about 0.502) the loading time at 9600 b/s. This slight loss in linear improvement is probably due to non-linearities in the model parameters such as the relationship between the average frame transmission time and the probability of frame error. Othe factors that have been considered include the transmit window sizes as well as the effect of the separation of the NC and the NA. From the representative values given here it is possible to estimate bounding values for a number of operating conditions and assumptions. We have also pointed out the difficulty in making reasonable estimates of the operating system and protocol latencies, which can easily yield the dominant term in the internal delay. Then, our results are as accurate as our limitations allow in that respect.

From Chapter 2, it was concluded that the hardware did not present a significant throughput bottleneck - the BP can handle IOP traffic with 75% success rate in a multiplexing environment with a number of constraints imposed. The NA hardware consists of essentially the same elements as the NC. Thus improving the hardware processing power of the NA will reduce only the internal latency quantities that have been referred to in the previous chapter. The other delays due to protocol processing and the physical channel limitations will remain.

Higher speed modems can also be used for the X.25 link to improve on the loading time. The data rate of the link is one the biggest system bottlenecks.

In the next chapter a new proposed design of a similar network entity, called the Domain Manager is examined. The reasons for the change have been to overcome some of the processing delays encountered here, by basing the new system on a newer higher-speed hardware and open software platform. The other reasons are based on operational demands and development flexibility as explained in the next chapter.

Chapter Four

Model of a Domain Manager

4.1 Introduction

The original Network Administrator (as considered in Chapter 3) is based on a proprietary hardware platform and the CP/M operating system. This architecture is more than ten years old. Current technology PCs provide a very adaptable cost effective solution, coupled with the multitasking, multiuser operating system UNIX. Even from considerations of operations & maintenance, system documentation procedures, and innovation it was felt that an upgrade was necessary. This was also in an attempt to improve on the processing power of the system and hence on the time required to restore service to a failed NC. This new system is known as the Domain Manager (DM).

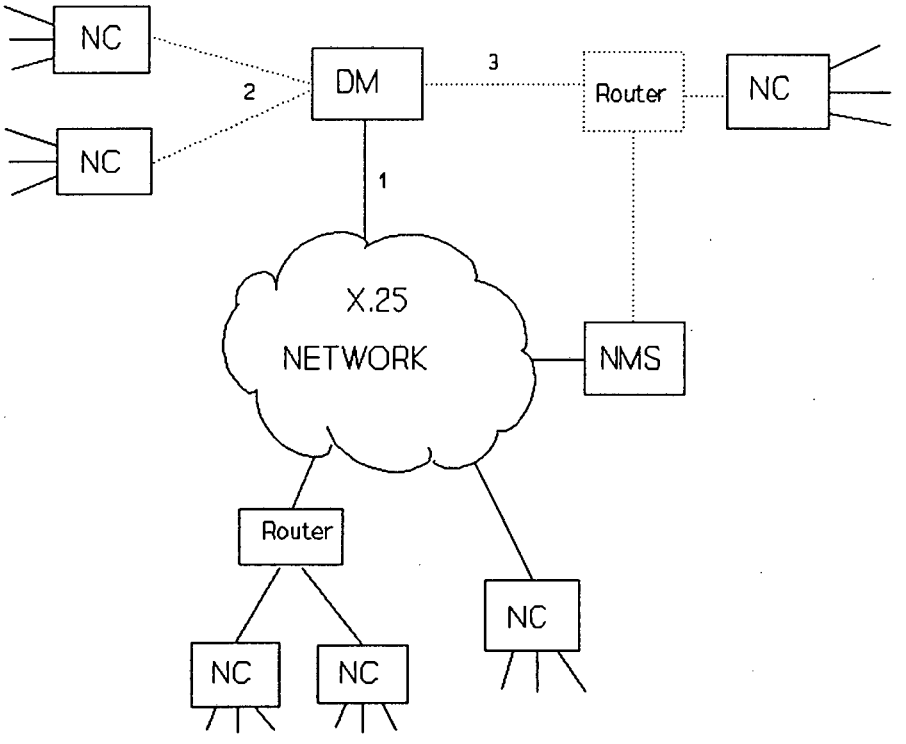


Fig.4-1 DM Environment

The Domain Manager considered here is used to provide management facilities for a set of geographically distributed NCs of the type discussed in Chapter 2. The DM runs on the platform of the InterActive Unix operating system. It also provides certain additional

operating system functionality to its own tasks, such as process initiation, message passing, and resource allocation. It provides the functionality of a transport handler. The DM itself is managed by a higher level manager, the network Management System (NMS) which is also connected to the same underlying X.25 network connecting the DM and the NCs. In this respect there are three possible scenarios for connecting the DM to its subject NCs;

- via the same X.25 network to which the NCs are connected
- by direct (back-to-back) connection
- by use of routers.

In a back-to-back arrangement, a total of 2 NCs may be connected to a single DM. As in the case of the NA and its subject NCs, the DM and NCs will not be aware of the nature of the physical connection between them as long as the X.25 interface protocol is adhered to.

During normal operation, the NCs perform their functions without the assistance of the DM. The DM is however required to be online for the running of certain tasks such as receiving information forwarded to it, supporting the login to the NMS facilities and general management, control and diagnostic functionality.

In general, DM functions can be grouped into those which are executed as a result of a request from an operator or a directive from the NMS (like building of down-loadable software sets); those which are executed periodically as result of system timeouts (e.g polling for billing and statistics information); and those which are driven by network related events (e.g dumping and loading of failed NCs, call handling etc).

During abnormal operation, the DM's main task is to return the failed NC to service without operator assistance.

4.2 Performance Measures

The performance of the DM system can be evaluated in a similar way to that done for the NA. The performance objective is still to obtain the time required to load a failed NC under similar conditions to those considered for the NA. In this section though extra care is taken to consider not only the physical hardware platform on which the system runs, but also the virtual Unix machine and its associated subtle effects of resource management etc.

4.3 Hardware Configuration

1. The PC Hardware

The PC system includes an 80486 DX processor running at 33 MHz, with a Weitek 4167 co-processor, and an ISA bus. A RAM of 16 Mbytes and secondary storage of 340 Mbytes and a 1.2 MB floppy drive are included. In certain circumstances the hard-disk can be considered in terms of its access delay when reading certain files. In general it is taken that all the downloadable software sets exist in maim memory during the loading process and normal considerations for calculating access delay of the RAM will be used.

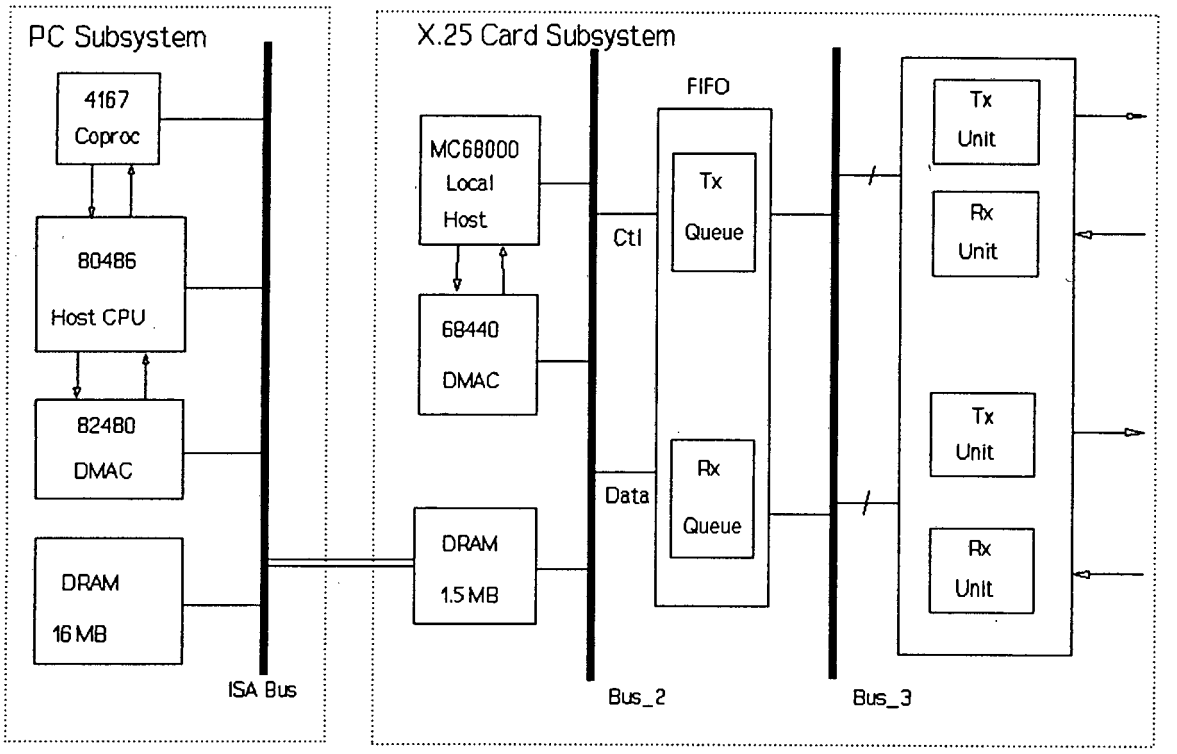


Fig.4-2 DM System Block Diagram

2. The X.25 Card

The Retix PC-320 WAN Coprocessor card is an intelligent network controller equipped with a 68000 CPU operating at 10 MHz, a 68440 DMA controller also operating at 10 MHz, and a dual-channel serial communication controller (SCC). Included on the PC-320 is a connector for the optional Retix PC-321 or PC-322 serial interface adapter card, which adds an

additional RS-232 port, whereas the PC-320 provides an RS-449 port. In addition, the PC-320 is equipped with 1.5 Mbyte on-board dynamic RAM. It interfaces to the network by a standard RS-232 port. Communications speeds range from 1200 to 9600 b/s using a low-speed link and 1200 b/s to 64 kb/s using a high speed link. The DM system uses the PC-321 card with an RS-232 for speeds up to 19.2 kb/s (due to the RS-232).

The X.25 card contains RAM and control locations in a block of addresses that are mapped to the ROM extension address range of the host PC. The host communicates to the card by reading and writing memory and control locations in the mapped address block. The mapped address block is an 8 KB block of PC host memory, starting between C8000h and DC000h. The X.25 card itself communicates with the host computer through an interrupt.
The X.25 Card provides for two links, Link 0 and Link 1 where link 0 is a low speed (9600 b/s), interrupt driven serial interface on the card. Link 1 is an interrupt-driven high speed (up to 19.2 kb/s) interface on the PC-321 card.

3. Modem eliminator

The modem eliminator is connected on the output of the X.25 card ports. The X.25 ports on the card do not provide a clock source as in the case of the current NA. The XIMs also do not provide clock in an unloaded state and therefore an external clock source is required. The modem eliminator provides for this. The modem eliminator is configurable for speeds up to 19.2 kb/s by means of hardware straps.

4.4 Hardware Models

In the following the hardware system presented above is modeled to obtain the parameters required to drive the simulation programs. The methodology used here follows that developed in Chapter 3. It considers several design aspects of the 486 processor and of the X.25 Card and its associated Serial Communications Controller.

(a) Processing Elements

1. *The 486 System*

The 486 processor has a basic cycle time of 30.3 ns and has an input controller in order to allow it to receive messages during execution of other instructions. The message list size is set to infinite so that there are no overflow messages. By using the bus burst mechanism [E-6], data can be strobed into the processor at a rate of 1 item every clock. The fastest non-burst bus cycle that the 486 processor supports is two clocks long. These cycles are called 2-2 cycles because reads and writes take two cycles each. The external system can insert any number of wait states into the basic 2-2 cycle at the end of the second clock by negating the processor's RDY# signal. The 486 also supports both 16- and 8 bit external buses by use of its bus sizing feature. In that case the CPU issues two extra cycles to complete a non-burst transfer. Hence the 486 using a 16-bit ISA bus can complete a bus cycle in 4 clocks.

2. *The 68000 System*

This processor has a basic cycle time of 100 ns and also has an input controller with an infinite message list size. The bus cycle for the 68000 is also 4 clock periods.

3. *The On-chip 387 Coprocessor*

Consider this to be part of the main CPU and does not affect the physical characteristics of the other elements connected the bus, except the effective cycle time of the CPU.

4. *The Off-chip 4167 Co-processor*

The 4167 carries out floating point operations in 32 bit, and has a basic cycle time of 30.3 ns like the main CPU. The interface with the 486 is via the global 16-bit ISA bus. When the 486 runs with the 4167, it automatically converts 32 bit transfers to 16 bit transfers and vice versa. Ignore the process of load balancing between these two co-processor systems. Also assume 4 clock periods for a bus cycle because of the 16-bit transfers over the ISA bus.

5. *The 68440 DMA Controller*

The basic cycle time of the 68440 is 100 ns. It also has an input controller with an infinite message list size. For the bus cycle we assume 4 clock periods. The 68440 can support up to four independent and concurrent DMA channels. It uses DMA block transfers in which a data word sequence of arbitrary length is transferred in a single continuous burst while the DMA controller is master of the system bus. However because of the difficulty in estimating the size of the blocks to be transferred , in the analysis we have assumed the alternative technique of cycle stealing DMA in which the DMA controller uses the system bus to transfer one data word, after which it returns control of the bus to the 68000 CPU and bus requests are later made until the transfer is complete.

The PC-321 card is considered to be passive for the purposes of this analysis.

6. *The DMA controller for the 486 board*

The system uses the 82480 DMA controller chip operating with a basic cycle time of 30.3 ns. It is also modeled by an input controller and an infinite message list size, and a 4-clock bus cycle.

7. *DRAM Refresh Logic for the PC Unit*

If the DRAM must be refreshed every 4 ms, requiring 256 cycles, and the refresh cycles are evenly distributed, then a refresh cycle occurs every 4 ms/256 = 15 μs. Assuming a refresh cycle time of 100 ns, this corresponds to 100/30.3 = 4 wait states each time, and the time to access the memory is accordingly extended. An external refresh logic circuitry has been assumed. As was done in section 2.4, the Refresh Logic is represented with an equivalent of 4 T states. The arbitration philosophy will be similar to that described in section 6.4. In the event of simultaneous requests for access to the DRAM, the priority list is the CPU, 4167 Coprocessor, Refresh Logic, and the 82480 DMAC.

8. *DRAM Refresh Logic for the X.25 Card*

Using a similar argument to the one above, the Refresh Logic for the PC-320 DRAM will also have 4 clock cycles assuming 350 ns refresh cycles.

The refresh logic circuits and the DMA control units are considered here to capture the effects of the bus/memory contention that they introduce (i.e increasing the bus cycle time), and are not included explicitly as PEs in the simulation programs.

(b) Storage Devices

The 486 system incorporates a 16 MB DRAM and an on-chip 8 KB unified cache for both data and code. 1.5 MB of the DRAM is mapped to the X.25 Card's own DRAM, which can be considered as the Shared Memory between the two subsystems, by which means all communications take place. The processes of the DM system however also share the rest of the memory space on the 486 platform. It is convenient to consider this Shared Memory as a 2-port memory module. The principal timing parameter influencing performance of the storage device is the access time. For each storage device let

T = the basic cycle time of the CPU,
T_{DCIL} = data setup time
T_{CLAV} = time for data to become valid
T_{AA} = RAM access time

Then the time required for the whole memory access operation is given by,

$$3T > T_{CLAV} + T_{AA} + T_{DICL}$$

or

$$T_{AA} < 3T - T_{CLAV} - T_{DCIL} \qquad (4\text{-}1)$$

Then from this, for the host PC DRAM the access time is found to be less than 71 ns. [E-6]

For the X.25 Card DRAM, the access time is less than [3x100ns - 60ns - 12ns = 228 ns].

The storage devices are completely described by the access time, the capacity and the files to access.

(c) Transfer Devices

For the purposes of this analysis, the complete system consists of three main transfer devices as shown in Fig.4-2. From that figure all refresh logic circuitry has been left out although the effects will be taken into account as explained above. The basic equation presented in Chapter 2 is used;

Bus cycle = queuing time + memory setting time + processor time

Queuing time is the waiting time from the time the request signal is sent until the processor receives a grant signal. Memory setting time is arbitrarily overestimated to one clock period, and allows the memory to set its address. Processor time is the time required by the PE in the memory access process and has been given in Section (b) above for each PE.

1. *Bus 3*

This TD connects the FIFO block to the serial subsystem in the X.25 card. The bus is driven effectively by the 68000 and hence has a basic cycle time of 4xT cycles, ie 400 ns.

2. *Bus 2*

This bus interconnects the 68000 CPU and its subsystems to the I/O Unit of the X.25 card. Its effective basic cycle time will now be determined.

Consider the case when 68000, the 68440 and the Refresh Logic request the use of the memory simultaneously. The minimum and maximum access times are computed as follows.

(i) The minimum bus cycle time occurs when a PE sends a request signal to the bus controller and immediately receives a grant signal. Then the queuing time is zero and,

$T_{b\_min} = 0 + 1 + 4 = 5$ cycles for each of the devices.

(ii) The maximum bus cycle time occurs when a PE requests the use of shared memory immediately after it has just completed using the memory, while other PEs and the RL

also request an access to the shared memory. Then the queuing time becomes 12 clock periods for each device. Then

T_{b_max} = 12 + 1 + 4 = 17 clock periods.

If both the extreme cases occur with equal probability, the average bus cycle time when accessing the memory is (5 + 17)/2, i.e 11 clock periods. This assumption allows us to compute the average bus cycle time.

Hence the average bus cycle time is found to be

T_{bus_avg} = 11 periods x basic processor cycle time

= 11 x 100 ns

= 1.1 μs.

3. *Bus ISA (Bus 1)*

This bus is the main bus connecting the Host PC system comprising the 486 CPU, the 82480 DMAC and the Refresh Logic for the PC DRAM to the PC-320 X.25 Card as shown in Fig.8-2.
Proceeding as in (2) above, it is found that

(i) T_{b_min} = 0 + 1 + 4 = 5 clock periods

(ii) T_{b_max} = 12 + 1 + 4 = 17 clock periods

If again it is assumed that both cases occur with equal probability ,

T_{bus_avg} = 11 x 30.3 ns

= 0.333 μs

4.5 Traffic Models

The transmission protocol used by the DM is defined by the X.25 interface using the Go-Back-N error control scheme. These issues were considered fully in Chapter 3. The main aspects of the analysis covered there, that are also applicable here include error conditions, send window values and network loading. The operational philosophy and the sequence of events that the DM follows during the dumping and reloading of a failed NC is approximately identical to that used by the NA and is fully described in Sections 3.8 and 3.9, and shall not be repeated here. The DM replaces the NA in all its functions in this case.

4.6 DM Software Model

Unix permits several processes to execute concurrently. Three of the DM processes that will be referred to in the model are the Scheduler process, the Communications Stack process and the Loader process. A brief overview of these is given below.

(a) Scheduler Process

The DM software is composed of a number of processes, some of which run throughout the life of the DM (resident processes) and some of which are executed only when their functions are required (transient processes). These processes require Unix O/S resources to be able to communicate among themselves. This is the task of the Scheduler Process. During the running of the DM system, the Scheduler Process forwards interprocess messages between the various running processes. If a message is received for a transient subtask which is not running, the scheduler process executes it if there are sufficient system resources. When the DM system terminates, the scheduler process waits until all the processes except the DM start up process have terminated. It then frees all operating system resources held by the DM system and terminates itself.

The scheduler process includes part of the functionality of the Subtask Manager STM process from the old NA. However whereas the STM process was responsible for managing only the subtasks, the scheduler process is responsible for starting and managing (and monitoring) all DM processes.

The scheduler process divides the other DM processes into process classes. The process classes are used to restrict the number of processes running at any one time. The classes consist of so-called Large Tasks, Transient subtasks, Loader subtask, Resident processes etc Whereas in the NA, the number of processes that could be active at any one time was explicitly coded, in the DM, this information can be changed in the scheduler process configuration file.

(b) The Communication Stack process

The COM process provides some of the services of a communications stack to the other processes of the DM system. It contains transport handler and session handler entities. It interfaces to the X.25 packet software below and to the DM application software above. Any process which requires data to be sent to the network writes a data-request message to the COM process. All such processes therefore have to be able to communicate to the COM process. The COM process is responsible for routing incoming messages - to itself, to other resident processes, or via the scheduler process. Thus the COM process should able to communicate to the scheduler process and other resident processes.

The DM supports a maximum of 2 links.

(c) The Loader process

This is a multi-instance transient subtask which must be run immediately when requested, and is located in its own special class. It is the process that performs the dumping/loading operation to restore service to a failed NC. A maximum of eight instances of the Loader subtask can run concurrently.

4.6.1 Interprocess Communication in the DM System

For two or more processes to cooperate in performing a task, they need to share data.
The DM uses both Unix streams and message queues to communicate information between processes. Some of the IPC mechanisms used in The DM system are briefly described below.

Shared Memory

Shared memory is used for passing data with an IPC message. The identifier (shmid) of the shared memory segment is passed in the text of the message. Before other processes are executed, the scheduler process allocates shared memory segments for information sharing between processes in the DM system. The General Shared Memory Segment holds system-wide information and is accesses by all DM processes. The Shared Memory identifier (shmid) of this segment is passed to all other processes on execution. The shmids of the other segments are placed in the general segment for access by the appropriate processes.

Semaphores

The semaphore type of IPC allows processes to communicate through the exchange of semaphore values. Many applications require the use of more than one semaphore. The Unix system has the ability to create sets or arrays of semaphores, where a semaphore set can contain one or more semaphores up to some arbitrary limit. Each semaphore within a set can be manipulated in two ways with the `semop` system call: incremented or decremented. The operating system ensures that only one process can manipulate s semaphore set at any given time. Two groups of Unix semaphores are used by the DM system. One group is used for controlling access to the shared memory segments ie for mutual exclusion. The other group is used by the DM code to control various DM activities.

IPC Streams

In essence a stream is a linear sequence of processing modules, in which data is passed between the modules in both directions. One end of each stream is bound to a user process, while the module at the other end is the device driver for the stream. The module closest to the user process forms the programming interface to the stream. The user process's calls to `write` are converted into messages which are sent down the stream and its `read` calls take data from the nearest stream module. When a stream is first opened (via a call to `open`), the two end modules are automatically connected together. Additional modules can be inserted in a stream via calls to `ioctl`. In Unix, the terminal driver sits as such an additional module, and another module can be introduced to handle the network protocol when the driver is an entry point to the network.

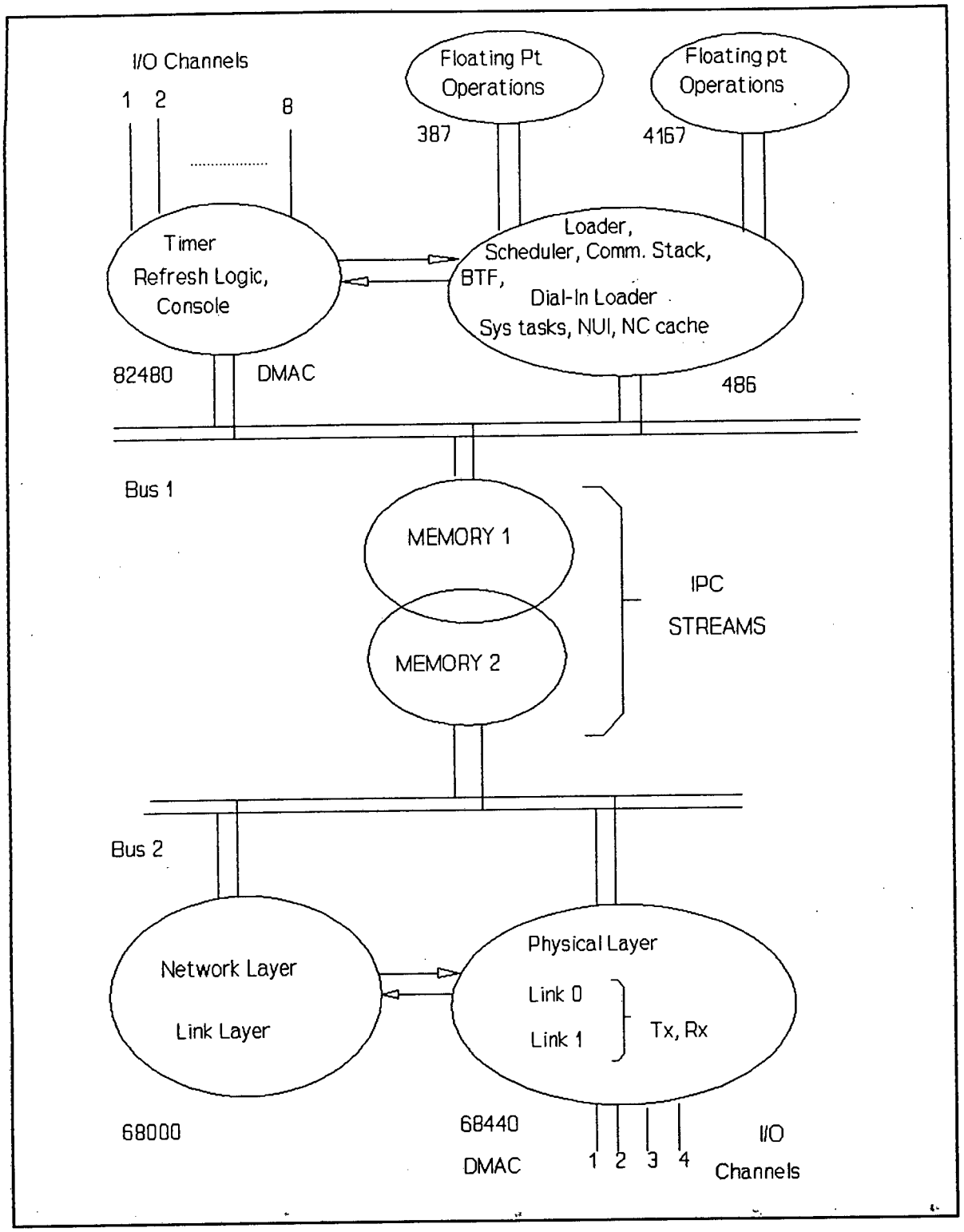


Fig.4-3 DM Software Model

Only the Communications stack (COM) process uses the streams mechanism of IPC for input. One stream is used for input from the timer process, another for input from all the other DM

processes, and multiple streams from the X.25 card.

Message Queues

In essence a message is simply a sequence of characters or bytes. Messages are passed between processes by means of message queues, which are created or accessed via the `msgget` primitive. Once a sequence is established, a process may, given appropriate queue permissions, place a message onto it with `msgsnd`. Another process can then read this message with `msgrcv`, which also removes it from the queue. All DM processes other than the COM process use the message queues for input. These are operating system resources which must be allocated and freed. They are allocated by the scheduler process on startup.

All interprocess messages are written in the same format, so that the destination process can read the message into a record variable before determining the origin and content of the message. The size of this message is 16 bytes and contains the fields, each 4 bytes in size: message type, originating process ID, destination process and the shared memory ID. In this analysis, all data message consist of a linked list of 70 byte message buffers. These are the two formats that will be used for IPC among the DM processes and between the DM processes and the WAN card.

In the DM system, workloads may be CPU intensive or I/O intensive or both. CPU intensive workloads may spend significant amounts of time in the kernel or in user mode. I/O intensive workloads may cause frequent interrupts. The operating system treats these different types of workload very differently, resulting in distinctly different performance for each workload. We attempt in the model to take into account the subtle effects of resource management in the operating system.

4.7 Internal Latencies

In the following it is attempted to characterize the internal system delays pertinent to the loading/dumping process, in both the host PC system and in the PC-320 X.25 I/O interface card. The DM system can be naturally partitioned into the Host PC and the I/O subsystems. The Host PC supports all the application processes that provide the management functionality of the DM. It also provides the transport layer interface of the system. The I/O subsystem

on the X.25 card can be thought of as providing all the network-, link-, and physical layer functions of the entire DM network.

(a) The Host PC System

Some of the major processes are the DM Startup process, Subtask Scheduler process, Loader Subtask, Dial-In Loader Subtask, Communications Stack Process, The timer Process and several other transient and resident processes. In the DM system, the DRAM is partitioned into Unix shared memory segments, and the DM processes can communicate via these shared memory segments. If any data is written to the 1 MB address space starting at either C8000h or DC000h (it does not matter which) in the PC DRAM, that data is also mapped to the PC-320 DRAM, and allows IPC between the DM processes and the I/O system. In this way data can be transferred in and out of the PC system.

When a failed NC makes a load request to the DM, the DM-NC system must go through the sequence of events given in Fig 4-4. The Scheduler process must start up the Loader process after determining that there are less than eight instances of the Loader process, otherwise the request is queued and started by the Communications stack process after one of the running processes terminates. The distribution of this delay is one of the factors that will affect the loading time, in addition to those already considered in Section 3.10.

4.7.1 Less than or Equal to 8 Instances of Loader Process

The loader process is started when data arrives for it. The Scheduler process passes to it certain parameters (process name, instance number, *shmid* of shared memory segment etc) which we represent as just a single 16 byte message of the format described in Section 4.6 above.

(i) Transmission time

The Loader process must read the message sent by the Scheduler process from shared memory. All read/write instructions are made to cache. The 486 contains a cache fill buffer and a cache read buffer, each 128 bits wide. The cache block size is also 16 bytes.[E-6] We assume that cache access takes only one clock cycle. The hit rate for this cache is 85 -

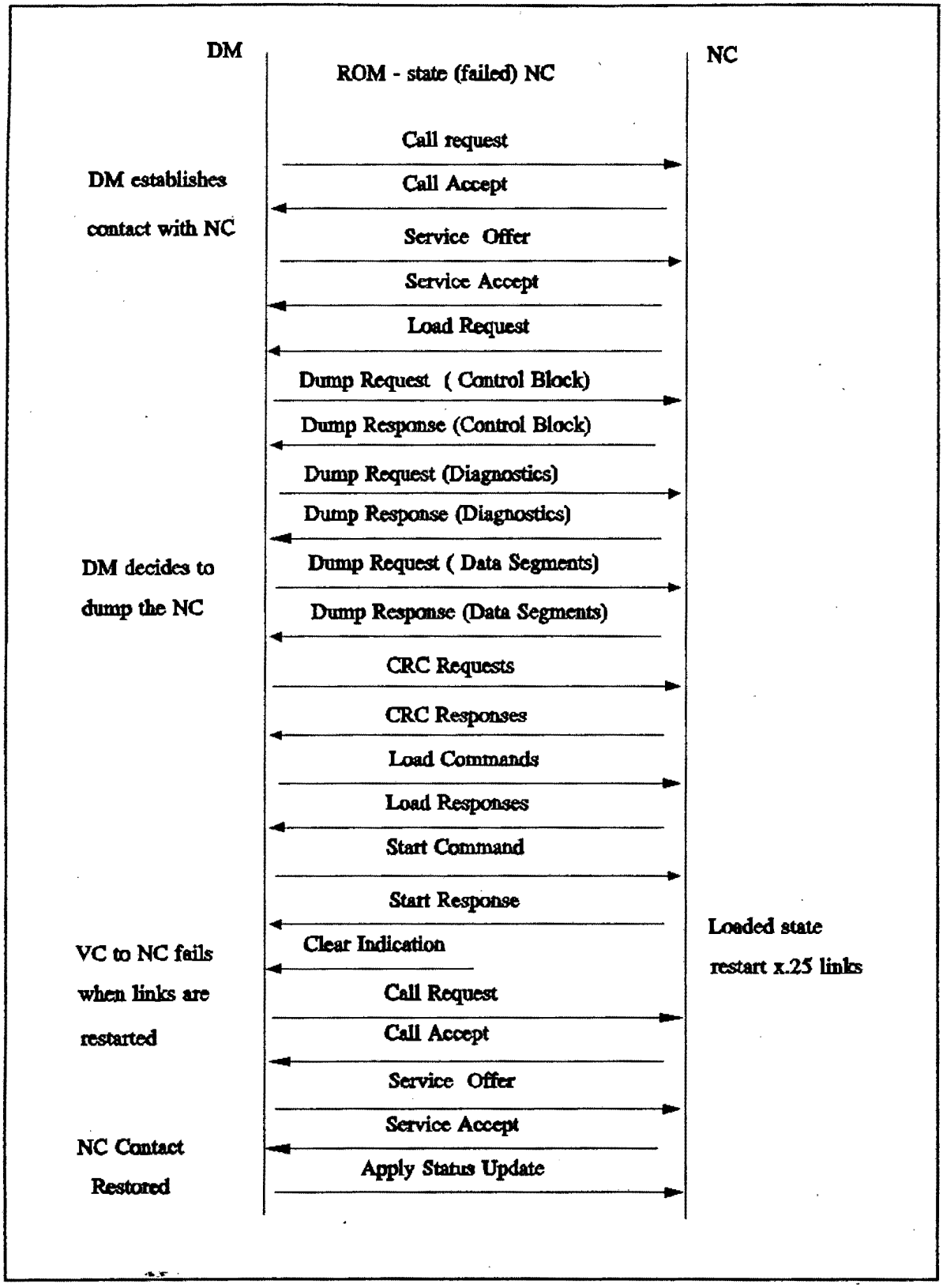


Fig.4-4 NC Load/Dump Sequence

86% [E-7] (say 85%). Hence upon a cache hit, a read or write cycle takes 1 clock period, i.e 30.3 ns. Upon a cache miss, main memory must be referenced, and this takes 0.333 μs to transfer 16 bits as determined above (ignoring bus burst), and 71 ns to access the memory. That is the total time to transmit 2 messages (32 bytes) is

$$32/2 * (0.333 + 0.071) = 6.464 \mu s$$

Using a miss ratio of 0.15, the average read/write time (32 bytes) is

$2*(0.85 * 30.3 \text{ ns}) + 0.15 * 6464 \text{ ns} = 1.021 \mu s.$

In general, all the data that must be exchanged or transmitted to the X.25 card will be in shared memory and the message will have a pointer field pointing to the data address.
A buffered write-through technique keeps main memory consistent with cache by passing all store operations to cache. After a block is copied to cache, the process does not wait for the block to be written to main memory. Hence because of buffers, a store operation is executed in one cycle. At a later time (after about 4 consecutive writes [E-7]), when the kernel requires a free buffer and if none is available, the kernel selects a buffer on which delayed write is pending, and writes the buffer to main memory.

(ii) Processor Scheduling Time

In the Unix System, some of the functions associated with the kernel are just processes that compete for the CPU with other DM processes. We assume that these processes are allocated the highest priority {1} in the system. No DM application processes or other user processes will be run until all high priority system processes that are ready to run have executed.

In the DM it is possible to identify the following subtasks that run at some time in the life of the system; {2 large tasks (such as the configuration process), 5 transient subtasks (eg Printer subtasks), 8 Loader processes, 4 Dial-in Loader tasks, about 8 resident processes (eg Timer, Comm Stack etc), 10 console subtasks, about 8 network user ID (NUI) processes, and about 4 high priority subtasks (e.g cache requests from the NC)}. From this it can be assumed that at any instant there will be about 40 processes in the system awaiting service at the processor. At the CPU each process can be placed in different classes. A priority is associated with a particular class. The Priority Subtasks will be placed in Class 2 and the Loader process will

for the sake of the analysis be arbitrarily placed in Class 4. In the following the goal is to calculate the waiting time for a process in Class 4 before it gets served at the CPU. the system can be modeled as a single server of a general service time $1/\mu$ pdf, served by r priority queues. The respective arrival rates at the queues are $\lambda_1, \lambda_2, \lambda_3, \ldots\ldots \lambda_r$.

Then make a number of important assumptions to facilitate the analysis to compute average waiting times for each loader instance;

(1) The service is non-preemptive across the priority classes.

(2) The arrivals for the high priority classes are independent and follow Poisson arrival statistics. This is because it has been assumed that all the high priority subtasks require a short time to complete execution, and no unfinished tasks needs be rescheduled.

(3) The arrivals of the lower priority are not independent and follow a general (deterministic) distribution. Barberis shows in [E-10] that although the interarrival times of the resulting composite flow are not independent of each other, due to time-slicing, this dependence is not important in the average delay evaluation.

(4) Within the individual classes, all the members are instances of the same subtask. Hence a Round Robin scheduling scheme will be used to improve the response time. It seems reasonable to assume that all the processes in the same class make approximately the same demands on the CPU time.

(5) The frequency of hardware interrupts that would cause preemption is negligible.

The rest of the analysis is completed in Appendix F.

To compute the average waiting time it is necessary to explicitly partition all the DM processes into priority classes as required by the above analysis. This is done using very rough estimates. Consider the following example for Class 1, the highest class.

Example for Class 1

The members in the class are all system processes and require short execution times. For example the First-Level Interrupt Handler which performs context swapping and transfers control to some interrupt service routine. In the 486 this process takes about 17 μs as explained below. Average this for all processes in the class to 50 μs. As for the arrival rate it can be assumed that since the Unix system uses a 1-second interval to forcibly preempt processes, the average arrival rate of these processes is about 2 processes per second. Proceeding in this way for all the 5 classes, we obtain the table below.

Table 4-1 A view of DM process classes

Class	Typical Processes	Example of process Functions	Average Service Time	Average Arrival Rate /s
1	System Processes	First-level Interrupt Handling	50 μs	2
2	Short term Scheduler, Dispatcher	Update PCB, Activate ready process	50 μs	100
3	Priority DM Processes	Service NC Cache Request	1 ms	0.002
4	Loader Process	Dumping/Loading Failed NC (via DMAC)	1 ms	0.033
5	General DM Processes	Spooling (DMA)	1 ms	0.001

From the above table and the results of Appendix F, obtain the traffic intensities as

$[\rho] = [50\text{x}10^{-6}, 5\text{x}10^{-3}, 2\text{x}10^{-6}, 33\text{x}10^{-6}, 1\text{x}10^{-6}]$

Now evaluate $E(T_o)$ as follows.

$\rho_1 = \lambda_1/\mu_1 = 2 \text{ x } 50\ \mu\text{s} \quad = 10^{-4}$

$\rho_2 = \lambda_2/\mu_2 = 100 \times 50\ \mu s = 5 \times 10^{-3}$

$\rho_3 = \lambda_3/\mu_3 = 0.002 \times 1\ ms = 2 \times 10^{-6}$

$\rho_4 = \lambda_4/\mu_4 = 0.033 \times 1\ ms = 3.33 \times 10^{-5}$

$\rho_5 = \lambda_5/\mu_5 = 0.001 \times 1\ ms = 10^{-6}$

and

$\sigma_1 = \rho_1 = 0.0001$

$\sigma_2 = \rho_1 + \rho_2 = 0.0051$

$\sigma_3 = \rho_1 + \rho_2 + \rho_3 = 0.005102$

$\sigma_4 = \rho_1 + \rho_2 + \rho_3 + \rho_4 = 0.0051353$

$\sigma_5 = \rho_1 + \rho_2 + \rho_3 + \rho_4 + \rho_5 = 0.0051363$

Proceeding

$$\sum_{k=1}^{r} \lambda_k E(\tau_k^2) = \sum_{k=1}^{r} \lambda_k [(\frac{1^2}{\mu}) + \sigma^2]$$

$= 2[(50 \times 10^{-6})^2 + 0] + 100[(50 \times 10^{-6})^2 + 0] + 0.002[(10^{-3})^2 + 0] + 0.0333[(10^{-3})^2 + 0] +$

$10^{-3}[(10^{-3})^2 + 0]$

$= 2.922 \times 10^{-7}$

so that

$E(W_1) = E(T_o) / (1 - \sigma_0)(1 - \sigma_1)$

$= 2.922 \times 10^{-7} / 2[(1 - 0)(1 - 0.0001)]$

$= 1.461 \times 10^{-7}$ sec

$E(W_2) = 2.922 \times 10^{-7} / 2[(1 - 0.0001)(1 - 0.0051)]$

$= 1.469 \times 10^{-7}$ sec

$E(W_3) = E(W_2)/(1 - \sigma_3) = 1.469 \times 10^{-7} / (1 - 0.005102)$

$= 1.476 \times 10^{-7}$ sec

$E(W_4) = E(W_3)/(1 - \sigma_4) = 1.476 \times 10^{-7} / (1 - 0.0051353)$

$= 1.484 \times 10^{-7}$ sec

$E(W_5) = E(W_4)/(1 - \sigma_5) = 1.484 \times 10^{-7} / (1 - 0.0051363)$

$= 1.491 \times 10^{-7}$ sec

It has been mentioned above that within the individual classes, the Round Robin scheme is used to give a fast response time for the user processes.

(iv) The delay due to Round Robin Scheme

This method forces a long process to go through the active and execute queues repeatedly, limiting the amount of time it could spend in the execute state on each cycle. If a process failed to complete its work in the allowed time slice, it is preempted from the execute state and placed on the end of the execute queue, and has to work its way to the head of the queue again before it is allowed to execute.

Let the time slice be q, and the CPU time required by an average process be T. Let m be the number of times the process must enter the execute state before it receives all the CPU time it needs. Then

$$T = mq$$

Suppose that when the process enters the active queue there are n processes ahead of it. Then

this new process must wait until all n processes ahead of it have been serviced, which will be at most

Response time $= nq$

The aim here is to obtain the wait time in the queue. If the process is short and can complete in q seconds or less, the wait time is the time it waits to first reach the execute state, and is equal to the response time

Wait time $= nq$

If the newly arriving process is a long one, where t is much larger than q, such that

$t = mq$

the time that the process must wait before it receives t seconds of CPU time can now be found as follows. Since it must pass trough the active queue m times, and each time through it waits nq seconds, the total wait time is

Wait time $= m(nq)$

$= t/q\ (nq)$

$= nt$ (4-4)

where t is the execution time for the individual process.

The assumption here is that there are always n processes ahead when the process enters the active queue.

Hence for the Loader process $n = 7$. To obtain the value of t, a simulation is performed assuming a single Loader process. All the other delays are taken into account except the delay within the Loader process class. From this an approximation of t is found as $t = 191.775$ sec. Compare this with 191.8347 sec for the NA under the same conditions. In other words an improvement of only 60 ms is realised over the total loading process. In general this is in agreement with the observation we made earlier, namely that performance improvement

would be obtained not from faster hardware but from a different software protocol processing strategy. In any case we obtain the total wait time as

$nt = 7 * 191.7746$ sec.

To this add the time that the process must wait from across the priority classes, ie 121 μs, to obtain the maximum latency in the DM system as seen by a single loader process to be

Latency = 1342.422 sec.

It is now possible to produce the table for any number of loader processes in the same way.

Table 4-2 Loading times for the DM for NCs on X.25 Network

Loader Instances	1	2	3	4	5	6	7
Loading Time	191.775 sec	383.549 sec	575.32 sec	767.10 sec	958.9 sec	1151 sec	1342 sec

There is a slight difference between these figures and measured values due to the dynamics of the traffic in a network environment.

Because of the presence of the two coprocessors, the waiting time is reduced by the proportion of floating point operations that each process contains. It is not clear how we can determine this quantity. Assuming the worst case, we shall ignore this benefit.

(iii) Context Switching

Once the Unix kernel allocates the CPU to a user process, the user process can hold the CPU until the next 1-second clock interrupt if no other interrupt occurs. If a hardware interrupt occurs while the CPU is executing a process in the kernel mode, execution of the interrupted process resumes after all the interrupts have been handled. If an interrupt occurs while the CPU is executing a process in the user mode, execution of that process is preempted. After the CPU has served the interrupt, the Scheduler process assigns the CPU to he process with the highest priority that is ready to run. The 486 supports task switching in hardware. It saves the entire state of the machine, loads a new execution state, performs protection checks, and

commences execution in the new task in 17 μs [E-6] [E-7].

4.7.2 More than 8 instances of Loader Process

In this case, the new process has the lowest priority in the system and therefore experiences the worst delay from the priority classes. In addition the above delay must be considered. The time spent in the queue can be determined from Little's Law, provided the arrival pattern of all the processes in the system is known.

(b) The I/O Processor System

The X.25 Card performs all of the X.25 interface protocol functions. It performs interprocess communications via shared memory, and network control functions relating to framing, error control, source address generation, and destination address checking.
The Tx and Rx FIFOs may be thought to be independent and of size 128 bytes each.

(i) Transmission time

Read/write messages to shared memory: the 16 bit bus's cycle time is 1.1 μs and the memory has an access time of 228 ns.

Hence to read and write two 16 byte messages requires

$32/2 * (1.1 + 0.228) = 21.25\ \mu s$

(ii) Error control

The X.25 protocol uses the Go-Back-N error control scheme and a baud rate of 9600 b/s and a frame size of 135 bytes and a network radius of 50 km are assumed. It has already been shown that for transmission in the presence of errors, the average time for a correct transmission to be received is

$$T_{avg} = T_I \frac{[1 + (a - 1) P_f]}{1 - P_f}$$

where

$$a \triangleq T_T / T_I = 1 + T_{out}/T_I$$

Then if a send window of K is implemented, it follows from the previous argument that

$T_T = KT_I + T_{out}$ where

$T_{out} = 2T_{prop} + T_{proc} + T_{ack}$

and the terms have been defined in Section 2.7.

Appendix C presents tables showing the effective frame transmission times for the Go-Back-N error control method in the presence of errors. T_{avg} translates to an **effective frame size.** This data forms a main input into the simulation, and will not be combined in the generalised service time.

(iii) Processing Delay

The PC-320 system performs several network layer and link layer functions including framing (which we assume to include addressing functions), the implementation of the X.25 protocol in the transfer of the downloadable software sets. It also provides IPC with the host PC system by use of Unix streams. Because of its DMA capability, the PC-320 can receive frames and store the in shared memory while the 68000 handles communication with the Host PC, or performs other tasks related to frame processing. Characterise the components of the processing delay time as follows.

- Frame processing consists of framing, error checking, address generation etc. Assume that each frame is a standard 135 bytes consisting of 128 bytes of data and 7 bytes of overhead. Approximating the framing delay by an equivalent of twice the 7 bytes in read cycles and an arbitrary number of processing cycles say 100 per frame. Then

  Framing delay = (2 * 7 * 8 bits / 16 bits) * (1.1 + 0.228) μs + 100 process cycles

$= 20\ \mu s$

- IPC with the Host PC occurs by means of Unix streams, and can be approximated by the time required by the Comm process to pass the requisite parameters such as buildset file parameters etc to the Loader process. Assuming that every message is 16 bytes long, and that the stream is opened (initiated) by the Loader Process, it takes an equivalent of two bus transfers before the data is received. Then

  IPC Delay per frame = $2 * 16 * (1.1 + .228\ \mu s) = 21.25\ \mu s$

- The last component of processing delay comprises pure processing cycles. If say each byte takes 10 cycles to process, then for a 135 byte frame it needs 1350 cycles or 135 μs.

Then the total processing delay is about 200 μs.

(iv) Process Scheduling Delay

There are several processes running in the PC-320 system such as the eight instances of the Loader process (which is an I/O process), error control and interrupt handling processes, Receive/Transmit processes, management information processing and other protocol specific processes. Other delays are due to the precondition time for semaphores etc, which can be measured from the simulation. Assume that once the Loader process has been scheduled to run on the PC system, it will have to be rescheduled at the IO Processor System. Assume that there are 3 priority classes in the system as follows.

Class 1 - protocol specific processes and system processes
Class 2 - Loader process
Class 3 - Management related processes

As an example consider Class 1. Framing occurs at least 7 times in every frame duration (112.5 ms) i.e 63 times/sec. Considering other similar process it is found that the arrival rate is about 150 processes per second. Proceeding similarly for the other classes we have the table below.

Table 4-3 A view of I/O Processor Process Classes

Class	Typical Processes	Example of process Functions	Average Service Time	Average Arrival Rate /s
1	Protocol fns, Sys processes	Framing Error checking	1 ms	150
2	Loader Process	Build sets verif-cation, data Xfer initiation	1 ms	20
3	Network Management Processes	Billing Console S/tasks	1 ms	0.2

Then from previous,

$E(W_1) = E(T_o)/(1 - \sigma_1)$

$E(W_2) = E(T_o)/(1 - \sigma_2)$

$E(W_3) = E(T_o)/(1 - \sigma_3)$

where

$\rho_1 = \lambda_1/\mu_1 = 1 \text{ ms} \times 150 \text{ /s} \quad = 150 \times 10^{-3}$

$\rho_2 = \lambda_2/\mu_2 = 1 \text{ ms} \times 20 \text{ /s} \quad = 20 \times 10^{-3}$

$\rho_3 = \lambda_3/\mu_3 = 1 \text{ ms} \times 0.2 \text{ /s} \quad = 2 \times 10^{-4}$

and

$\sigma_1 = \rho_1 = 150 \times 10^{-3}$

$\sigma_2 = \rho_1 + \rho_2 = 170 \times 10^{-3}$

$\sigma_3 = \rho_1 + \rho_2 + \rho_3 = 170.2 \times 10^{-3}$

Then

$$\sum_{k=1}^{r} \lambda_k E(\tau_k^2) = \sum_{k=1}^{r} \lambda_k [(\frac{1}{\mu})^2 + Var\ (\tau^2)]$$

$$= 150\ [(10^{-3})2 + 0\ ] + 20\ [(10^{-3})2 + 0\ ] + 0.2\ [(10^{-3})2 + 0]$$

$$= 170.2 \times 10^{-6}$$

so that

$$E(W_1) = 170.2 \times 10^{-6} /2(1 - 150 \times 10^{-3})$$

$$= 1.00 \times 10^{-4} \text{ sec}$$

$$E(W_2) = 10\text{-}4/(1 - 170 \times 10^{-3})$$

$$= 1.21 \times 10^{-4} \text{ sec}$$

$$E(W_3) = (1.21 \times 10^{-4}) / (1 - 170.2 \times 10^{-3})$$

$$= 1.45 \times 10^{-4} \text{ sec}$$

If a single Loader process in the PC-320 is assumed, then an initial value that can be used for starting the simulation is obtained. The implementation of the Round Robin scheme will result from further simulations. Thus, finally we obtain the delay for the I/O processor system as

Latency = 200 μs + 121 μs = 321 μs.

(c) The NC System

The ROM-state NC is assumed to have the same internal delays as that used during normal NC operation in Chapter 2. This is so as to create a basis for comparison between the old NA and the new DM systems. From Chapter 2, the IOPs were modeled by a latency of 8.7 ms, and the Block Processor a latency of 9 ms.

4.8 DM Loading an NC over a Sample Network

In this section, the example network of Section 2.7.2 is used to investigate the loading time of the NC by the DM over a network. The configuration of the network from Section 2.7.2 is modified and given again here in Fig 4-4.

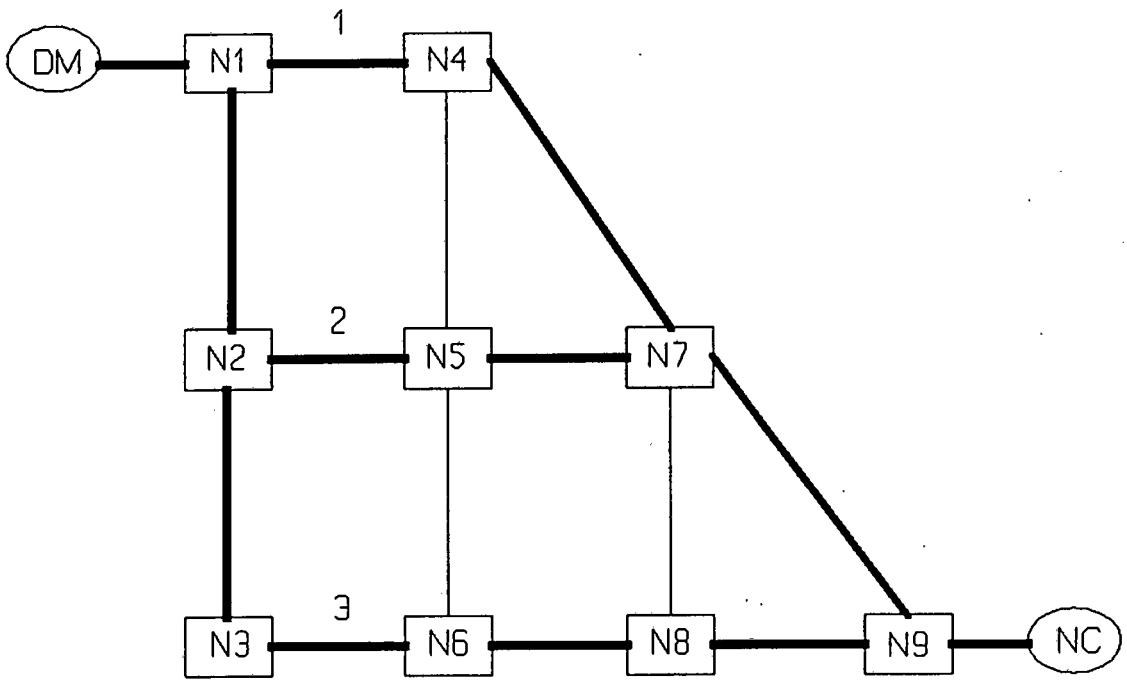


Fig.4-4 Sample network for DM loading a remote NC

The conditions to be assumed in this example are the following;

There are a total of three virtual circuits between the edge nodes N1 and N9 with varying possibilities of each being chosen for any one frame transmitted.

The interference interval has a mean of 5 seconds, a lower bound of 100 microseconds and an upper bound of 200 seconds for each of the stations 4 5 and 8, where each interference message size is 512 bits.

The DM sends a total of 220 164 bits of data in commands over the network to the NC, and the NC in turn sends a total of 80 632 bits in responses to the DM. These figures correspond

to the assumption made earlier that the DM down-loads about 204 KB of data to a failed NC.

For this setup, the following results were obtained;

Minimum loading time = 192.7 seconds

Maximum Loading time = 345.2 seconds

Average Loading time = 268.8 seconds

Standard deviation Loading time = 54.3 seconds

The simulation of the above scenario requires a considerable amount of time to run the program (nearly 17 hours on the machine that was used). As a result it was not practical to perform the simulation over several other variables such as nodal latencies, different number of NCs to be loaded across the network, error conditions etc. This was not a serious drawback since all these factors have been adequately analyzed elsewhere in this report. The major objective of this exercise was to carry out a performance measurement, and that objective was achieved by using an optimised program similar to those that were used in NC response time measurement in Section 2.7.2. Also of great note is the fact that since the loading time of the DM was found to be identical to that of the NA, te results that came out of this exercise apply equally validly to the NA.

4.9 General Findings and Discussion

The main assumption used in the tests carried out here is that the software sets to be downloaded are identical in the NA and the DM.

1. Under low transmission error conditions, the DM takes 191.775 seconds to load an NC if it is the only one to be loaded. This is identical to the NA loading time of 191.835 sec.

2. In the event that there are 8 NCs that must be loaded at the same time, the worst case loading time is 1533.88 seconds.

3. If the X.25 link uses 19200 b/s modems, the loading time is nearly halved to 95.9 sec,

for a single NC in a back-to-back arrangement.

4. If the NC system latencies are reduced by half, the loading time becomes 191.735 seconds.

5. If the latencies of the DM system are reduced by half, the loading time becomes 191.772 seconds.

6. If two 9600 b/s links are used to load the NC via ports 1A and 1B, the loading time is also cut by slightly less than a half to 96.83 seconds.

From these observations it is clear that the loading time will not be improved by the current proposal of the DM system. Further improving the processing power of the DM will not improve on this loading time. It is clear then that the easiest and most economical means to dramatically improve the loading time is by using higher speed modems.

Since it can be assumed that the incidence of NC failure is fairly low, the normal tasks of the DM are not related to NC loading but to general network management functions of network configuration, software sets building, general and billing information processing as well as supporting operator functions. All these functions derive an execution time benefit that is linearly related to the processing power of the DM. Therefore these functions are well served by the DM. The other functions of statistics collection and other file transfers are not time-critical and their response time performance is irrelevant to the network user.

Chapter Five

Summary

This chapter presents a brief overview of the performance aspects of the Value Added Network that were covered in the thesis. It is important that the remarks made in this section be seen in the context of all the arguments, assumptions and premises, limitations and other discussions made in the respective chapters in the rest of this report.

5.1 Response Time Modelling of the Asynchronous Network

Two main scenarios were used in modelling the response time of the asynchronous terminal network.

(a) End User - Local NC Response Time

The scenario that was considered in this section comprised a cluster of terminals or workstations that access X.25 network services through an ATX NC. The main performance issues that were investigated for this User Terminal - to - Local NC case include the following

- The average requester traffic is either interactive as in the case of terminals or can consist of bursty transfers in the case of workstations. In either case, the messages are small, and the end users are easily conscious of any delays in system response time. A user who transmits smaller messages expects a better response time than a user who transmits large files. The case of small messages was investigated using messages of not more than 6 bytes of user data.

- The baud rate characteristics of the X.28 connection, in order to find optimal performance values.

- Message generation pattern, in terms of the distribution of messages and their generation

rate.

- Relative NC and terminal processing power, varied over an order of magnitude for the terminal/workstation.

- How many user terminals the NC is connected to - from one to eight users.

Major Results

The major results obtained here indicate that

- Baud rate has the most dramatic effect on the response time. Doubling the baud rate in the range 300 - 19200 b/s produces an average of 19% improvement in response time. It has been shown in this report that link speed can easily become the biggest bottleneck in the network performance.

- Terminal processing power has minimal effect on the response time. This is because message transfer is an I/O intensive process and consumes few processing cycles. Hence if the processing power of a terminal is increased by 50%, the improvement in response time is only less than 1%. Of course all CPU intensive processing benefits from an increase in the processing power of the CPU.

- NC processing power has a slightly more noticeable improvement in the response time but is also not critically beneficial. The reason for this is the same as for the case of terminals considered above, namely that most of the NC activity is I/O intensive. As the latency of the NC is reduced by 50%, the response time improves by only 5%.

- The number of users supported by the NC has a direct relationship on the response time. Because the 8086 Block Processor architecture does not support multitasking, all loads must be queued and processed in a semi-sequential manner. Hence as the number of users increases, so the response time of the system degrades correspondingly, in a linear relationship. It is expected though that if the number of users continued to rise to large

numbers, the response time would start increasing disproportionately to the additional users because of inadequate storage buffers and consequent dropping of messages.

- Models were developed to measure the turn around delay for small messages that are serviced completely at the NC. For messages of up to 6 bytes, we found that for the worst case of 300 b/s baud rate, and slow terminals [20 ms to 1 μs cycle time], the response time was under one second between the user and the local edge node NC.

In Section 2.7.1, the above issues and the results were covered in greater detail.

(b) Response Time for Two Remote Users

In the second scenario, the ATX NC can be seen as a dedicated PAD that provides functionality for terminals to access the X.25 network when communicating with another end-station across the WAN through at least two packet switches. Three switched virtual circuits were implemented. The users generated traffic according to the Poisson distribution. For this case the primary considerations were

- The node delays. To each packet switch was ascribed a certain delay time required for message storage, error checking and/or correction and forwarding.

- Protocol parameters including flow control mechanisms, window flow control, errors on the transmission channel

- The number of links and virtual circuits and their transmission characteristics in terms of baud rates.

- Message generation pattern and statistics.

- Caching of network services at the local edge NC, ie to find out the response time given a certain probability of finding the required service (eg NUI information) at the local NC.

- Network traffic due to other network users over and above the two users under observation.

- The relative data rates of the internode links and the local links to the NC

Major results for the configuration of 7 switching nodes and two edge NCs are the following

- interference traffic has an effect whether it is on the VC or affecting the node through another link

- the response time is position dependent in the network (number of nodes crossed)

- the case of caching is relevant for management information and services only but not for end user communication. This can be useful for optimising the location of network management resources.

- For the configuration that was used, consisting of 7 packet switches and two edge NCs, the worst case round trip delay time was found to be about 1.3 seconds. This response time is affected mostly by the baud rate of the communication line, node latency, the access time of the 'server data base' at the destination user, and the intensity of traffic due to other network users on the links in question.

The network response time in this case can be optimised by

- minimal number of hops
- minimal nodal delays
- high bit rates used on the internode links.

These issues are discussed more fully in Section 2.72

5.2 NC Performance Modelling

The NC is a central component of the VAN that was studied in this work. As an ATX NC it acts as a gateway for the X.28 terminals into the X.25 network, and in its generic capacity it provides data channel multiplexing, and it participates together with the NA/DM in the management of the VAN by collecting statistics regarding faults, providing configuration/status information, as well as provision of billing data. In general however, the major NC function is I/O intensive, hence the emphasis here on the two measures of throughput and response time when considering the performance of the NC system. Further, a good understanding of the performance limits of the NC is important in order to relate it to the performance of other network elements such as the NA/DM, as well as to help make decision on whether a performance upgrade is necessary, and which of the sections of the network elements require upgrading, and perhaps be able to design around those limitations in the other elements like the DM system. Information on NC performance is also important because it allows optimization of the links connecting to the NC both from the X.25 side and from the X.28 side. It also permits a decision on the number of users that the NC can handle. Finally a knowledge of the NC bottlenecks permits the understanding of the implications of these constraints in the software distribution phases, eg during initial installation or during the downloading of operating software after a fault.

The issues that were considered in the evaluation of NC performance include the following

- congestion in the BPM
- different relative processing levels of the BP and the IOPs
- utilization of shared memory
- modelling the traffic in the both the ATX and the XTX NC
- throughput measurement

The major task here was the abstraction of an existing value added network into a simulation model that would capture vital characteristics of the system in terms of

- the hardware blocks of mainly the BPM

- the traffic pattern in the NC subnetworks
- system dynamics in terms of latency of the BPM elements
- the transmission environmental effects - errors and transmission delays
- software aspects of the system including protocol overheads

Various configurations of the system were investigated with the aim to measure throughput limitations and identify system bottlenecks. The investigations performed here indicate a system operating below capacity for the ATX NC in terms of asynchronous terminal traffic throughput at the operating point of 8 terminal ports and 9600 b/s. If a realistic traffic rate of continuous messages is assumed then a throughput value of 75 % is still achieved with an 8 port Single Board Unit XTX NC.

5.3 NA Performance Modelling

The NA implements the full X.25 protocol in all its communication whether with a packet switch or with the NC. For that reason, much of the analysis of its performance relies on the properties of the X.25 protocol concerning frame sizes, error control, window flow control and the use of virtual circuits. Another parameter that was investigated is the protocol processing overhead of the X.25 protocol and the software distribution protocol for a ROM-state NC. Two specific topological configurations were analyzed. In the first case The NA and the NC were connected back to back but at the same time abiding fully with the X.25 protocol. In this first case there was minimal interference to the communication processes between the two systems. In the second case, the NA is remote from the NC, and the software distribution is performed over a number of packet switching nodes.

The major conditions investigated included the following

- transmission under error conditions
- using different frame sizes
- using different window sizes
- the effect of flow control
- number of virtual circuits supported on the same links as the NA and the NC

- overheads of buffer management
- different topological configurations, back to back, etc
- the impact of propagation delay

In this section various models were developed to evaluate the performance of the NA in terms of loading time when automatically restoring service to failed network elements. Again the main tasks involved model abstraction. Where necessary, analysis was used to help in the reasoning and evaluation of model parameters. Various network conditions, configurations, and loading were investigated.

At a fixed baud rate, the main limiting factors to the loading time of the system in an approximate order of decreasing importance were found to be

- Network traffic. If the traffic intensity increases, the loading time increases in an approximately exponential manner. The loading time is lower bounded by the back-to-back loading time of about 192 seconds. In practice, the NA will abandon the loading process if the time required to restore the NC exceeds about 7 or 8 seconds, according to the network operators of the Easy Access system.

- Line error levels. If the frame error probability is varied from 10^{-6} to 10^{-1}, the loading time changes from about 192 seconds to more than 335 seconds for a send window value of $K = 7$ and a baud rate of 9600 b/s.

- Transmission window size. The effect of increasing the send window size is to increase the loading time in the presence of high error rate. When the incidence of errors is low, the loading time converges to the same value irrespective of the window size. This is an expected result from a knowledge of the Go-Back-N protocol.

Because the amount of data to be transferred from the NA to the NC is not always known in advance, that is another important variable that determines the loading time. By optimising the downloadable memory segments, error levels, transmission windows, and assuming low extraneous network traffic, a loading time average of about 3 minutes was obtained for a

back-to-back NA/NC arrangement, and up to 5.6 minutes if the NC was loaded remotely via a network using a baud rate of 9600 b/s on the X.25 link.

5.4 DM Performance Modelling

Since the DM was intended as an upgrade of the NA, exactly the same indices had to be used for the analysis of the DM performance, as for the NA. Because the DM is based on a newer PC architecture for the Host PC as well as an I/O Processor subsystem, and on the Unix operating system, the process of model abstraction was rather challenging. The main considerations that had to be made include the following,

- IPC between the Host System and the IO Processor Subsystem
- Resource management of the Unix system
- Protocol processing overhead

In addition, a number of other issues were taken into account;

- Host PC architecture
- IOP architecture and its performance
- Protocol and NC loading performance
- Error performance

In this section the new system was modeled with two particular objectives; to predict the absolute performance of the system as well as to draw parallels between the DM and the NA. Hence, the performance measures used were the same. The modelling however took into account the characteristics of the different hardware platform and the resource management features of the Unix operating system.

Over the same operating range, the DM's response time was measured to be virtually the same as that of the NA in loading a failed NC. The main reasons for this are seen to be as follows.

1. The physical medium capacity limitations. Using 19.2 kb/s modems in both the NA and the DM is seen to improve on the loading time by a factor of just less than 2.

2. The internal latencies of the NC system itself are still the same. Also, the loading process is a form of message passing communication between the DM and the NC. There is a form of inherent synchronization in this process of communication; the DM has not finished loading until the NC has transmitted a START RESPONSE back to the DM. In other words the DM is tied down by the delays that the NC may have. The impact of the proportion of the delay factor is however small - see also Section 2.7.

3. Other system effects which could not be accurately captured, such as an effective model of the operating system, as well the X.25 Wide Area Card System. The work in Section 4.7 is an attempt to reduce this uncertainty.

4. The X.25 baud rate was still assumed to be 9600 b/s. Section 4.8 gives the results for a test using 19200 b/s links.

5. The communication protocol between the DM and the NC is still the same as it was between the NA and the NC - i.e, the protocol processing overhead is still unchanged.

However because the more predominantly used functions of the DM relate to information processing such as configuration file building, billing information processing, support of operator functions etc, the DM system's low latencies which are about an order of magnitude better than those of the NA, the response time for internal processing will be correspondingly improved. That is, a significant performance benefit is realized for local processing functions.

As for the NA, the DM was evaluated in terms of the time that it takes to download operating software sets to a failed NC. Aspects that affect the performance of the DM over and above those affecting the NA include the computing platform of the DM which consists of the PC platform, the peripheral I/O subsystem and the multitasking abilities of the Unix Operating System.

5.5 Performance Measurements Obtained from the Easy Access Network

The Easy Access Network is one type of a Value Added Network in which the elements discussed in this report, namely the NCs, NAs and Domain Manager are implemented. In the Easy Access Network, there exists a Control Centre in which the high level Network Management System known as the Easy Access NMS, and some Network Administrators are located. Both these types of elements can control the entire Easy Access Network from this point, including performing software upgrades, NC force fails, NC load/dump tasks etc.

Because the network is live, there was a limit to the number of test operations that could be performed on the Easy Access Network. It was possible however to perform force fails for those NCs with partner NCs in a back-up configuration, and from that loading time measurements were obtained.

For the purposes of this report, the basic operational network is configured as follows;

- Each of the NCs are loaded via only a single link. The other link is only used as a standby.

- The X.25 links have been upgraded to 19 200 b/s while all the asynchronous links are running at 2400 b/s.

- The X.25 network is configured for modulo 8 operation, with transmission windows of 2 for both send and receive with packet sizes of 128 bytes. The software build sets are all different sizes for each NC, depending on the configuration of the particular NC, with an average of about 14000 blocks.

These operational parameters are set the same for both the NA and the DM.

As stated before in this report, the desired key parameter that was measured was the NC loading time for both the NA and for the DM. For this network, the following results were obtained;

In a back-to-back configuration the loading time varied from about 2.5 minutes to 3 minutes.

When loading a remote NC over the X.25, the following observations were made.

- Minimum loading time about 2.5 minutes,
- Average loading time of just over 3 minutes
- Maximum loading time of just below 6 minutes in conditions of error and depending on load conditions in the X.25 network cloud.

The performance of the DM was also found to be the same as that of the NA, except for local operator functions in which the screen refresh rate was noticeably faster in the DM than in the NA.

Closing Remarks

The analysis performed in this thesis was tailored to accommodate the simulation tool that was used, namely Network II.5. A great effort was devoted to the analysis of the component blocks of the Value Added Network in order to extract the latencies of each as well as to derive the operational parameters in terms of statistical values, the means of which were useful in starting the simulation programs, that in turn allowed further investigation of the more elaborate aspects of the network. Using the models developed using analytical reasoning, the actual simulation programs were constructed in the syntax of Network II.5 and run on an AT computer using a 80286 processor. The validity of the simulation logic and for most of the program structure was tested via two means. In the first case, an extensive study of the performance evaluation of multiprocessor systems was performed and from that models of multiprocessor systems were developed, yielding results that in cases coincided well with predictions made in other published analytical research. The results obtained from these exercises are not included in this report but provided an invaluable insight for the writer into this important technology.

In the second case, the simulation results obtained for the loading time of the NC via the NA and via the DM were compared with the results of actual measurement in the commercial Easy Access Network run by Telkom South Africa. Again a match was obtained, thereby not only giving credence to the logic that was developed in the simulation exercises but also confirming the suitability of the Network II.5 as a general purpose computer and computer network simulation package. Through these validations, one of the key objectives of the thesis was realised.

5.6 The Simulation Package

The simulation work was facilitated by use of the commercial simulation package Network II.5. It took away the responsibility for coding not only the procedures that represent the simulation model itself, but also the supporting procedures such as random number generators, simulation results collector etc. This package is also useful for simulating communication networks in general, apart from simulating computer systems. The results obtained throughout can give confidence in the use of the package as a general purpose performance evaluation tool.

The models are presented together with this thesis as a set of NETWORK programs, each sufficiently documented to enable the reader to follow through the logic, and equipped with a number of handles for fine-tuning the system should particular conditions need to be tested. Long descriptive names have been used, in line with the systems's own clear reserved words and nomenclature. Only minimal annotation is then necessary.

The basics of the package are discussed in Appendix A, and the reader is referred to [C-1] and [C-2] for details. In Appendix A we have also highlighted some of our experiences with the package.

5.7 Meeting Project Objectives

The objectives of this thesis have for the most part been met;-

(a) A good understanding of the issues involved in multiprocessing were obtained, in addition to an exposure to some of the leading published research work in the field.

(b) The project also offered an opportunity to study, develop and test for performance measures in a general computer system or communication network environment.

(c) In work not reported here, a number of useful models for multiprocessing systems were produced and in important cases reproduced results obtained from analytical studies via the simulation method. In this way, it was again possible to enhance confidence in the package as a useful general purpose computer performance evaluation tool.

(d) The work has also been useful in offering exposure to work involved in providing VAN services, including elements of managing data communication networks.

(e) Specifically, quantitative performance indices for certain aspects of the two generations of the Value Added Network; the NCs, NA, and finally the current DM system were derived .

References & Bibliography

A: The Technology of Multiprocessing

[1] A.L DeCegama, *The Technology of Parallel Processing, Vol 1*, Prentice-Hall, 1989

[2] Kai Hwang and Douglas Degroot (Editors), *Parallel Processing for Supercomputers and Artificial Intelligence,* McGraw-Hill, 1989.

[3] Erol Gelenbe, *Multiprocessor Performance,* Wiley, 1989

[4] Allan Burns and Andy Wellings, *Real-time Systems and their Programming Languages,* Addison Wesley, 1989.

[5] H. Stone, *High Performance Computer Architecture,* Addison Wesley, 1989

[6] A.J Van de Goor, *Computer Architecture and Design,* Addison Wesley, 1989

[7] J P Hayes, *Computer Architecture and Organization,* McGraw Hill, 1988

[8] X Zhang, "System Effects of Interprocessor Communication Latency in Multicomputers", *IEEE Micro*, April 1991.

[9] D. Talia, "Message-Routing Systems for Transputer-Based Multicomputers", *IEEE Micro,* June 1993.

[10] F.A Briggs, "Synchronization, Coherence and Event Ordering in Multiprocessors", *IEEE Computer*, February 1988

[11] D. Alpert and M.J Flynn, "Performance Trade-offs for Microprocessor Cache Memories", *IEEE Micro*, August 1988

[12] S. Gudvangen and A G Holt, "Performance models for message passing architectures", *IEE Proceedings*, January 1993

[13] T.N Mudge, J.P Hayes, D. C Windsor, "Multiple Bus Architectures", *IEEE Computer*, June 1987

[14] I.H Önyüksel and K B Irani, "Markovian Queuing Network Models for Performance Analysis of a Single-Bus Multiprocessor System", *IEEE Trans on Computers,* July 1990

[15] H Azaria and Y Elovici, "Modeling and Evaluation of a New Message Passing System for Parallel Processing Multiprocessor Systems", *Parallel Computing*, June 1993

[16] J H Patel, "Multiprocessors with Private Cache Memories", *IEEE Trans on Computers,* April 1982

[17] D W Yen, J H Patel, E S Davidson, "Memory Interference in Synchronous Multiprocessor Systems", *IEEE Transactions on Computers*, November 1982.

[18] J Kriz, "A queuing analysis of a symmetric multiprocessor with shared memories and buses", *IEE Proceedings*, May 1983

B: Communication Networks & Their Management

[1] M. Schwartz, *Telecommunication Networks: Protocols, Modeling and Analysis*, Addison Wesley, 1987.

[2] F. Halsall, *Data Communications, Computer Networks and Open Systems*, Addison Wesley 1992.

[3] D. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall 1987

[4] R E Ziemer and W H Tranter, *Principles of Communication*, Houghton Mifflin 1985

[5] S Haykin, *Digital Communications*, Wiley 1988

[6] M. Gerla and L. Kleinrock, "Flow Control: A Comparative Survey", *IEEE Trans on Communications*, COM-28, 1980

[7] JS Wu and WC Chan, "Performance Analysis of Queuing Networks with end-to-end Window Flow Control", *IEE Proceedings*, Sept 1989

[8] K S Shanmugam, *Digital and Analog Communication Systems*, Wiley 1979

[9] W Bux, K Kümmerle and H L Truong, "Balanced HDLC Procedures: A Performance Analysis", *IEEE Trans on Communications*, November 1980

[10] R E Ziemer and RL Peterson, *Introduction to Digital Communications*, Maxwell MacMillan 1992

[11] D Bear, *Principles of Telecommunication Traffic Engineering*, IEE, 1976

[12] J H Yuen et al, "Modulation and Coding for Satellite and Space Communications", *IEEE Proceedings*, July 1990

[13] J Koljonen, "Managing X.25 Access", *Telecommunications*, September 1991

[14] J Warren, "LAN/WAN integration - exploring the options", *SA LAN Times*, Issue No 41 1993

[15] J Heinänen, "Review of backbone technologies", *Computer Networks & ISDN Systems*, Volume 21-1991

[16] C Smythe, "Networks and their architectures", *Electronics and Communication*

Engineering Journal, February 1991

[17] B T Doshi and P K Johri, "Communication Protocols for high speed packet networks", *Computer Networks & ISDN Systems*, Vol 24 1992

[18] C Partridge, "Protocols for high speed networks: some questions and a few answers", *Computer Networks & ISDN Systems*, Vol 25 1993

[19] I Chlamtac and W R Franta, "Rationale, Directions and Issues Surrounding High Speed Networks", *IEEE Proceedings*, January 1990

[20] IBM Corp, "X.25 Concepts" in *RouteXpander/2 User's Guide and Reference*, 1993

[21] Saponet, "X.25 Packet Switching", *Saponet Technical Publication No 1 Issue 3* November 1986

[22] Televerket Telecom, *Datapak*, 1985

[23] J J Barrett and E F Wunderlich, "LAN Interconnection using X.25 Network Services", *IEEE Network Magazine*, September 1991

C: Simulation Principles and Tools

[1] CACI Products Company, *Network II.5 User Manual*, 1992

[2] CACI Products Company, *Network II.5 Data Structures*, 1992

[3] S. Cheung, S. Dimitriadis, W.J Karplus, *Introduction to Simulation Using Network II.5*, CACI Products Company, 1988

[4] Kobayashi, *Analysis and Modeling*, Addison Wesley, 1978

[5] D. Ferrari, *Computer System Performance Evaluation*, Wiley & Son 1979

D: Network Concentrator & Network Administrator Documentation

[1] M J Ventura, *Amdahl 4404E Software Development*, Communications Research Group, University of Cape Town, 1987

E: General

[1] R. Rector and G. Alexy, *The 8086 Book*, McGraw-Hill, 1980

[2] Intel Corp, *Memory Components Handbook*, 1984

[3] R. Tocci, *Digital Systems*, Wiley 1986

[4] S M Ross, *Introduction to Probability Models*, Academic Press, 1989

[5] A. Silberschartz, J Peterson, P. Galvin, *Operating System Concepts*, Addison Wesley 1991.

[6] Intel Corp, Microprocessors Handbook, 1990

[7] K Haviland and B Salama, *Unix System Programming*, Addison Wesley, 1987

[8] InterActive, *Unix System V/386 Programmer's Guide I*, 1988

[9] S A Rago, *Unix System V Network Programming*, Addison Wesley, 1993

[10] G Barberis, "A useful tool in the theory of priority queuing", *IEEE Trans on Comm*, September 1980.

[11] J R Pinkert and L L Wear, *Operating Systems*, Prentice-Hall 1989

[12] H A Taha, *Operations Research*, Collier McMillan 1987

Appendix A

The NETWORK II.5 Simulation Package

A.1 Hardware Building Blocks in Network II.5

Network II.5 makes available three major generalised building blocks for hardware components in the simulation model. Each of these blocks has a series of attributes whose values must be specified by the user in the input program. The simulated system is put together using the required number of these building blocks. The blocks are as follows

Processing Elements

Represents any device that executes instructions or makes decisions eg CPU, Gateway, Controller. Each PE has its own instruction repertoire which defines the instructions it can execute and describes how to execute them. It may send or receive a message, read or write a file, set, reset or modify the count of a semaphore, perform internal computation. Software modules constitute the workload of a PE, and tell the PE which instructions to execute. The parameters that will be included in the system capture include cycle time and word size etc.

Transfer Devices

This represents any link that moves bits between devices. It defines the connectivity of the simulated system, and serves one PE at a time. A bus access protocol must be specified for each TD , and can be modelled by a number of schemes, of interest to us being First Come First Served, and Priority schemes. All the IEEE 802 (except the 802.6) LAN protocols can be modelled as well. Among the attributes of a transfer device are bus cycle time, bits per cycle, and a connection list.

Storage Devices

Represents any place that stores bits; they have a capacity measured in bits. A storage device may be defined to serve more than one PE simultaneously, by specifying that the SD has *n* ports, where *n* is the number of simultaneous accesses that can be handled by the SD. Storage devices support a file structure.

Other attributes of storage devices include a read word access time, a write word access time, a read word overhead time, read/access delay etc.

A.2 Software Building Blocks

Software Modules

This is the specification of a task to be performed by a Processing Element. It consists of scheduling conditions, host PE options, a list of instructions to execute, a list of module(s) to execute when this module completes.

Instructions Mixes

This is a list of names of instructions, other instruction mixes and Macro instructions with a certain percentage associated with each, the percentages summing up to 100%. This percentage represents the probability of execution of the particular instruction etc each time the IM is referenced.

Macro Instructions

These are a collection of instructions which are referenced by a single name. They can be made up of actual PE instructions, IMs, or other Macro Instructions and are useful for recursion.

Files

Represents the organised storage of information in a storage device, specified by name, size, initial residence and by read only or not attributes.

Messages

Commonly used to schedule and coordinate tasks, ie they represent data dependency among processing elements. A message is sent from one PE to another over a transfer device as a result of a message instruction. Whenever a message is received it is always filed in the PE's Received Message List. Once in that list, modules with that message in their Required

Message List will contend for that message by some form of priority principle.

Semaphores

These are global flags in the model with a value of SET or RESET, and used for synchronization and control among PEs. Scheduling a task by a semaphore differs from scheduling by a message in that

(a) Scheduling is instantaneous (no transfer time)
(b) All modules monitoring the semaphore will immediately take action (messages only start one module).
(c) RUN WHEN and RUN UNTIL semaphore conditions may stop an executing module (messages are strictly preconditions).
(d) Semaphore actions may be timed.

A.3 Some Experiences with Network II.5

1. It may be necessary to use a large set of software modules, which constitute the actual workload of the model. Network II.5 limits both the number of processing elements and software modules to 15 each. For the PEs, this is a reasonable limit for most real applications for commercial multiprocessing systems. However when it is desired to investigate certain limiting cases for multiprocessor systems, then this constraint is severe, and also affects the generalizations that can be made concerning the behaviour of such systems.

   In the case of software modules, this constraint can be overcome by creating the desired number of PE instructions and then incorporating a number of these in a single legitimate module for that particular PE. In addition, Macro Instructions will be found useful in this regard. Then synchronization primitives and preconditions must be applied carefully to maintain the logical process flow.

   The User Manual does not disclose how this parameter can be changed from within the program.

2. The package's results collector is designed to be quite general purpose, with several pages

of results after each simulation run. For example some of the DM exercises yield more than 50 pages of results per run. The person running the simulation must then extract the relevant information and process or integrate it manually with the other results for a broad interpretation.

3. The reader does need a certain minimum time to learn about the package if they require to make big changes to a simulation model. The models presented here though are quite well documented and the main parameters can be easily modified without the need to go through and understand the whole program.

4. Considerable effort and time was expended developing various models of both shared memory and message passing multiprocessor systems using standard performance measures, and from those experiments, comparisons were drawn against analytical results obtained for similar systems as reported in published literature. The congruence of the two sets of results gave us confidence in the use of the package as a suitable simulation tool for computer networks.

Appendix B

B.1 The Exponential Distribution

In our analysis of traffic patterns throughout the presentation we have referred to the exponential distribution of packet interarrival and service times. There exists an important relationship between the Poisson distribution and the exponential distribution. If the Poisson random variable represents the number of arrivals between two successive events, the exponential random variable will represent the time between the two successive arrivals. In effect then the exponential distribution can be derived from the Poisson distribution [E-11].

A continuous random variable is said to have an exponential distribution with parameter λ, $\lambda > 0$, if its probability density function is given by

$$f(x)=\begin{cases}\lambda e^{-\lambda x}, & x \geq 0\\ 0, & x < 0\end{cases}$$

or equivalently if its distribution function is given by

$$F(x)=\int_{-\infty}^{x} f(y)dy=\begin{cases}1 - e^{-\lambda x}, & x \geq 0\\ 0, & x < 0\end{cases}$$

The mean of the exponential distribution, E[X] is given by

$$E[X] = \int_{-\infty}^{\infty} xf(x)dx=\int_{0}^{\infty}\lambda x e^{-\lambda x}dx$$

Integrating by parts with $u = x$ and $dv = \lambda e^{-\lambda x}dx$, we obtain

Also the variance of X

$$E[X] = -xe|_0^\infty + \int_0^\infty e^{-\lambda x} dx$$

$$= \frac{1}{\lambda}$$

$\text{Var}(X) = E[X^2] - (E[X])^2$

But the moment generating function $\phi(t)$ is given by

$$\phi(t) = E[e^{tX}] = \int_0^\infty e^{tx} \lambda e^{-\lambda x} dx = \int_0^\infty \lambda e^{(t-\lambda)x} = \int_0^\infty \lambda e^{-(\lambda - t)x} dx$$

$$= \frac{\lambda}{\lambda - t} \quad for \quad t < \lambda$$

Then

$$E[X^2] = \frac{d^2}{dt^2}\phi(t)|_{t=0} = \frac{2\lambda}{(\lambda - t)^3}|_{t=0}$$

$$= \frac{2}{\lambda^2}$$

Hence

$$Var(X) = E[X^2] - (E[X])^2$$

$$= \frac{2}{\lambda^2} - \frac{1}{\lambda^2}$$

$$= \frac{1}{\lambda^2}$$

The exponential distribution features heavily in the analysis of queuing systems.

B.2 Exponential distribution of arrival of terminal traffic

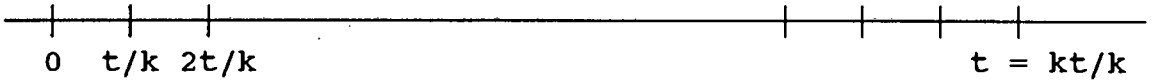
The following is an attempt to characterise this asynchronous traffic. The same argument can be used for X.25 links. The traffic pattern of each terminal can be described by a counting process $\{N(t),\ t \geq 0\}$ where $N(t)$ represents the total number of transmissions that have occurred up to time t. $N(t)$ is assumed to be non-negative and integer valued. Further, if $s < t$ then

$N(s) \leq N(t)$ and $N(t) - N(s)$ equals the number of transmissions that have occurred in the interval [s,t].

The number of transmissions from a terminal are stationary increments, i.e the distribution of the number of transmissions that occur in any time interval depends only on the length of the time interval. The number of transmissions in the interval $(t_1 + s,\ T_2 + s)$ has the same distribution as the number of transmissions in the interval $(t_1,\ t_2)$ for all $t_1 < t_2$ and $s > 0$. This counting process will be a Poisson process having a transmission rate λ, $\lambda > 0$ if [C-4]

(i) $N(0) = 0$
(ii) The process has independent increments
(iii) The number of transmissions in any interval of length t is Poisson distributed with mean λt.

The system under consideration satisfies these requirements according to the following reasoning. Consider an interval of time t, subdivided into k equal parts where k is very large.



Then it can be shown [C-4] that as k increases to ∞, the probability of having two or more transmissions in any of the sub-intervals goes to zero. Hence $N(t)$ will just equal the number of sub-intervals in which a transmission occurs. That is

$$\lim_{k\to\infty} k\,[\lambda\frac{t}{k}+o(\frac{t}{k})] = \lambda t+\lim_{k\to\infty}\frac{[to(t/k)]}{t/k} \qquad (2\text{-}1)$$

$$= \lambda t$$

where a function $f(\cdot)$ is said to be $o(h)$ if

$$\lim_{h\to 0}\frac{f(h)}{h}=0$$

and $t/k \to 0$ as $t \to \infty$.

Hence the parameter λ denotes the arrival rate per time interval.

(b) Distribution of Interarrival Times

If for the Poisson process $N(t)$ we denote the time of the first transmission by T_1 and for $n > 1$, T_n denotes the elapsed time between the $(n-1)^{st}$ and the n^{th} transmission, then the sequence $\{ T_n, n = 1, 2, ...\}$ is the sequence of interarrival times. Now the event $\{T_1 > t\}$ takes place if and only if no transmissions occur in the interval $[0, t]$, and thus

$$\text{Prob}\{T_1 > t\} = \text{Prob}\{N(t) = 0\} = e^{-\lambda t}$$

which is an exponential distribution with mean $1/\lambda$ [E-4]. The exponential distribution is given in Appendix B.

Now $\text{Prob}\{T_2 > t\} = \text{E}[\text{Prob}\{T_2 > t|T_1\}]$

However

$$\text{Prob}\{T_2 > t|T_1 = s\} = \text{Prob}\{\text{ 0 transmissions in } [s, s+t]|T_1 = s\}$$

$$= \text{Prob}\{\text{ 0 transmissions in } [s, s+t]\}$$

$$= e^{-\lambda t} \qquad (2\text{-}2)$$

and T_2 is also an exponentially distributed random variable with mean $1/\lambda$, and furthermore T_2 is independent of T_1. Repeating the same argument would show that the T_n, $n = 1, 2, \ldots$ are independent identically distributed random variables with mean $1/\lambda$. This is the key result we require.

Appendix C

Effective Frame Transmission Times for Go-Back-N

The table below shows values of the effective frame transmission time in ms, for a range probability of frame error values from 10^{-1} to 10^{-7}. Each row represents values for a certain network radius in the range [2, 5, 10, 20, 50, 100, 200] km.

TT (ms)	a	Tavg (ms)			
		Pf = 1E-1	Pf =1E-2	Pf =1E-3	Pf =1E-4
792.85	7.0476	200.5949	120.5086	113.2936	112.57930
792.88	7.0478	200.5981	120.5089	113.2937	112.57930
792.93	7.0483	200.6038	120.5094	113.2937	112.57930
793.03	7.0491	200.6149	120.5104	113.2938	112.57931
793.33	7.0518	200.6482	120.5135	113.2941	112.57934
793.83	7.0563	200.7038	120.5185	113.2946	112.57939
794.83	7.0652	200.8149	120.5286	113.2956	112.57949

T_{avg} calculation (cont)

TT (ms)	a	Tavg (ms)		
		Pf = 1E-5	Pf =1E-6	Pf =1E-7
792.85	7.0476	112.50793	112.50079	112.50008
792.88	7.0478	112.50793	112.50079	112.50008
792.93	7.0483	112.50793	112.50079	112.50008
793.03	7.0491	112.50793	112.50079	112.50008
793.33	7.0518	112.50793	112.50079	112.50008
793.83	7.0563	112.50794	112.50079	112.50008
794.83	7.0652	112.50795	112.50079	112.50008

Appendix D

Program Documentation Notes

This document should be used to interpret any of the programs contained in the NETWORK MODELS disk. The input programs have extension name .NET, while the listing file has extension name .LIS and the plot data file has extension name .PIN. This document gives all the information required to manipulate each program as well as general information that applies to all the programs. This document can also be found in Appendix D in the main thesis document.

(a) General Information

1. Global variables

(i) Batch facility

This feature provides an alternative to the interactive environment where the user is prompted for simulation parameters. All the programs have been set to the batch mode. To begin the simulation, type **network**, at the NetLab prompt. Each of the programs on the disk have been checked and verified and run at once. If any modifications are made before the model is run and an error occurs, a text editor can be used to edit the file.

(ii) *The Simulation Run length*

For those programs that have an iteration period, the simulation run length has been chosen to be at least 10 times the execution time of the largest module. For non-iterating programs, the run time has been chosen in such a way that all the modules run at least once. Any desired time can be used.

(iii) *Plot Data File*

In most of the programs, the creation of the plot data file has been disabled in order to conserve disk space only. If it is required to plot the results of that simulation run, then the

creation of the file must be enabled by setting

PLOT DATA FILE = YES

in the Global Variables.

For the use of other variables in the Global Variables section, please refer to [C-1] and [C-2].

2. Statistical Distributions

This section of the program contains most of the knobs that can be used to fine-tune the variables of the simulation program. Self descriptive names have been used throughout and notes added to each SDF where necessary. Note that most of the values used here, eg memory access times, service times etc have been carefully calculated, and caution must be exercised when making changes, to observe the convergence, independence conditions etc. A lot of the analytical work is hidden in these SDFs, a lot of them used in the CONSTANT type. The various types of SDFs are given in [C-2].

3. The Network II.5 Editor

Whereas any preferred text editor can be used to manipulate these programs, Network II.5 makes available an integrated editing environment that can be used for creating and editing the input programs as well as the listing file. It was found that using a text editor is the most convenient form of creating input programs, more so when these become large, or contain many similar units or building blocks.

The editor can be invoked by typing **edit,** at the NetLab Command prompt.

4. Using the Graphical Environment

Our observation is that this tends to be slow and somewhat cumbersome especially for large models. It is suitable only for very small programs containing only a few building blocks. Also, if the modules must be drawn, the resolution becomes bad for large models. This is in spite of all scaling facilities available.

For these reasons, NETIN [C-1] was used for the creation and verification of all the models. However the graphical models are useful for verifying the correctness of the topology as well as giving a faster view of the network than would be possible from the textual representation of the program.

5. The Listing File

The results are collected and presented in Network's listing file. Numerous values and measures are presented and depending on the application, the relevant portion of the results can be extracted. For each simulation run, a listing file is produced and to obtain trends, the required information must be manually extracted from each file and processed separately.
The graphical complement of the .LIS file is the .PIN file from which results can be plotted either in NETPLOT or in GWORK.

(b) The Individual Programs

On the disk labelled Network II.5 Models are a set of files used follows.

1. CHAP21.NET - NC response time for a single terminal user - local access

2. CHAP22.NET - NC response time for several terminal users - local access

3. CHAP23.NET - User-to-user Response time across network - node delays only

4. CHAP24.NET - User-to-user Response time across network - spurious traffic

5. CHAP25.NET - approximate throughput and hardware utilization of the NC

6. CHAP26.NET - more practical throughput and hardware utilization of the NC

7. CHAP27.NET - practical results for an ATX NC

8. CHAP31.NET - NA loading for a back-to-back configuration 9600 b/s

9. CHAP32.NET - NA loading for a network configuration 9600 b/s

10. CHAP33.NET - NA Loading for back-to-back configuration 19200 b/s

11. CHAP41.NET - DM loading for a single Loader Process instance

12. CHAP42.NET - DM loading for several instances of Loader Process

13. CHAP43.NET - DM loading for a sample network topology - 19200 b/s

Any of these programs can be activated as explained in Section **(a)-1**, above. More information is provided within each of the programs as necessary.

Appendix E

The Go-Back-N protocol with variable K in the NA system

Assume that the link has non-zero error rate. Hence to successfully transmit a frame we need to transmit the same frame an average of N_r times, where N_r is obtained from a knowledge of the bit error rate P_b of the link. If P_b is the probability that a bit will be corrupted then, assuming random errors, the probability P_f that a frame of N_i bits will be corrupted is given by

$$P_f = 1 - (1 - P_b)^{N_i}$$

$$\approx N_i P_b$$

assuming $N_i P_b \ll 1$

Now, if P_f is the probability that a frame will be corrupted, then $(1 - P_f)$ is the probability that an uncorrupted frame will be received. Hence,

$$N_r = 1/(1 - P_f)$$

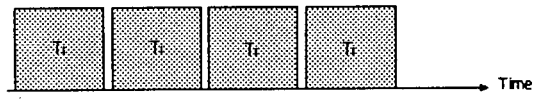
Then we follow Schwartz [B-1] to derive the frame interarrival times at the NC receiver.

Let

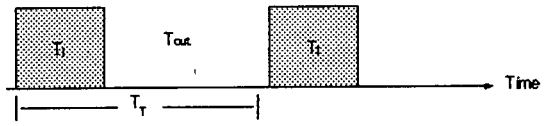
T_I = frame transmission time
T_{out} = time-out interval,
T_{prop} = propagation delay
T_{proc} = frame processing time
T_{ack} = time to transmit ack or nack frame
T_T = time between retransmissions

Consider first the error case, and that the transmitter has sent its full complement of the send window. To derive the average interframe time, use the stop-and-wait case to guide the

analysis, and the diagrams below. Assume that the transmitter always has packets to transmit, then when T_{out} expires, either an ack will be received to advance the send window, or frames



(a) Go-Back-N with no errors



(b) Go-Back-N with errors

will be retransmitted. If P_f is the probability of a frame being received in error at the NC, then the average time for a correct transmission to be received at the NC for a stop-and-wait scheme is given by

$$T_{avg} = T_T + (1 - P_f) \sum_{i=1}^{\infty} i p^i T_T$$

$$= T_T + (1 - P_f) \frac{T_T}{(1 - P_f)} \sum_{i=1}^{\infty} P_f^i$$

where it was used the result

$i = N_r = 1/(1 - P_f)$

from previous.

But if $|x| < 1$, then from the property of the convergence of the geometric series, then

$$\sum_{i=1}^{\infty} x^i = 1 + x + x^2 + \ldots - 1$$

$$= \frac{1}{1-x} - 1 = \frac{x}{1-x}$$

from which it can be obtained

$$T_{avg} = T_T / (1 - P_f) \tag{E-1}$$

where it can be seen that to get to the N_r^{th} retry, the frame must have been delivered in error N_r times. The probability of receiving the frame correctly on the N_r^{th} retry is just $(1 - P_f)$. Assume that acknowledgments are received without error since they are small and thus have a correspondingly small probability of being corrupted. Use the same argument for all the control and supervisory frames such as call request and the response frames, which we assume to have a maximum frame size of 10 bytes as we discuss later.

For the case with no errors the diagram below shows that the time between frames is simply T_I, and the general case where we have line errors becomes

$$T_{avg} = T_I + (1 - P_f)\sum_{i=1}^{\infty} iP^i T_T$$

$$= T_I + \frac{P_f T_T}{1 - P_f} \tag{E-2}$$

$$= T_I \frac{[1 + (a - 1)P_f]}{1 - P_f}$$

where

$$a \triangleq T_T / T_I = 1 + T_{out}/T_I$$

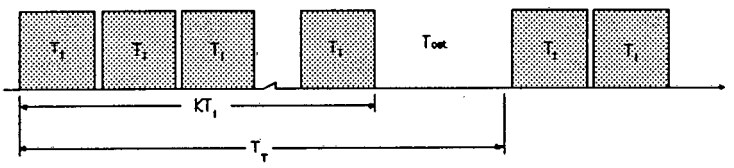
Then if we implement a send window of K, we will have the scenario shown below, where

$$T_T = KT_I + T_{out_{w-s}} \tag{E-3}$$

where

$$T_{out} = 2T_{prop} + T_{proc} + T_{ack} \qquad \text{(E-4)}$$

and the terms have been defined above.



(c) Go-Back-N with a general K

The assumption here is that in the Go-Back-N scheme, if a NACK is received before the send window is full, the transmitter will continue to send *I*-frames and then retransmit the corrupted packet in the $(K + 1)$ frame. Assume also that when the full window of frames has been sent, the transmitter waits for a duration equal to the timeout interval. The result is to give the upper-bound of the effective frame transmission time for the case under consideration. The effect of the separation distance between the NA and the NC is included through the term T_{prop}.

Appendix F

Analysis of process scheduling time in the DM system

Consider class p ($1 \leq p \leq r$). Let a process of this class arrive at the CPU at an arbitrary time t_0. Its random waiting time W_p measured from its arrival time until it enters service, is due to contributions from 3 sources;

- It must wait a random amount of time T_0 until the processes currently executing complete service.

- It must wait a random number T_k units of time until all the processes of priority k lower than or equal to p already queued at the arrival time t_o, complete service.
- Finally, it must wait a random time T'_k to service customers of each class k of priority lower than p arriving during the wait time W_p.

Combining terms we have

$$W_p = T_0 + \sum_{k=1}^{p} T_k + \sum_{k=1}^{p-1} T_k'$$

Following Schwartz [B-1], take expectations term by term to obtain the average waiting time

$$E(W_p) = E(T_0) + \sum_{k=1}^{p} E(T_k) + \sum_{k=1}^{p-1} E(T_k') \qquad \text{(F-1)}$$

Now $E(T_k)$ is due to an average number $E(n_k)$ processes of class k waiting in the system. Each requires $1/\mu_k$ units of service on the average, so that

$$E(T_k) = \frac{E(n_k)}{\mu_k}$$

But using Little's formula, presented in Section 2.3, we have

$$E(n_k) = \lambda_k E(W_k)$$

so that

$$E(T_k) = \rho_k E(W_k) \quad ; \qquad \rho_k \equiv \lambda_k / \mu_k$$

The term $E(T'_k)$ is due on average to $E(n'_k)$ processes of class k arriving during the interval $E(W_p)$. Since the arrival rate is λ_k and each process again requires on average $1/\mu_k$ of service, then

$$E(T'_k) = \lambda_k E(W_p)$$

The term $E(T_o)$ is the residual service time of a process that is still in service, and is independent of the queue discipline - it must be the same if processes of all k classes are served with equal priority in their order of arrival. The system is approximated by a single server with general service time and Poisson arrivals, i.e $M/G/1$, queue. If $E(\tau) = 1/\mu$ is the average service time and σ^2 is the variance of the service time distribution, then the second moment of the service time distribution is

$$E(\tau^2) = \sigma^2 + \frac{1}{\mu^2}$$

giving an average wait time $E(W)$ in an $M/G/1$ queue as

$$E(W) = \frac{\lambda E(\tau^2)}{2(1 - \rho)}$$

from which

$$E(T_o) = \lambda E(\tau^2)/2 = \sum_{k=1}^{r} \lambda_k E(\tau^2_k)/2$$

and finally that

$$E(W_p) = \frac{E(T_o)}{(1 - \sigma_p)(1 - \sigma_{p-1})} \qquad \text{(F-2)}$$

where

$$\sigma_p = \sum_{k=1}^{p} \rho_k \quad , \qquad \rho_k = \frac{\lambda_k}{\mu_k}$$

Then solve recursively to obtain the waiting times for any class as follows.

$$E(W_1) = \frac{E(T_o)}{1 - \rho_1}$$

$$E(W_2) = \frac{E(T_o)}{(1 - \rho_1)(1 - \rho)}$$

$$= \frac{E(W_1)}{1 - \rho}$$

with $\rho = \rho_1 + \rho_2$

by proceeding in a similar way for all the priority classes.