# List of Tables

# List of Figures

# Contents

# ACKNOWLEDGEMENTS

I wish to acknowledge the guidance and suggestions of my course mentor Mr Neill Harris, throughout the duration of this research.

I would like to acknowledge the help offered by the other members of the Metroring group. The Metroring group consists of Mr Neill Harris, Mr Bradley Dahl, Mr Nik Tjirkali and myself.

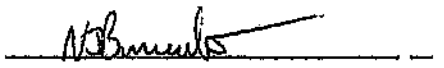
I would also like to acknowledge WEB Systems (Pty) Ltd, Second Floor, Hatfield Gardens, Arcadia for the use of their simulation facilities and the assistance of Mr Mark Chodos, Programme Manager at WEB Systems.

# ABSTRACT

In recent years organisations have become heavily dependant upon their capability of exchanging information rapidly using computer networks. Consequently, the number of networks and internets has proliferated dramatically, resulting in an increased volume of information being transported over computer networks. The Metroring networking protocol has been specifically designed to offer high reliability while coping with heavy loading. The Metroring architecture incorporates concurrent transmissions, a ring topology, full-duplex links and uses fibre optic technology. This dissertation presents a simulation model of the Metroring system constructed in the object-oriented paradigm. The model is used to evaluate the performance characteristics of the protocol. The efficiency of Metroring does not degrade as physical size increases. High effective throughput compared with current protocols makes Metroring suitable for a wide range of modern, bandwidth-intensive applications.

# DECLARATION

I declare that this dissertation is my own, unaided work. It is being submitted for the Degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg It has not been submitted before for any degree or examination in any other University.



(Signature of candidate)

29 day of April 1994

# PERFORMANCE ANALYSIS OF THE METRORING NETWORKING PROTOCOL

N.J.BURMEISTER
8801233/X

A DISSERTATION SUBMITTED TO THE FACULTY OF ENGINEERING, UNIVERSITY OF THE WITWATERSRAND, IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN ENGINEERING.

JOHANNESBURG, 1994

As with throughput, delay may be normalised. Using the notation from the discussion of throughput, the average link transmission time is $\frac{T}{R}$ seconds per frame. If we denote average transfer delay as $T'$, the normalised delay becomes,

$$T' = \frac{RT}{T}$$

In addition to these indices, other variables exist which reflect network performance. Shaikh (1989) lists message survivability as an important criteria in evaluating network performance. He defines this variable to be "the ratio of correctly delivered messages over all the messages that can be delivered over a given time interval." (Shaikh 1989).

Schoemaker (1978) identifies a two additional performance measures: reliability and the probability of transmission errors and failures. Although reliability can be defined in a number of ways, Schoemaker states that reliability usually expresses the probability that two users can connect to each other. In a data communications system transmission errors mean a re-transmission of data. This imposes an additional load on the network. The size of this load, and thus the impact on performance, is dependent on the error rates and the efficiency of the error correction scheme.

Another performance measure closely related to throughput is **utilisation**. Broadly speaking, utilisation is defined as the ratio of the actual usage of a channel to the total possible usage of the channel (Burmeister, 1992). Harris and Burmeister (1992) and Hammond (1986) give an in depth discussion of utilisation definitions

*Throughput*

Tanenbaum (1988) defines throughput as the number of bytes of user data transferred per second, measured over some recent time interval. Hammond (1986) follows this definition, although he uses units of bits per second as opposed to bytes per second, adding the stipulation that the measure of throughput should only include error-free packets. In their definitions, Sunshine (1987) and Cobb (1992) are more specific as to the point in the network where throughput should be measured. They define throughput as the transmission rate of data per unit time. Sunshine (1987) also stipulates that the throughput rate should exclude any control information or retransmissions required by the protocol. Boggs (1987) and Hammond (1986) normalise the throughput parameter with respect to bandwidth. Boggs defines throughput as the fraction of the nominal bandwidth actually used for carrying data. Unlike Sunshine, Boggs includes packet headers in this measure.

Hammond (1986) normalises throughput on a single channel with respect to the channel transmission rate. For example, if the data input rate to a section of the network is $\lambda$ frames per second, the link transmission rate is $R$ bits per second, and there are $\bar{X}$ bits per frame on average, then the *unnormalised throughput is just* $\lambda\bar{X}$ *bits per second*. The normalised throughput, $S$, is a dimensionless quantity and is given by

$$S = \frac{\lambda\bar{X}}{R}$$

Throughput may apply to a single link as well as to an entire network. If the network consists of multiple channels, then the total unnormalised throughput will merely be the sum of the throughput on the individual channels (Hammond, 1986).

*Delay*

The second important performance index is delay, also known as **end-to-end delay**. Sunshine (1987), Boggs (1989) and Hammond (1986) define delay as the time from starting to transmit a packet at the sending node to the successful arrival of the entire packet at the receiver.

## 2.3 Simulation Applied to Computer Networks

The complexity of a computer network can make performance prediction difficult. Computer network performance is sensitive to a number of different factors: protocol, topology and traffic characteristics being the most important (Frost, 1990). Simulation plays an important role in the modelling, analysis and design of computer networks because it offers a flexible approach to performance evaluation. Simulation enables the prediction of changes in performance under various conditions (Shanmugan, Frost).

Roth, Ilyas and Mouftah (Roth, 1992) describe discrete event simulation as a system of moving objects traversing waiting lines and making use of fixed resources in some sequential order. This description is very similar to that of a communications network where data frames traverse links and queues, waiting for service by nodes with resources such as buffer space and processors.

The operation of a computer network may be modelled on many different levels of abstraction, from the level of individual bits to the handling of a user application session. Simulation models are able to reflect this variation in detail. A simulation model may be constructed at a suitable level of complexity to achieve the required aims of the simulation. For example, if a simulation model is derived to study the performance of an entire computer network, it may not be necessary to model the operation of the individual networking cards in detail. In this dissertation the operation of queues under the Metroring protocol is studied.

## 2.4 Performance Definitions

The literature offers a number of performance measures as well as several definitions for each of these measures. The two main measures of performance in a computer network are **throughput** and **delay** (Schoemaker, 1978), (Shanmugan, 1989).

and their analysis and description. However, these methods are not particularly useful in modelling and simulating the operation of large computer networks (Shanmugan, 1992). Queuing models, on the other hand, have been used extensively in both analytical and simulation modelling of networks (Woodside, 1989). Queuing models are better able to model topological aspects of computer networks because the flow of information in a network is better visualised in the queuing paradigm. For these reasons the majority of network simulators have used the queuing paradigm as the basis for their models. Hence the queuing paradigm forms the basis of the Metroring model developed in this investigation.

LAN simulation models are usually concerned with the Media Access (MAC) layer, and the Logical Link Control (LLC) layer (e.g. (Frost, 1990)). LAN simulation models using special purpose languages possess a limited application as the model is usually specific to a particular protocol. The investigation of a specific problem in networking frequently provides the justification for developing such simulations.

Due to the relative infancy of MANs in comparison to LANs, there has been considerably less activity in the simulation of MANs. MAN simulation models, like the LAN models, generally do not model layers higher than the LLC layer.

Recently simulation models have been developed to be more general in their application, possessing the ability to model a number of different types of LANs. This is especially true of simulation models of internets. These models must be capable of modelling a number of different types of LANs. The general simulation models provide a set of "primitives", i.e. basic building blocks such as queues, which are used to construct a customised simulation model, for example BONeS (Shanmugan, 1992), and GLAS (Frost, 1990). The basic structures of these models are written in general purpose languages such as C and FORTRAN because of the added flexibility and ease of extension provided by these languages. The user is provided with a customised set of building blocks with which to create a model. Whereas LAN simulation models generally incorporate only layers 1 and 2 of the OSI reference model, simulation models of internets also have to model the networking layer to cater for the routing functions of routers and transparent bridges.

The MetaRing protocol features full-duplex links, spatial reuse and concurrent transmissions. From this point of view the MetaRing has much in common with the Metroring protocol. MetaRing uses two access modes, slotted and buffer insertion. Although the slotted mode reduces buffer delay, the frame size is fixed and a node may only transmit when it receives an empty slot. The buffer insertion mode of MetaRing is similar to the operation of Metroring. A brief comparison of FDDI, DQDB, MetaRing and Metroring is given in table 2.1.

| | FDDI | DQDB | MetaRing | Metroring |
|---|---|---|---|---|
| Medium | Optical Fibre | Optical Fibre | Optical Fibre | Optical Fibre |
| Topology | Ring | Bus | Ring | Multiple Rings |
| Physical Topology | Dual counter-rotating rings | Dual contra-directional buses | Full duplex point-to-point links | Full duplex point-to-point links |
| Transmission Rate | 100 Mbps | 150 Mbps x 2 busses | 100 Mbps per link | 100 Mbps per link |
| Spatial Reuse | No | No | Yes | Yes |
| Media Access Method | Permission Token | Slot Requests | Hardware control signal | Autonomous link control |

Table 2.1: A comparison of the FDDI, DQDB, MetaRing and Metroring protocols

## 2.2 A Survey of Present Simulations

In the last two decades simulations have been used to study the operation of a wide range of computer networks. The majority of these simulation models are created using the event driven approaches described in section 3.2. Early network models were generally constructed using special purpose simulation languages like SLAM, while more recent models have concentrated on modelling networks in the paradigms of Queues, Finite State Machines and Petri Nets (Shanmugan, 1992). Petri Nets and Finite State Machines are well suited to the formal verification of protocols

# Chapter 2: Survey of Current Developments

This chapter covers work related to current developments in high speed networks and the modelling of computer networks, as well as current trends in network simulators. A survey of performance measures is included in this chapter.

## 2.1 Current Developments in LAN/MAN Protocols

FDDI (Fibre Distributed Data Interface) and DQDB (Dual Queue Dual Bus) are two recent protocols to appear in the LAN/MAN arena. Much of the data used in performance studies of these networks has been derived from simulation studies (Dahl, 1993). Unfortunately, apart from general performance comparisons these studies have little relevance to the Metroring protocol. Neither FDDI nor DQDB allow concurrent access to the physical medium and neither protocol employs spatial reuse. **Spatial reuse**, or the ability to provide concurrent transmission over disjoint segments, is a fundamental characteristic of Metroring.

The recently developed Parallelring (Qu, 1992) and MetaRing (Cidon, 1993) are two protocols appearing in the literature that do feature concurrent multiple transmissions. Parallelring is a Token Ring LAN that has the ability to support concurrent multiple communication paths on a single loop. Parallelring is different from Token Ring LANs in two main aspects:

1. Frames are removed from the ring by their destination stations rather than their source stations.
2. A Parallelring station may initiate a message transmission not only upon receipt of the token, but also upon receipt of a data frame addressed to it (Qu, 1992).

behaviour of the protocol. A discussion of the design of the Metroring simulation model is given in Chapter 5. Relevant aspects of queuing theory incorporated in the model are also reviewed in Chapter 5. Chapter 6 details the implementation of the Metroring model using Object-Oriented Programming techniques in the C++ paradigm. Chapter 7 presents a discussion of the results yielded by the Metroring model. The results produced by the model are compared with other simulation results and the arguments of the dissertation are concluded in Chapter 8. Proposals for future research and development of the Metroring model are also included in Chapter 8.

Appendix A: an overview of the OSI model and comparison with the Metroring protocol stack.
Appendix B: the Metroring error control, routing and flow control mechanisms in detail.
Appendix C: the Poisson, Uniform and Exponential distributions and the relevant formulae for the M/M/1 queue used in the Metroring simulation.
Appendix D: additional explanations of the object-oriented concepts of abstraction, encapsulation and modularity introduced in Chapter 6 as well as the Booch notation for representing object-oriented software design.
Appendix E: the primary reasons for using C++ as the implementation language of choice for this simulation study.
Appendix F: a brief description of the most important classes in the Metroring software together with several of their respective methods.
Appendix G: the results discussed in Chapter 7.

The model and its results are restricted to the case of Metroring networks. However, because the initial model of the Metroring protocol describes the behaviour of the protocol at several levels of complexity, it may be applied to various other networks at lower complexity levels. At more complex levels, i.e. levels containing greater detail such as I/O buffering, models are developed specifically for a particular protocol.

Several limitations and assumptions are imposed on the scope of this research:

1. Only the physical layers up to the MAC level are considered in this investigation. Issues influencing performance at higher levels are difficult to predict as the Metroring protocol is still undergoing development.
2. Performance characteristics are applicable to **steady state** operation only (i.e. a static topology without any link or node failures) for the duration of the simulation.

Detailed limitations and assumptions are given in section 5.3.3, section 6.3.2 and section 7.1.

## 1.4 Validation of Results

Results from operating Metroring networks are not currently available for comparison purposes because this research forms part of the Metroring development programme. The final results of the simulation model are validated in two ways. Firstly, simulation results are compared with analytic estimates of performance. Secondly, simulation results are compared with those obtained using different simulation models. These comparisons appear in Chapter 7.

## 1.5 Outline of the Dissertation

Chapter 2 presents an survey of current developments in computer network simulation. Relevant background information on computer networks and simulation techniques is provided in Chapter 3. The Metroring protocol is described in Chapter 4 with a discussion of the operation and

## 1.1 Research Objectives

The primary objective of this research is to investigate the performance of the Metroring networking protocol. It is important to gain some indication of the performance characteristics of Metroring before the protocol is implemented.

A model of the system is developed to predict the performance of the networking protocol under various load conditions and network configurations. This model should be constructed at a suitable level of complexity to examine general performance characteristics. The model needs a certain degree of flexibility as the Metroring protocol is still evolving. The principal aim in constructing a simulation model is to capture the essential operation of the Metroring protocol. Construction of a simulation model will also prompt changes to the protocol specification

## 1.2 Proposed Solution

Simulation is used to construct a model of the Metroring protocol. Simulation, in particular discrete-event simulation, is particularly well suited to the modelling of computer networks. Roth (1992) describes discrete-event simulation models as "moving objects traversing waiting lines and making use of fixed resources in some sequential order." This description applies equally well to a computer network. A computer network essentially consists of a number of frames traversing links between stations on the network.

In addition, simulation is well suited to the requirement of flexibility since simulation requires few assumptions and approximations (Shanmugan, 1992).

## 1.3 Limitations of the Research

# Chapter 1: Introduction

In recent times the productivity of organisations has become heavily dependant on their capability to exchange information rapidly (General DataComm, 1993). This fact has resulted in a proliferation of computer networks (Sunshine, 1990) as well as a dramatic increase in the amount of data being transferred on these networks. The move towards an information-intensive society, an ever increasing number of interconnected computing devices and an escalation in the volume of data being transferred over computer networks necessitates the need for more efficient communication networks. The Metroring networking protocol has been specifically developed to provide a reliable, high speed, internet configuration suited to heavy loading.

The sensitivity of a protocol to changes in network topology, buffer sizes, error rates, the number of devices attached to the network, and the traffic load imposed on the network, is becoming increasingly important in protocol design as data transfer speeds and configuration complexities increase. In this study the performance of the Metroring protocol is evaluated using a simulation model.

Simulation is a powerful tool in the analysis of communications protocols because of their complexity which often render mathematical models unfeasible. Other modelling methods such as Petrinets are used to analyse the operation of protocols, but are not suitable for performance analysis (Shanmugan, 1992). The strengths and weaknesses of the protocol emerging from simulation can be used to direct further development.

LCP - Link Control Protocol

MAC - Media Access Control

MAN - Metropolitan Area Network

Mbps - Mega bits per second

MP - Management Protocol

NACK - Negative Acknowledgement

OSI - Open System Interconnect

pdf - probability distribution function

pmf - probability mass function

RAM - Random Access Memory

RCP - Routing Control Protocol

RQ - Repeat Request

SEN - Segment End Node

WAN - Wide Area Network

# List of Acronyms

ACK - Acknowledgement

ARQ - Automatic Repeat Request

bit - binary digit

bps - bits per second

CPU - Central Processing Unit

CRC - Cyclic Redundancy Check

cdf - cumulative distribution function

DMA - Direct Memory Access

FCS - Frame Check Sequence

FDDI - Fibre Distributed Data Interface

IEEE - Institute of Electrical and Electronic Engineers

GUI - Graphical User Interface

ISO - International Standards Organisation

LAN - Local Area Network

2. The duration of the simulation is determined by some parameter value in the simulation run. For example, when simulating a queue of people at a bank teller, one might choose to stop the simulation after 10 clients have been served.
3 The simulation is terminated once a required level of accuracy has been reached.

The Metroring simulation currently uses the first stopping condition, i.e. the simulation runs for a length of simulated time specified by the user.

| Event | Description | Time |
|---|---|---|
| 1 | Receive | 0.02 |
| 3 | Generate | 0.15 |
| 2 | Transmit | 1.56 |
| 4 | Stop | 2 |

Table 3.2: A Future Events List

The following chapter gives an overview of the Metroring protocol and discusses the operation of the protocol in detail.

*Discrete Event Simulation*

Real-world systems may be categorised into two basic groups: **continuous** and **discrete** systems. In continuous systems the state variables change continuously over time. In discrete systems, however, the state variables change only at discrete instants in time. **Events** are instantaneous occurrences which may change the state of a system at discrete points in time.

In a communications system different events are constantly occurring. Simulation of such a system requires the maintenance of a structure, called a **future events list**, that will determine what is to happen next during the simulation. The times of various events can be stored in the future events list ensuring that they are performed in the correct chronological order. A future events list containing typical Metroring events is shown in Table 3.2.

A means of recording the passage of time is also required. This is achieved via the **clock**. The first entry in the future event list is known as the most imminent event. After the system state and simulation clock have been initialised, and the future events have been computed and placed in the correct order on the future event list, a typical discrete event simulation would proceed as follows:

1. The simulation clock is advanced to the time of the most imminent event.
2. The system state is modified according to the most imminent event.
3. Knowledge of event times scheduled to occur in the future, is updated.
4. The simulation clock is advanced to the most imminent event and the procedure is repeated.

The simulation is terminated upon reaching a **stopping condition**. The process of advancing the simulation clock from one event to the next continues until a stopping condition is reached. Three stopping conditions generally exist:

1. A stopping time in simulated time units is specified by the user before the simulation begins.

1. Simulation allows the study of, and the experimentation with, the internal interactions of a complex system (Banks, 1984), which is the precise requirement in the Metroring protocol design.
2. Changes in the input data and the structure of the communications system can be made, and the effects of those changes on the performance of the system may be studied.
3. By varying input parameters to the model and examining the resulting changes in the output parameters, valuable insight may be gained as to which variables are the most critical and how changes in certain variables affect others.
4. Simulation methods are generally easier to apply than analytic methods. In the case of Metroring, it is currently the only means of deriving a solution, since an analytic model has not been developed.
5. Few assumptions and approximations are required in a simulation (Shanmugan, 1992).
6. Simulation can provide a rapid means of analysing and refining performance of the system during the design phase (Shanmugan, 1992), (Roth, 1992). This is very relevant to the Metroring protocol which is still undergoing development.
7. Developing a simulation model aids in the understanding of the operation of the Metroring protocol, and hence may result in improvements to the protocol.
8. The simulation model may be used repeatedly to analyse proposed design changes to the Metroring protocol.

***Simulation Terminology***

The terminology used by Law and Kelton (1982) and Banks and Carson (1984) is adopted throughout this dissertation. The definition of a system given above may be expanded as follows, "A system is defined as a group of objects that are joined together in some regular interaction or interdependence toward the accomplishment of some purpose" (1984). This definition may easily be applied to elements such as nodes within a Metroring, as well as to the protocol itself. The objects of interest in the system are called **entities**. In the Metroring case, a node would constitute an entity. An **attribute** is a property of an entity. Again, expanding on the example of a Metroring node, the processing time of the node would be an attribute of the node. The **state** of a system is that set of variables necessary to describe the operational state of a system at a particular time.

required. According to Sauer (1983), two methods for modelling computer communication systems exist. These two methods are **analytic modelling** and **simulation**.

The method of analytic modelling views the system to be modelled at a sufficient level of abstraction so that probability theory and other tools of applied mathematics may be used to develop equations which characterise the system. However, as Sauer points out (1983), often the required level of abstraction in analytic models is so high that their results are questionable. Haenle and Giessler (Schoemaker, 1978) point out that the complexity of communication systems restrict the analytic methods used for their investigation to rather simple and idealised systems.

The alternative to analytic modelling is simulation. The principle advantage of a simulation model over an analytic model is its generality. A simulation model may be applied to a wide range of systems, whereas an analytic model is specifically developed for a single system. Whereas the analytic model requires a high level of abstraction, essentially any level of detail may be incorporated in to a simulation model (Sauer, 1983).

### 3.2.2 Simulation Modelling

"A simulation is the imitation of the operation of a real-world process or system over time" (Banks, 1984). A system is the facility or process of interest (Law, 19882). An example of a system would be a Metroring node. Typically, real-world systems are very complex. In fact in most cases they are too complex to analyse intuitively or mathematically. In such cases, simulation can be a powerful and effective analytical tool. Simulation is particularly applicable in the case of analysing the performance of the Metroring protocol as it is a new protocol that is still being developed. The impact on performance of potential changes to the protocol can be assessed using simulation.

(Banks, 1984), (Shanmugan, 1992) and (Roth, 1992) each list purposes for which simulation is particularly apt. The following uses for simulation are applicable in simulating a Metroring:

The single network resulting from the connection of two or more networks is known as an internet. The transfer of information within any one of the networks comprising an internet is defined by a set of protocol layers (Schoemaker, 1978). In order for the internet to provide a single, integrated network, it is responsible for resolving incompatibilities resulting from the different protocols, topologies and technologies of the networks that comprise it. To overcome the compatibility problems of different networks, ISO (International Standards Organisation) developed a layered reference model, which separates the functions of a communications protocol into seven, clearly defined layer[illegible]ver reference model is commonly known as the OSI (Open Systems interconne[illegible] model is relevant to this research as it is as a basis for the description of the [illegible]ing protocol. The OSI protocol stack is also used to define internets and give its functions. Layers 1 to 3 of the OSI model deal with the network environment, i.e. the protocols and standards relating to the underlying communications network, while layers 4 to 7 add additional application-orientated (network independent) protocols. The function and operation of each layer of the OSI model is explained in Appendix A.

## 3.2 Simulation

Simulation is particularly apt in the case of Metroring because of complexity as mentioned in section 2.3. The methods and the approach adopted in developing a model for simulation have a major bearing on the structure of the model and the results obtained from simulation experiments. In this section the principles of simulation are introduced.

### 3.2.1 Analytic and Discrete Event Simulation Modelling of Communications Systems

In many instances a need arises to estimate the performance of a system being designed or developed before that system is implemented. For this reason a method of modelling the system is

Table 3.1 summarises the characteristics of LANs, MANs and WANs[2].

### 3.1.5 Metroring

Following the network classifications in the preceding sections, Metroring falls into the general classification of a MAN. However, a Metroring may also connect several stations in close proximity to one another forming a high speed LAN. Commonly a Metroring network would connect several LANs performing routing between stations. In this case a Metroring must be classified as a MAN, not a LAN. The Metroring protocol is described in detail in Chapter 4.

| *Characteristic* | *LAN* | *MAN* | *WAN* |
|---|---|---|---|
| Physical Dimensions | 0-1 km | 1-100 km | 10-10000 km |
| Data rate | 1-100 Mbps | 1-100 Mbps | 0.1-100 kbps |
| Delay | 1-100 msec | 1-100 msec | >500 msec |
| Probability of error | $<10^{-9}$ | $<10^{-9}$ | $10^{-5}$-$10^{-6}$ |

Table 3.1 Comparison of LAN, MAN and WAN characteristics

The OSI stack will be used to describe the Metroring protocol. The concept of an internet is also important to the Metroring description. For these reasons internets and the OSI reference model are described in the following section.

### 3.1.5 Internets and OSI

[2] Values reflected in the table are derived from (Smythe, 1991) and (Tanenbaum, 1988).

associated with each device, i.e. a router will have to perform more functions than a relay will and will therefore require more time.

### 3.1.3 The Wide Area Network (WAN)

WANs are networks which span a large geographical area, typically across international boundaries. WANs are often used to connect several LANs which are physically separated by large distances. WANs use point-to-point links and are characterised by relatively low data rates and high error rates.

In addition to the LAN and WAN, the Metropolitan Area Network (MAN), forms a third type of network. In terms of physical dimensions, the MAN falls roughly in between the LAN and WAN classifications. Tanenbaum (1988) describes a MAN as a network which spans a city using LAN technology.

### 3.1.4 The Metropolitan Area Network (MAN)

Basically, a MAN is a network providing high speed (greater than 1 Mbps) connectivity over distances characteristic of a metropolitan area (Klessig, 1986). Because the interconnection of LANs formed the early MANs (Smythe, 1991), many of the technical concepts underlying MANs are derivatives from LAN concepts (Klessig, 1986). MANs typically connect a number of LANs on a city-wide scale using LAN technology.

A MAN has a structure similar to a distributed LAN. Smythe (1991) states that the characteristic performance of the MAN is similar to that of the LAN although delay times are higher in MANs.

The DQDB (Dual Queue Dual Bus) is an example of a MAN protocol.

A number of distinct LAN topologies exist. The three most common topologies are the star, ring and bus topology. These topologies need not be physical, they may only be logical. For example, it is possible to implement a logical ring topology over a network which has a physical star configuration.

Currently much emphasis is being placed on the interconnection of LANs. LAN interconnection is achieved using a number of devices which are briefly introduced in the following section.

*LAN Interconnection Devices*

The interconnection of LAN segments is achieved by using four basic connecting devices: repeaters, bridges, routers and gateways. The generic term for these devices is **relay**.

*Repeaters.* Repeaters merely replicate an electronic signal between different network segments. They operate at layer 1, the physical layer, of the OSI model. Segments connected by repeaters do not constitute an internet. They only serve to increase the physical extent of the LAN.

*Bridges.* Bridges operate at layer 2 of the OSI model. They provide a means of connection between different LANs. A bridge is able to store a message until the destination node can receive the message. This is known as the "store and forward" ability of bridges. It is this facility which allows bridges to break the timing sequence between network segments (Harris, 1990).

*Routers.* Routers operate at the networking layer, layer 3 of the OSI model. The primary function of layer 3 is the routing of frames (see Appendix A). Routers are able to route messages to specific destination LANs. In some cases routers are able to avoid links that are broken or congested.

*Gateways.* The final connection device is a gateway. Gateways operate at the application layer (layer 7) and are able to perform high level protocol conversions. Gateways would commonly form LAN-WAN or WAN-WAN interfaces.

The data throughput rate of these relays becomes progressively slower as one moves higher up the layers of the reference model. The decrease in throughput is due to the relative complexity

1. Miniature (<5cm). Computational elements implemented on the same integrated circuit.
2. Small (<50cm). Interconnection of computational elements which are located in the same piece of equipment.
3. Mean (<1km). These networks connect computing equipment within the same localised area such as a university campus or a factory. A Local Area Network (LAN) would fall into this category of computer network.
4. Large (>10km). Such networks connect computing equipment distributed over a wide area (e.g. nation wide or even world wide). Wide Area Networks (WANs) fall into this category.

Because of their small physical dimensions, the first two kinds of networks often share memory. These networks typically only pass memory pointers between elements as opposed to complete messages. Hence, they are known as **closely coupled** networks. Networks 3 and 4, physically pass entire messages between computer devices. They are known as **loosely coupled** networks. Metroring is therefore classified as a loosely coupled network.

### 3.1.2 The Local Area Network (LAN)

With the advent of microprocessor technology and integrated circuitry, the number of autonomous computer devices increased rapidly. Although each of these devices commonly possessed some processing power, devices in the same locality often needed to exchange information and share a single resource (e.g. a mainframe computer). In order to facilitate these requirements, the LAN was developed.

LANs are characterised by their high data throughput, and low delays. LAN data rates are at least several million bits per second (Mbps)[1]. LANs are also generally very reliable, i.e. they have low error rates.

---

[1] Mbps = mega (million) bits per second. MBps = mega bytes per second

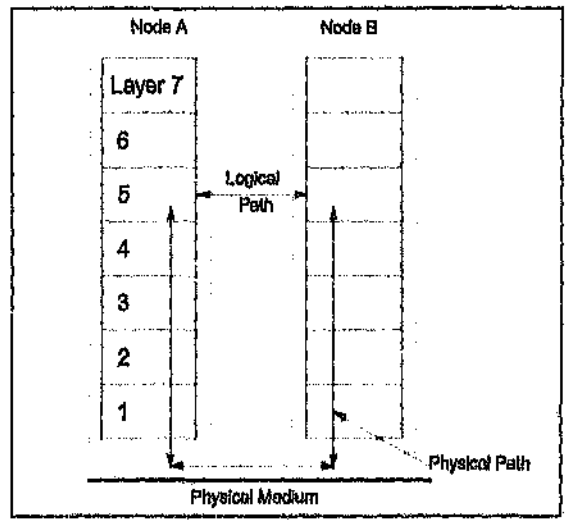



Figure 3.2 Peer-to-peer Communication

### 3.1.1 The Interconnection of Computer Equipment

There are a number of reasons which make the interconnection of computer equipment desirable:

1. The sharing of scarce or expensive resources.
2. The sharing of data.
3. Load sharing.
4. Communication purposes.
5. Provision of redundancy for reliability purposes.

The networks resulting from this interconnection of computing equipment can be classified according to their physical dimensions.

#### *Classification*

Halsall (1988) classifies computer networks into four main categories according to the physical separation of the computing devices:

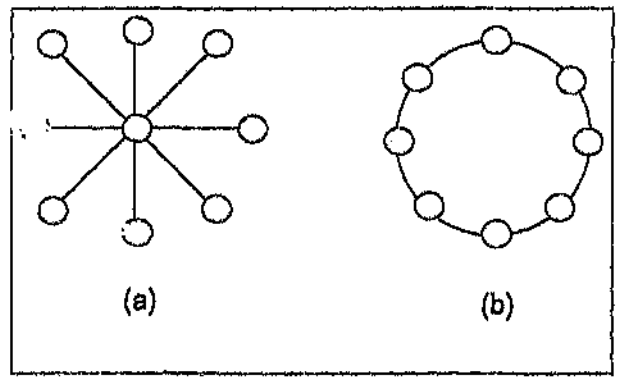



Figure 3.1 (a) Star topology, (b) Ring topology

Computer networks are designed in a structured manner. Most networks are organised as a series of layers (Tanenbaum, 1988). Each layer is designed to offer services to higher layers, and to shield higher layers from the implementation details of those services. This layered structure is shown in Figure 3.2. Inter-node communication takes place between corresponding layers, and is called **peer-to-peer** communication. For example, layer 5 in node A communicates with layer 5 in node B, as shown in Figure 3.2. The path of communication shown in the figure is only a logical one. In reality any messages passed between the two nodes will be transferred across the physical medium. The set of rules and conventions which govern conversations between peer layers is known as a **protocol** (Tanenbaum, 1988).

# Chapter 3: Background

Relevant terms used in computer networking are defined in section 3.1 of this chapter. Section 3.2 provides a background to simulation. Section 3.1 and 3.2 provide the basis for the discussion of the Metroring protocol and the Metroring simulation model presented in Chapters 4 and 5 respectively.

## 3.1 Network Terminology

Tanenbaum (1988) defines a computer network as "an interconnected collection of autonomous computers". This definition is appropriate in this dissertation since each Metroring node is an autonomous computer. Transmission lines, also called **links** or **channels**, provide the paths for messages travelling between nodes in a network. The messages that nodes exchange are called data **frames**.

Point-to-point networks, as defined by Tanenbaum (1988), consist of many channels, each connecting two nodes. If nodes that do not share a channel wish to communicate they do so indirectly, via other nodes in the network. Thus a frame may pass through a number of nodes on the journey to its destination. Each node that receives the frame will store it, wait until the required channel is free, and then forward the frame. These networks are called **store-and-forward** or **point-to-point** networks. Common interconnection topologies for point-to-point networks are the star topology and the ring topology. Figure 3.1 (a) and (b) depict a star and ring topology respectively.

1. Link capacity. This parameter deals with the bandwidth of a link and represents the transmission capacity in megabits per second (Mbps).
2. Ports per node.
3. Load. The number of frames per second at a particular time.
4. Timeout period for frames. This is the threshold time limit for frame existence in the system.

### 5.2.2 Performance Trends

As a network becomes more loaded, i.e. as the offered load becomes greater, throughput increases. This trend will continue until too many packets are present in a section of the network (subnet) causing **congestion**. Congestion results in a degradation in throughput and delay as shown in Figure 5.1. Congestion arises when frames arrive at a node faster than the node CPU is able to process them. Alternatively, even if the CPU is infinitely fast, congestion may still occur if the incoming data rate exceeds the capacity of the output links. In both cases queues will build up and frames will be discarded. This situation tends to produce a "snowball" effect getting progressively worse as Figure 5.1(a) indicates. Frames that were discarded by receiving nodes are retransmitted, resulting in more traffic on an already congested subnet. In addition, the sending node cannot release the retransmitted frame from its buffer until it has been acknowledged. Thus congestion at the receiver's end forces the sending node to refrain from releasing a buffer it would normally have freed. In this manner congestion backs up in the network.

Congestion control should not be confused with flow control. Congestion control is a global issue concerned with ensuring the amount of traffic on a network does not exceed the carrying capacity of that network (Tanenbaum, 1988). Flow control, on the other hand, is a point-to-point issue seeking to ensure that the sending node does not transmit data faster than the receiving node can accept it.

### 5.2.1 Performance Definitions

As stated in section 2.4, the two major performance measures are throughput and delay.

**Throughput** - In this study Hammond's (1986) definition of throughput, presented in Section 2.4 is adopted. That is, throughput is defined as the number of bits of useful information transmitted per unit time, where useful information excludes any frames containing errors but includes frame header information. Unless otherwise specified, the throughput is applied on an individual link basis thus the unnormalised throughput is the sum of the throughput on the individual channels according to Hammond (1986) as stated in section 2.4.

**Delay** - In section 2.4 end-to-end delay was defined as the time from starting to transmit a packet at the sending node to the successful arrival of the entire packet at the receiver. The end-to-end delay is usually in the order of hundreds of milliseconds according to Schoemaker (1978).

In addition to the term throughput and delay, two other terms will be used in throughout this investigation. A brief description of each follows:

- **Channel capacity** - This refers to the maximum possible throughput attainable for a given set of parameters.
- **Offered Load** - Throughout this study offered load will refer to the rate of new traffic requests at each node in bits per second.

Shaikh (1989) defines throughput and delay as dependent variables. These are variables that we have no real control over. They serve as indices of performance.

Independent variables, on the other hand, are controllable. We may adjust these variables to determine their impact on the dependent variables or measures of performance. The systematic variation of these variables can be used to determine the sensitivity of the simulation model. Among these independent variables are:

# Chapter 5: The Simulation Model

This chapter discusses the objectives, design and analysis of the simulation model.

## 5.1 Simulation Objectives

In this study, the aim of creating a simulation model of the Metroring system is to assess the performance of a Metroring network as stated in Section 1.1. The performance of a computer network depends on the following factors:

1. The network topology,
2. The capacity of the links within the network,
3. The routing strategy,
4. The speed of the nodal computers contained in the network, and
5. The size of buffers and queues.

## 5.2 Network Performance

The two main measures of performance in a computer network are **throughput** and **delay** (Schoemaker, 1978), (Shanmugan, 1989). The literature offers several definitions for each of these parameters. This fact necessitates the definitions of these measures within the context of this particular investigation.

transmitting node. The receiving node ascertains the integrity of the frame by computing the frame check sequence (FCS). If an error is detected by the receiving node a Negative Acknowledgement (NACK) is sent to the transmitting node, together with the corrupted frame's sequence number. The transmitting node will then re-transmit the corrupted frame from its output queue. A frame is only removed from the output queue of the transmitting node once a positive acknowledgement (ACK) is received from the receiving node.

### 4.3.3 The Metroring Protocol Stack

Unlike LANs and MANs, there is no simultaneous demand between nodes for access to the media in a Metroring network. This is because a dedicated link exists between any two Metroring nodes. This greatly simplifies the functions of layers 2 and 3, the Data Link and Networking layers, in the OSI model. Consequently, these two layers are combined in the Metroring protocol into a single layer, called the Path Layer as shown in figure 4.4. Appendix A contains a detailed explanation of the Metroring protocol stack compared with the OSI reference model (Halsall, 1988).

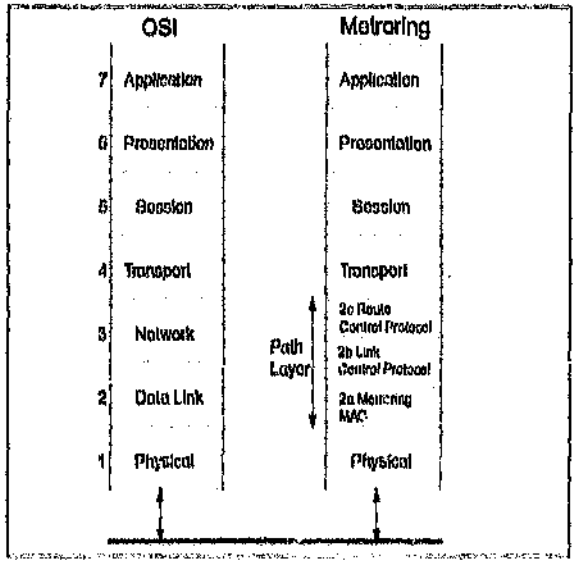



Figure 4.4: The OSI and Metroring Protocol Stacks

passed to an output queue which will be dependant on the route that the frame is to travel. For example, if the route requires that a frame be transmitted on link *x*, the frame will be placed in the output queue corresponding to link *x*. The routing mechanism determines the route that the frame is to follow, and thus into which output queue the frame will be placed. This routing operation is simple in a two port node as the data frames merely need to be forwarded on the link which did not receive the frame, i.e no routing tables are necessary.

In a segment end node (SEN), the process is more complex. Routing tables are used to ascertain the optimal path for the frame. The frame is then enqueued in the appropriate output queue. All networking functions are performed by a Metroring networking card.

### *Routing*

The Metroring protocol follows a dynamic routing scheme, i.e. the network can adapt to changes in topology. As Metroring is designed to be a highly reliable network, dynamic routing is essential. The routing scheme uses a hierarchy addressing structure to provide the shortest path between source and destination, whilst seeking to minimise routing table sizes. The hierarchy method does not require every node in the network to maintain the address of every other node. Rather, knowledge of node addresses has limited scope. Two-port nodes maintain a table of all the nodes on their segment together with the shortest "distance" (hop distance, i.e. the least number of node traversals) to these nodes. Two-port nodes also keep the address and shortest path to the SENs on their segment. Segment End Nodes have explicit listings of all nodes on the segments which they terminate. The Metroring routing scheme is described in detail in Appendix B.

### *Error Control*

In order to provide an acceptable level of service to higher protocol levels some form of error control is required. The Metroring protocol uses a form of continuous RQ (request), Go-Back-N error control (Halsall, 1992)[3]. Each Metroring frame awaiting transmission is marked with a sequence number. When a frame is transmitted, a copy is retained in the output queue of the

[3] further detail is presented in Appendix B

A Metroring node is shown in Figure 4.3. This diagram shows the following major conceptual components:

1. An output queue for each ring link.
2. An input buffer for each ring link.
3. An application port.
4. A Central Processing Unit (CPU).
5. A routing mechanism.

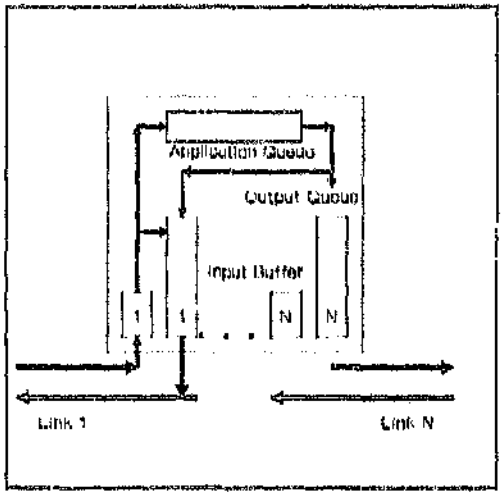



Figure 4.3 The Metroring Node

The buffers and queues are implemented in the system Random Access Memory (RAM). Routing is carried out by a software entity.

### 4.3.2 Operation of a Node

As far as possible incoming data frames are decoded while the frame is being received. Frames received from the transmission medium into an input buffer will wait there if the CPU is busy. If the destination address of a frame corresponds to the node address, the frame is passed to the application layer. If, however, the destination address is different from the node address, the frame is

The fast forward option is intended to facilitate the rapid transfer of specific frames. A frame in which the first bit is set, i.e. has a value of 1, is designated a fast forward frame. When a node receives such a frame it immediately adds the frame to the head of each of its output queues. Hence, the fast forward frame will be immediately transmitted from any empty output queue. The destination address of the frame is examined while the frame is being transmitted. The destination address of the frame determines from which output queue the frame should have been transmitted. Transmission from every output queue, except the correct one, is then halted. When the processing time taken to assess the fast forward frame's destination is shorter than the transmission time, truncated frames will propagate from all the incorrect output queues. These truncated frames will be rejected by the receiving nodes.

Note: the fast forward facility is not equivalent to an expedient data facility in which the frame is placed at the head of the queue.

## 4.3 Metroring Operation

### 4.3.1 Structure of a Node

A Metroring operates in a similar manner to an internet in that each Metroring node is capable of providing routing functions. Routing is generally defined as the action taken to find a suitable path from the source to the destination (Tanenbaum, 1988). Metroring routing involves the routing of an incoming frame in a node to one of at least two output ports, viz. a ring port or the application port. Each Metroring node operates completely autonomously. Hence nodes may transfer data on any link at any time.

Traffic may enter a node from two possible sources:

1. A user application running on the node, i.e. the application port.
2. Any one of the input ring ports.

4. The Port. This field specifies the application port which generated, or is to receive, the frame. Port numbers 0, 1, 2 and 3 are used by path layer entities. **Port numbers are unique to a specific node.**

*The Control Field*

The last 2 bytes of the header are used as control fields. Each byte has a different function.

1. Sequence Number. The sequence number is used for error control[3] . This number uniquely identifies a frame within a time period.
2. Acknowledgement Field. This field contains the number of the last correct frame received by a node. Section 4.3.2 and Appendix B for an explanation of the operation of Metroring error control.

### 4.2.2 The Data Section

This section of the frame contains the data bytes from the transport layer, and higher layers, that is being transferred between two nodes. The maximum length of this section is 4079 bytes.

### 4.2.3 The Trailer

The trailer contains a Frame Check Sequence (FCS) of 4 bytes. The FCS is used for error control. The 4 bytes contain a sequence of binary digits which are a function of the frame contents[4].

### 4.2.4 The Fast Forward Frame

[3] see Appendix B for a full discussion.

[4] A 4 byte, 32nd power polynomial CRC is used by Metroring.

### 4.2.1 The Header

The header may be divided into two sections. The first byte, and the last two bytes of the header are used for control information. The ten bytes in between these control bytes are used for the frame source and destination addresses.

*The Control Field*

The first byte of the Metroring frame forms the control field. This field is split into eight bits which are assigned the following protocol information.

*Bit 1* - The Fast Forward Flag. When bit 1 is set, i.e. it has a value of 1, the frame is designated as a Fast Forward Frame. A detailed discussion of the Fast Forward Frame is given in Section 4.2.4.

*Bit 2* - The Frame Type Flag. A value of 1 indicates a Control Frame while a value of 0 indicates an Information Frame.

*Bits 3 to 5* - Priority Indicator. These bits indicate the priority of a frame. Three priorities are currently defined, priority 1 being the highest priority frame.

*Bits 6 to 8* - Reserved Bits. These bits are currently unused.

*The Address Field*

The next 10 bytes of the header comprise the address field. This field is split into a source and a destination address, each one being 5 bytes long. Each address has four distinct sections:

1. The Region. *This single byte indicates the region on which the node resides. Region values* of 0 and 255 are reserved.
2. The Segment. Two bytes are reserved for the segment address, allowing a possible 65534 segments (values 0 and 65535 are reserved). **The segment address is unique within a specific region.**
3. The Node. A single byte offers a possible 254 nodes on any segment (0 and 255 are reserved). **The node address is unique on a specific segment.**

Every device attached to a Metroring is known as a node of which there are three distinct types:

1. Two port nodes. These are nodes with only two links connected to two neighbouring nodes. Generally, this is the most common type of node in a Metroring network.
2. Segment End Nodes (SENs). SENs have three or more links and connect Metroring segments.
3. Bridge nodes. Bridge nodes connect Metroring regions together.

## 4.2 The Metroring Frame

Data is transmitted between nodes in the form of frames. A Metroring frame is shown in Figure 4.2. The Metroring frame can be divided into three distinct sections as shown in Figure 4.2, namely the header, the data and the trailer sections. The frame header and trailer contain the pro-

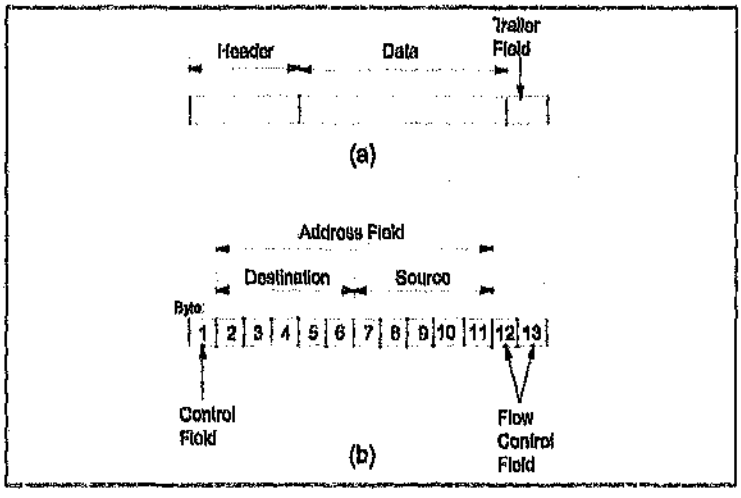



Figure 4.2 The Metroring Frame

tocol management information. The data portion of the frame contains any other information involved in the communication between nodes. The length of Metroring frames are consistent with those of FDDI, i.e. a maximum length of 4 KBytes.

In addition, the Metroring protocol offers a fast forward facility providing rapid transfer of selected frames across lightly loaded links. The Metroring topology incorporates high link redundancy resulting in a highly reliable network. These features make Metroring a reliable and practical solution for the design of internets or MANs required to carry heavy loads.

## 4.1 Metroring Topology

A Metroring network is depicted in Figure 4.1. Nodes on the network are connected in such a way that they form one or more closed rings. Each node has a minimum of two links. Each link consists of two unidirectional, point-to-point, fibre optic channels. Logically, one channel forms

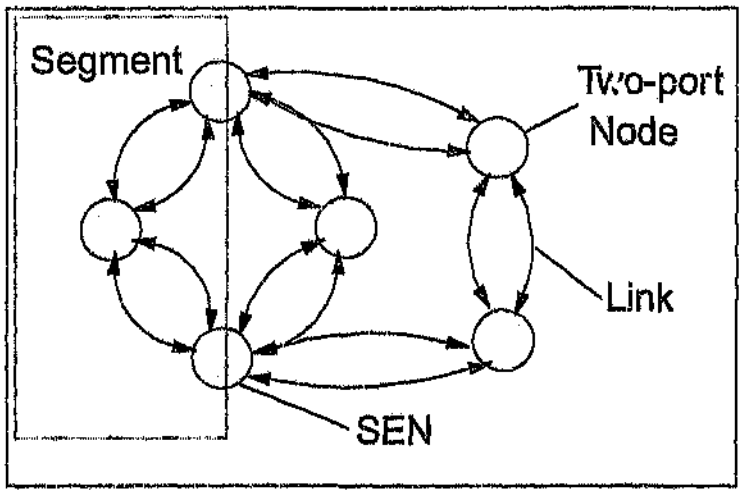



Figure 4.1 A Metroring

the transmit path and the other the receive path. Metroring nodes may be added to a ring, thereby forming a segment. The nodes which connect segments together have a minimum of three links and are called **Segment End Nodes** (SENs). A **region** consists of a number of segments. Metroring regions are connected by **bridges**.

# Chapter 4: The Metroring Networking Protocol

Although this dissertation is concerned with the performance evaluation and simulation of the Metroring protocol, a brief explanation of the Metroring protocol is required in order to properly appreciate the chief goals of this investigation.

Metroring is a highly reliable protocol that is able to handle heavy traffic loads using existing technology. Metroring achieves this through the use of simultaneous transmissions, spatial reuse and fibre optic technology. Metroring has been developed to link computing devices in complex configurations fulfilling an internet role. The physical extent of the Metroring configuration corresponds to that of a MAN, i.e. a diameter of about 100km (Smythe, 1991). The Metroring possesses a ring topology resembling that of a FDDI or Token Ring network, i.e. end-to-end fibre links as the connections in the ring. Although similar in topology, Metroring differs fundamentally from Token Ring and FDDI in that:

1. End-to-end transfer is limited to adjacent nodes in a ring. This enables concurrent interlink transfers which Token Ring prohibits and to which FDDI approximates by allowing multiframe packets on the ring.
2. Metroring allows parallel segments which enable load distribution and reliability to be enhanced. Token Ring and FDDI do not allow this.

Metroring features a number of facilities which make it suitable for handling network loading. Among these unique features are:

1. The ability to handle simultaneous data transfers,
2. Priority forwarding facilities, and
3. Parallel processing of input/output operations.

# Chapter 6: Model Implementation

This chapter describes the process of implementing the simulation model derived in the previous chapter in software. As the model is implemented in the object-oriented paradigm, section 6.1 gives a brief introduction to object-oriented concepts. The software design procedure is described in section 6.2. Section 6.3 presents the software implementation issues in detail and section 6.4 discusses the verification and validation of the simulator.

## 6.1 The Object-Oriented Paradigm

When approaching the problem of developing a simulation of an Internet type computer network, it is important to choose a programming style appropriate to the problem. The style is really "a way of organising programs on the basis of some conceptual model of programming and an appropriate language to make programs written in the style clear" (Booch, 1991). According to Bobrow and Stefik (Booch, 1991), the major programming styles are:

- Procedure-oriented
- Object-oriented
- Logic-oriented
- Rule-oriented
- Constraint-oriented.

A particular problem may favour any one of these styles. There is no set rule as to which style is the best for a given application. However, the object-oriented style is particularly suited to the

Model verification and validation will be addressed in greater detail in Chapter 7, where it is applied to the Metroring simulation model. Before discussing these issues, however, it is necessary to construct a software model based upon the simulation model synthesised thus far.

It is important to note the following points concerning the Metroring simulation model:

1. As the simulation model is only concerned with networking operations, all details concerning protocol layers higher than the OSI networking layer are ignored.
2. It is assumed that the receive data rate is faster than the transmit data rate for the networking nodes. Hence the flow control mechanism is not included in the model.
3. Congestion control takes the form of packet discarding (Tanenbaum, 1988).
4. Observation of performance under normal conditions, and across an extremely reliable medium, makes the inclusion of complex error control techniques unnecessary. Other parameters, such as the number of frames discarded, may be used to gauge the effect of discarding packets.

## 5.4 Model Validation and Verification

Once a model of the real-world system has been derived, the model needs to be implemented. This step translates into a simulation model implemented through software. By examining the results which the simulation model yields, the model may be verified and validated. Model **verification** relates specifically to the computer program and poses the question, "is the model implemented correctly in computer code?".

Model **validation** involves deciding whether a model is an accurate representation of the real world system. It is therefore necessary to grasp the operation of the actual system and to have a good idea of the system's expected behaviour. A model may then be validated by observing the behaviour of the simulation model in relation to the behaviour expected from the real system. Liaison wi' people who are well aquainted with the real world system ensures a close mapping between the simulation model and the actual system, and is an important aid in validating the simulation model. As the model is validated, changes might be needed in order to increase confidence in simulation results. The model is therefore refined to ensure that the operation under simulation produces results in the correct form and in sufficient detail.

*Links*

When a frame is generated by an application it immediately joins the queue of the network node. Conceptually, the frames move through the queue awaiting their turn to be processed. Once processed, the frames are transmitted on the correct link. When a frame reaches its destination it is removed from the system. Further assumptions about links in the Metroring simulation model are included in Section 5.3.3.

### 5.3.3 Assumptions

The following assumptions regarding the traffic model have been made:

1. Interarrival times of packets are assumed to follow an exponential distribution. This results in a Poisson arrival process of frames which is widely used in the simulation of computer networks.
2. Packet lengths are assumed to vary randomly following an exponential distribution. The user may also choose to simulate constant packet sizes.
3. Every node on the network is able to generate and receive a packet.
4. Every node, with the exception of the node creating the packet, is equally likely to be chosen as the destination for each new packet.

Since performance under normal conditions is being investigated, the following assumptions automatically follow:

1. Frames are not lost, duplicated or reordered during their propagation over links.
2. Link faults and the malfunction of other components of the system are omitted from the model as such aspects are only of interest when reliability, availability or security are investigated (Schoemaker, 1978).
3. Any frame transmitted on the link arrives at the other end in a finite amount of time.

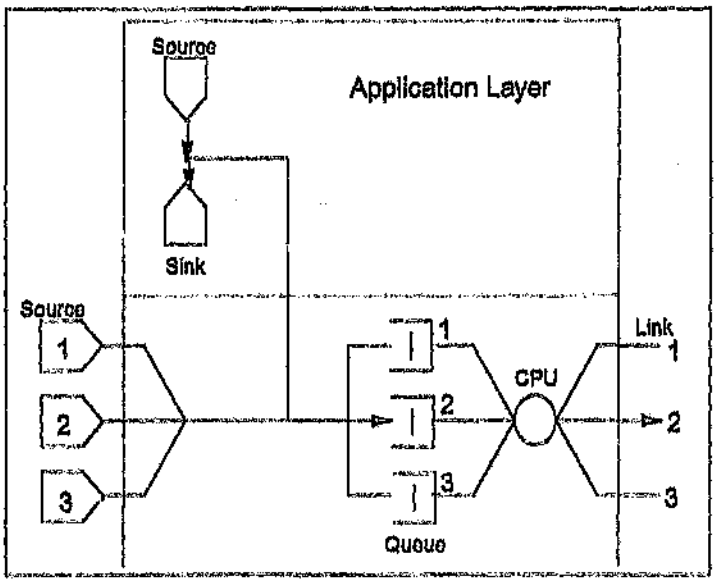



Figure 5.3: Queuing Model of a Metroring Node

Following from these points it is apparent that a "node" in the Metroring model is, in reality, the Metroring networking controller, and not the complete computing device. As this investigation is concerned with the Metroring behaviour, i.e. the networking functions, 's fact is of no consequence to the simulation model. The simulation model assumptions made concerning Metroring nodes are listed in Section 5.3.3.

***Routing***

When a Metroring node receives a frame it will allocate the frame to an output port based upon the routing information contained within that node's routing table. The Metroring routing mechanism is described in section 4.3.2 and Appendix B. The presence of a routing mechanism ensures that there are no duplicate frames circulating in the network, and that frames do not loop forever but traverse a finite number of hops from source to destination. To avoid unnecessary complexity in the Metroring model the true Metroring routing protocol is not implemented. Instead, static routing tables, denoting shortest path routes based on hop count, are used. These tables are entered by the user.

caused by a node as in traditional models, the Metroring model uses a number of queues in each SEN corresponding to each individual path.

*Queues*

It is assumed that all queues are served by a First-In-First-Out (FIFO) queuing discipline. The most important characteristics of a queue are the length of the queue (mean, maximum, variance) and the queuing time (mean, variance) (Schoemaker, 1978). The queuing time, or the time that a frame spends in a queue (representative of the amount of time that a frame is delayed by a node), is indirectly determined by a statistical distribution. Queuing time will depend upon the rate at which the server is able to serve the waiting transactions, i.e. the rate at which a node's processor is able to process frames waiting in the queue. This service rate is known as the **service time** and is dependent on the processing power of the node and the size of frames.

*Nodes*

The operation of a node is captured in the queuing model. The delay that a frame encounters when passing through a node is comprised of the software time and the Direct Memory Access (DMA) time (Sventek, 1989). The software time accounts for software overheads associated with the protocol. DMA times are the times taken to copy data to and from memory. The Metroring model incorporates these delays in a single parameter specified by the user. It is important to note the following points about a Metroring node:

1. All networking operations will be handled by a Metroring networking controller card.
2. The Metroring networking controller card will have its own, dedicated microprocessor and RAM (Random Access Memory). This offloads the real-time processing burden of the network from the node CPU (Taylor, 1989). This feature is extremely relevant with intelligence gravitating out to the nodes in current networks (Taylor, 1989), because placing the added burden of communications processing on the nodes detracts from their effectiveness in the tasks they were designed to perform.
3. The dedicated networking memory will be a dual port memory supporting full duplex direct memory access, permitting concurrent transmit and receive operations.

the real system and the simulation model (Schoemaker, 1978). The simplifications and assumptions concerning the Metroring model are detailed throughout the remainder of this section as well as in Section 6.3.

The impact which the model simplifications have on the correspondence between the system in the real world and the simulation model may be examined at a later stage by observing the sensitivity of the model's outputs to small changes in the model's input variables (Schoemaker, 1978).

Since there are certain variables in the Metroring model which are random, for instance the inter-arrival times of frames, the model is stochastic (Law, 1982). Hence the output data will be random and will only yield an approximation of the operation of the real system.

A network consists of a number of nodes interconnected via links. These nodes communicate with one another by means of frames. Le Goff and Le Lann (Schoemaker, 1978) identify three basic resources in a computer network: storage capacity, transmission capacity and processing capacity. These resources are implemented as buffers, links and processors (Central Processing Units and Input/Output processors) respectively. From the discussion above, it is evident that there is a close correspondence between the description of a queuing model and that of a Metroring network. Frames form the customers in the Metroring simulation model, while the processing facility of a node is analogous to a server or resource. In the Metroring simulation the number of frames far exceed the number of processors. Frames will therefore wait in queues for service at a node.

Any node may generate a new frame at any instant in time. A node is therefore a source, i.e. it has the ability to introduce new frames to the simulation. Similarly any node may remove a frame from the system. This happens when a node receives a frame destined for itself. In this case a node may act as a sink - it is removing frames (or transactions) from the system. The Metroring queuing model is shown in Figure 5.3.

This queuing model differs from existing LAN models because SENs have more than a single path for frames to follow. This reflects the internet nature of the Metroring protocol compared with a LAN protocol, viz. a routing function. Thus instead of only a single queue representing the delay

interactive data traffic in computer networks. Moreover, very often the only information available about these random variables is their mean values. When one has little information about random values other than mean values, then it is reasonable to arbitrarily assume that the random values have a distribution which is completely specified by the mean, e.g., the exponential distribution (Sauer, 1983). Three distributions relevant to this investigation, namely the exponential, Poisson and uniform distributions, are discussed in Appendix C.

The literature available on simulation and modelling of computer networks affirms the use of the exponential distribution to characterise packet interarrival times and packet lengths. All the simulations and models surveyed in this investigation (Frost, 1990), (Halsall, 1991), (Cobb, 1992), (Shanmugan, 1992), and (Chiarawongse, 1988), as well as the surveys carried out by Boggs (1988) and Sventek (1989), indicate the use of the exponential distribution to characterise the network traffic.

Following the above arguments, the Metroring model assumes a Poisson arrival process of packets at the network nodes, which results in an exponential interarrival time distribution (see Appendix C for a detailed explanation). The length of Metroring frames in the simulation may be chosen to be either constant, or of a random size varying about some specified mean size according to the exponential distribution.

The Metroring frame sizes in the simulation are bounded by limits which are specified in the protocol, viz. an upper limit of 4096 bytes, and a lower limit of 21 bytes (Tjirkalli, 1993).

## 5.3.2 Metroring Simulation Model

When simulating a complex computer network such as a Metroring it is impractical to include every detail of the network into the simulation model. It is therefore necessary to make certain simplifications concerning the real world system. Law and Kelton (1982) point out that a good modelling principle is to begin with a simple model which can later be made more sophisticated if required. One of the main problems in simulating a system is to find a suitable trade-off between

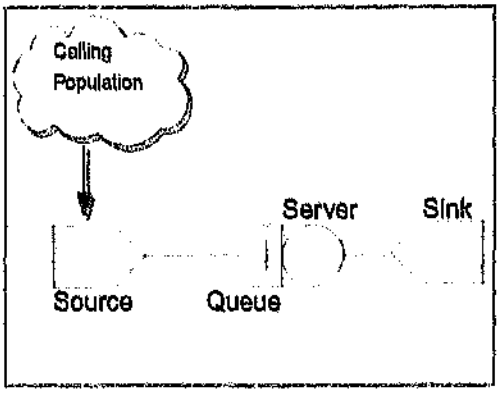



Figure 5.2: Simple Queuing Model

part in the load imposed on a network. In real computer networks devices are often inactive for long periods of time and then require high data rates for a short time period (Boggs, 1988). Hammond (1986) points out the difference in the extremes of required data rates. Consequently defining a "typical" steady-state distribution for network traffic is a difficult task.

A second parameter that affects the traffic on a network is the individual packet lengths. The packet lengths indirectly influence the service times at nodes. Even though the time taken to process a single bit over a channel or through a device is constant, the time taken to process a packet will depend upon the number of bits in that packet. In computer networks the packet length is randomly variable, and thus the processing time for a packet also becomes a random variable.

Information on the distribution of packet lengths is also highly variable (Hammond, 1986). For example, investigations cited by Boggs and Hammond show that the packet length distribution on Ethernet tends to follow a bimodal (i.e. most packets either near the minimum or the maximum packet length) pattern.

Despite the variability in network traffic, some description must be used as a starting point for performance analysis. Hence it is necessary to determine what distributions for interpacket arrivals and packet lengths can give the most tractable analytic results. Hammond (1986) concludes that the Poisson arrival process, which gives an exponential distribution of interarrival times, and the exponential or geometric packet length distributions seem to provide the best characterisation of

*A/B/c/N/K*

*A*: *The interarrival time distribution.* These distributions are discussed in Appendix C. Some of the more common interarrival distributions are *M* (exponential), *D* (deterministic), $E_k$ (Erlang type k), and *G* (general).

*B*: *The service-time distribution.* The same letters are used as in the interarrival case to denote similar distributions.

*c*: *The number of parallel servers.* In some systems several servers may simultaneously serve customers.

*N*: *The system capacity.* System capacity refers to the limit on the number of customers that may be in the waiting line or system. Most real systems have a finite system capacity. If this parameter is omitted from the notation, the system capacity is assumed to be infinite.

*K*: *The size of the population.* If this parameter is omitted from the notation, the population is assumed to be infinite.

*Queue discipline.* These are indicated by FIFO (first-in, first-out), LIFO (last-in, first-out), SIRO (service in random order), PRI (priority), and GD (general discipline). If omitted a FIFO discipline is assumed.

For example, *M/M/1* represents a single server system that has unlimited queue capacity and an infinite population of potential arrivals. The interarrival times and service times are exponentially distributed. A FIFO queue discipline is assumed.

### *Network Traffic*

Data flow within a computer network is typically non-uniform or stochastic in nature. Hammond (1986) states that at any point in such a network, the arrival times of frames at nodes are random variables. Although these times appear to vary randomly, often they may be described by some statistical model.

Unfortunately network traffic is a highly variable quantity affected by many variables. Factors such as the nature of the traffic and nodes, the protocols used, and the time of the day all play a

## 5.3 Simulation Model Design

### 5.3.1 Queuing Models

Hammond (1986) points out that performance analysis of computer networks is concerned with the nature and characteristics of data flow. According to Hammond (1986), throughput, delay, and other performance parameters are measures of how the network processes the messages it must transmit to fulfil its function. Queuing theory is a frequently used tool in the prediction of these measures of performance. The notation and operation of queuing models is now described.

A queuing system describes the arrival of customers at a server, the time that the customers spend waiting for service and their subsequent departure from the server. The term **customer** refers to any type of entity that can be viewed as requesting **service** from a system (Banks, 1984). A **server** can be defined as any resource offering service in a system. In virtually all real systems there are a limited number of resources, or servers. Whenever there are not enough servers to service all the customers simultaneously, queues result (Graybeal, 1980). For example, in a bank there are usually more clients requesting service in the bank than there are tellers to service their requests. The result is that queues are formed as clients wait for service from a teller.

The **calling population** is the population of potential customers. In systems with a large population of potential customers, the calling population is usually assumed to be infinite (Banks, 1984). A **source** introduces new customers into the queuing system. A **sink** removes customers from the system. The time between two successive arrivals at a server is referred to as the **interarrival** time. If arrivals of customers occur at random points in time, the interarrival times are usually characterised by a probability distribution. Figure 5.2 shows a simple queuing model.

*Notation*

The notational system proposed by Kendall (1953) to describe a queuing system has been widely accepted and is presented below. With Kendall's convention a queuing system is described by a series of symbols separated by slashes:

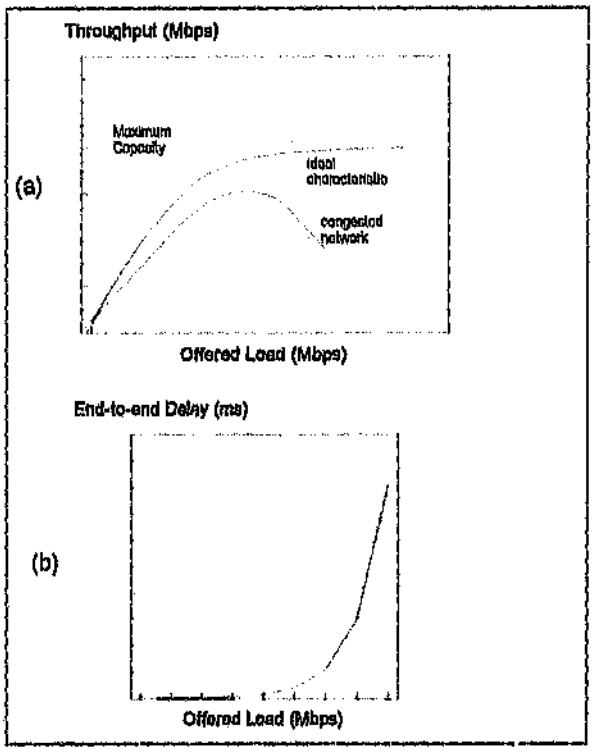



Figure 5.1 (a) Throughput, (b) End-to-end Delay Characteristics

Increasing load on a network also leads to an increase in the end-to-end delay, i.e. the time taken for a frame to reach its destination, of frames. Generally end-to-end delay increases exponentially with increasing load. This trend is shown in Figure 5.1(b).

In order to improve network performance it is desirable to maximise throughput and to minimise end-to-end delay. Unfortunately these are very often conflicting goals. Networks boasting a high throughput frequently have high loading and a high occupancy of buffers in network nodes. These characteristics tend to increase the end-to-end delay as frames are forced to wait in buffers for long periods of time.

validation processes are important in decreasing scepticism towards a system model. Banks (1984) gives two reasons why these processes are significant:

1. In order to produce a model that represents the true system behaviour closely enough for the model to be used as a substitute for the actual system for the purpose of experimenting with the system.
2. To increase to an acceptable level the credibility of the model so that the model will be used by managers and other decision-makers.

It is clear that model verification and validation form an integral part of model development. These two processes will be described in greater depth, and they will be applied to the Metroring model.

### 6.4.1 Verification

The process of verification examines how accurately the conceptual model represents the operational model. There are a number of suggestions in the literature for verifying a model. Many of these were applied to the Metroring simulation model.

- Flow diagrams indicating the possible action a system can take when an event occurs were used to follow the model logic for each event type. Figure 6.3 and Figure 6.4 shows the flow diagrams for the receive and transmit events in the `BaseNode` class.
- The model output was closely examined for reasonableness under a variety of settings of input parameters. One method of achieving this was to have the simulator program produce a wide variety of output statistics in addition to the ones required by the simulation study. The C++ language also has extensive facilities for error trapping and integrity testing of variables. Variables were observed in relation to expected trends to discern how the program was behaving. The performance measures obtained from simulation experiments were also good measures of reasonableness.
- The use of a trace - a detailed printout giving every variable in the program - is another useful method of verifying that the program is behaving as expected. C++ has a

The event list queues all events according to the times at which they are scheduled to take place. When an event is removed from the event list, the specific operations associated with the event are handled by the appropriate class in the network hierarchy.

The random number generator uses the additive congruential method described in (Sedgewick, 1992) to generate random numbers. The standard deviation algorithms are derived from Press et al (1988). Random numbers taken from Banks (1984), Appendix A, are used to initialise the random number streams.

Future evolution of the Metroring protocol meant that the simulation software be designed to remain as flexible as possible.

### 6.3.2 Limitations Imposed on the Metroring Simulator by Implementation

- Due to limited memory a practical limit is imposed on the size of the Metroring being simulated. This maximum size is dependant on nodal buffer size, the number of nodes in the network and the number of frames generated during a simulation run.
- The Borland `PriorityQueue` class was only able to handle a limited number of objects having the same priorities. Thus some simulations using the fixed frame size may not produce accurate results when too many events are scheduled at the same time.

## 6.4 Model Verification and Validation

The terms verification and validation were introduced in section 5.3. Essentially model verification refers to the comparison of the conceptual model to the computer code and asks the question, "is the model implemented correctly in computer code?", while model validation refers to the act of determining that the model is an accurate representation of the real system. The verification and

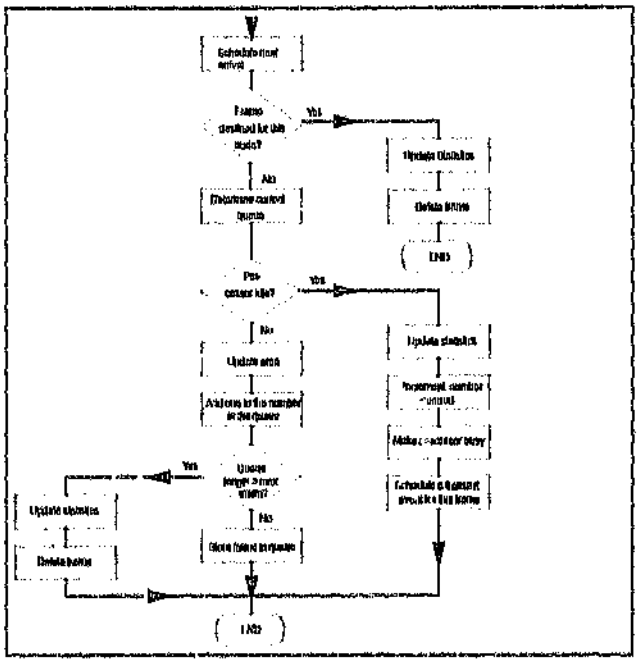


Figure 6.3: Flow Diagram for Frame Arrival Process

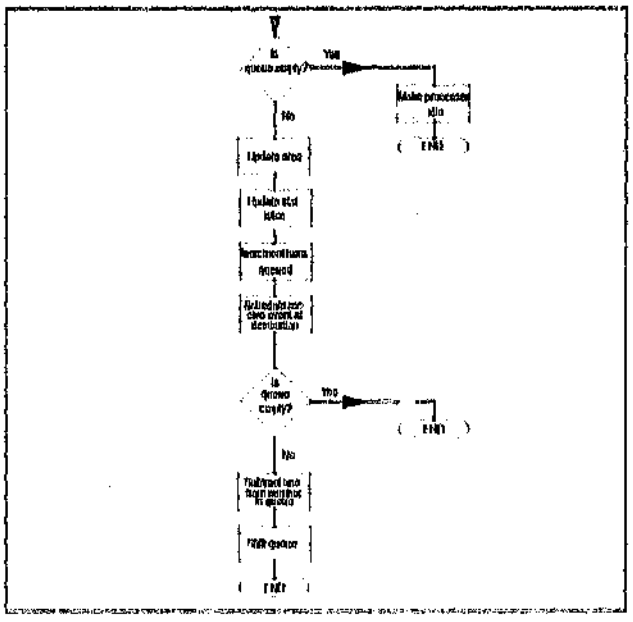


Figure 6.4: Flow Diagram for Frame Departure Process

It is convenient to split the discussion of the simulator implementation into the two categories of section 6.2.4, viz. the topology specific, and the simulation specific aspects of the model.

*Topology Specifics*

The class hierarchy of section 6.2.4, depicted in Figure 6.1, intuitively follows the description of a Metroring network. A network consists of a number of regions. Each region consists of a number of nodes, and each node contains, among other things, some queuing mechanism. Each class in this progression uses an array to contain the objects below it. For example, a network object uses an array to contain the region objects comprising the network. The queue is presently the lowest level of abstraction implemented in software, however, the hierarchy may be extended to any detail. Should more detail be required, for example the representation of an input buffer, this class would merely be added to the class hierarchy.

When wishing to change the attributes of a node, or when it is necessary to examine the statistics relating to a specific node, one need merely look at the relevant node object. In a simulation model implemented using conventional process-oriented design and programming, it would have been extremely difficult to offer the flexibility associated with an object-oriented counterpart. This is mainly because statistics and attributes would be tied up in complex data structures.

The processes to handle the arrival and departure of a frame, together with the attributes of the node (e.g., mean service time which is dependent on the node's processing power and queue sizes etc.) and the simulation statistics connected with the node are all be effectively encapsulated within the node object. The arrival and departure processes are shown in the flow charts 6.3 and 6.4 respectively.

*Simulation Specifics*

The simulation tools provide a framework for many different discrete event simulations. The event list and random number generators could be used in any discrete event simulation software. Indeed this reuse of classes is one of the major benefits of object-oriented programming.

This is the step where the interface arguments are considered and specified. When considering class interfaces it is important that they do not encompass more than a single level of abstraction. Typically, all operations on a class should support the same level of abstraction (Stroustrup, 1991).

It should be noted that the focus of the design is on the outside view of the classes. In general, implementation issues are not considered in the design phase. Implementation details which do need to be included are best handled in step 2.

As the iterative procedure outlined above progresses, the class hierarchy will need to be reorganised. Classes and their interfaces will change.

### 6.2.6 Implementation

This phase often forms part of the iterative design process. It is during this phase that the implementation details of classes and objects are considered. The focus shifts from an outside view of each class to an inside one. Design decisions concerning the representation of classes and objects could well result in a leap back to a previous step in the design process.

The implementation of the design in software is the subject of section 6.3.

## 6.3 Implementation

The Metroring simulation model was implemented using Borland C++, Version 3.1. The reasons for choosing this language appear in Appendix E.

### 6.3.1 Software Implementation of The Metroring Simulation Model

Region and BaseNode objects respectively. The Frame class is used in the BaseNode and NodeQueue class interfaces.

The classes specific to the simulator engine are shown in Figure 6.2. The central class in this hierarchy is the Simulator class. For every Simulator object, there is one Generator and one EventList object. The EventList class inherits its functionality from the Borland PriorityQueue class. The EventList class uses the BaseEvent class, derived from the Borland Sortable class, in its internal queuing structure. The Generator class uses the class

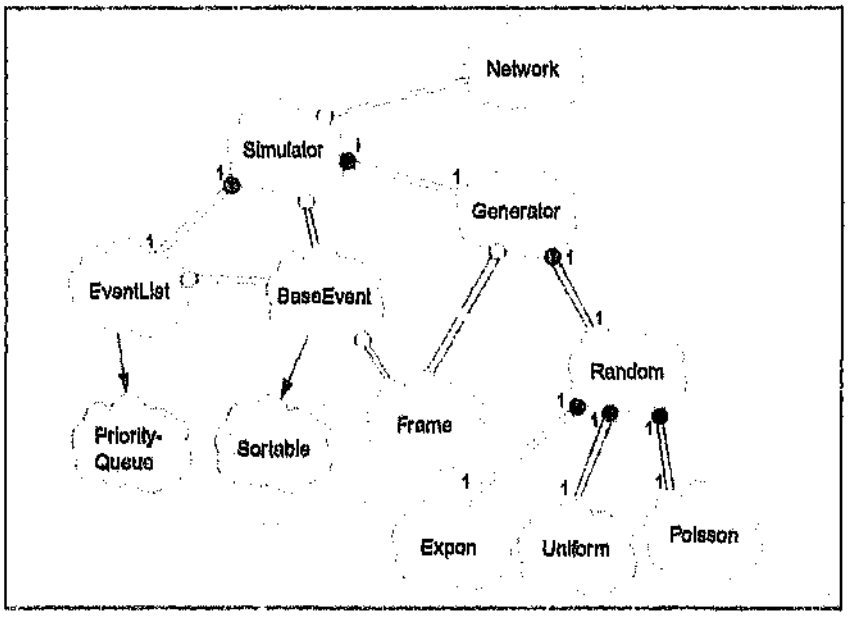



Figure 6.2: Mirroring Simulator Class Diagram

Random in the generation of network traffic. Every Generator object will have a single Random object associated with it. The class Random uses the classes Expon, Uniform and Poisson in its implementation. A function of the Generator class is to generates frames and thus the Frame class is used in the Generator class' implementation.

## 6.2.5 Step 4: Specify Interfaces

### 6.2.4 Step 3: Specify Dependencies

This stage of the design process aims to specify the dependencies of classes on other classes. The

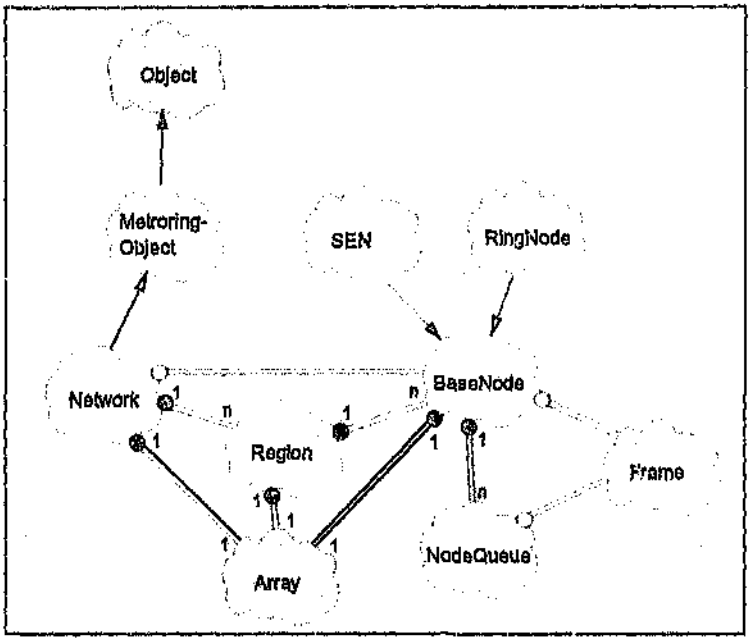



Figure 6.1: Metroring Simulator Class Diagram

main dependencies are inheritance and using relationships. The dependencies among the main classes in the Metroring simulator are shown in Figure 6.1 and Figure 6.2. The shaded class icons represent classes from the Borland Class Library.

Figure 6.1 shows the class hierarchy that emerged from considering the physical Metroring domain, or the topology specific aspects of the model. The `Network`, `Region` and `BaseNode` classes use the `Array` class in their respective implementations. The cardinality expresses that each of these objects will only use one array object. The `SEN` and `RingNode` classes inherit from the more general class `BaseNode`. The class `BaseNode` uses the `NodeQueue` class in its implementation, and for every instance of `BaseNode` there will be $n$ instances of the class `NodeQueue`. Similarly for every `Network` and `Region` object there will be $n$ instances of the

The most obvious classes appear in the vocabulary of the problem domain. Very often nouns (i.e. those things which are tangible) will correspond to the classes and objects needed in the program. This is true in the case of the Metroring Simulator. Nodes, SENs, frames and queues are a few of the classes present in the program which were extracted directly from the vocabulary of the problem domain. A list of classes included in the Metroring simulator software appears in Appendix F.

Typically this step ends when no new basic concepts or mechanisms remain, or when classes or objects that we have discovered may be constructed from existing software components.

### 6.2.3 Step 2: Specify Operations On Classes

The aim of this step is to establish how classes or objects interact within the system. Stroustrup (1991) stresses the need to minimise the set of operations which this step produces, "the more functions, the more likely they are to remain unused and the more likely they are to constrain the implementation and the further evolution of the system". He suggests the following strategy in deciding on what functions are needed for a class:

1. Consider how an object is to be constructed, copied and destroyed.
2. Define the **minimal** set of operations required by the concept that the class is representing.
3. Consider which operations could be added for notational convenience and include only the most important.
4. Consider which operations should be virtual, i.e. which operations will be implemented by a derived class.
5. Consider any similarity in naming or functionality of classes.

Appendix F contains a brief description of some of the functions or methods that the various classes use.

under the constraint that it will be changed in many ways". This is an extremely important design aim in the case of the Metroring simulator as the system specifications, as well as the simulation requirements are likely to change during the course of the design. Bearing the need to design for change in mind, Stroustrup identifies three aims: flexibility, extensibility, and portability.

The way in which these aims are achieved in the object-oriented paradigm is by encapsulating the areas of the system which are likely to change. In this way, when changes do occur, only the relevant class needs to be changed.

The simulation program is really a model of reality, with each class representing a concept. Stroustrup (1991) identifies four steps in the design phase:

1. Find the concepts (or classes) and their most fundamental relationships.
2. Refine the classes by specifying the sets of operations on them.
3. Refine the classes by specifying their dependencies on other classes.
4. Specify the interfaces for the classes.

It is important to note that these steps are iterative. One would typically loop through these four stages several times before arriving at a final design.

The Booch Notation will be adopted in the presentation of the Metroring simulator program design. A brief discussion of this notation is presented in Appendix D. For a full explanation the reader is referred to (Booch, 1991). The major steps in the software design follow.

### 6.2.2 Step 1: Identification Of Classes

Using the object oriented design approach, the first step is to identify the key concepts or classes and their basic relationships. The first few iterations of this step will produce a series of candidate classes. The final classes will be determined only once a design has been finalised.

The object-oriented design and programming approach best fits the problem of simulating a Metroring for a number of reasons.

1. The process of identifying the entities within the system which is being simulated corresponds closely to the initial step of identifying classes in object-oriented design. Often the entities identified for the purposes of discrete event simulation are identical to those identified via the object-oriented design process.
2. The variables which denote the state of a system may be encapsulated in classes. This decreases the system complexity, and because access to this data may be restricted, data integrity is maintained.
3. Components which are common to all discrete event simulations may be encapsulated in re-usable modules. Software re-use is a major benefit of object-oriented technology. In the Metroring simulator's case components such as the event list, clock and random number generator may be encapsulated in base classes which may be ported to other discrete event simulation models.
4. The object-oriented simulator is more resilient to change. This is an extremely important factor considering the Metroring protocol is still being developed. This factor also becomes significant when the simulation is required to examine operation of the system at a greater level of complexity. The transition to a lower level of abstraction in the object model is easier than it would be in any other model.
5. As the object model bears such a close resemblance to the real system, it appeals to human intuition. This is especially helpful when it comes to the extension and maintenance of the simulator.

## 6.2 Design of the Metroring Simulation Software

Once the simulation model of the Metroring computer network has been formulated, and once the choice of programming language and appropriate design approach has been established, the task of designing the simulation software begins. Stroustrup (1991) identifies simplicity as a central aim of the design process; "we must aim for a design and an implemented system that is simple

extension. This is really a packaging convenience, providing the user with easy access to the information needed to incorporate the module in an application.

*Hierarchy*

Hierarchy is a ranking or ordering of abstractions (Booch, 1991).

In addition to these elements, it is necessary to define the concepts of *objects* and *classes*, and the relationships existing between them.

*Objects and Classes*

The concepts of an object and a class are tightly interwoven. However there is an important difference between the two. Whereas an object is a concrete entity that exists in time and space, a class represents only an abstraction, the "essence" of an object, as it were (Booch, 1991). In the context of object-oriented design, an object is very often an instance of a class. A class is a set of objects that share a common structure and a common behaviour (Booch, 1991).

There are a number of relationships that may exist between classes The following two relationships are important in the design of the Metroring simulator:

- Inheritance relationships
- Using relationships

Inheritance denotes a "kind of" relationship. For example, a SEN is a kind of node, i.e. a SEN is a specialised subclass of the more general class, node. Using denotes a "part of" relationship. Therefore a queue is not a kind of node, it is a part of a node.

### 6.1.2 The Suitability of the Object-Oriented Paradigm to a Metroring Simulation

problem of simulating the internet style Metroring networking protocol for a number of reasons as set out in section 6.1.2.

### 6.1.1 Elements of an Object Model

The key elements of an object model are abstraction, encapsulation, modularity and hierarchy. Booch classifies these as major elements of the object model, and further states that any model without any of these elements is not object-oriented (Booch, 1991). Because these elements form such a fundamental part of the design of the Metroring simulator it is necessary to clarify these ideas briefly in the following paragraphs. These fundamental concepts are discussed in greater detail in Appendix D.

***Abstraction***

Abstraction is one of the natural ways in which the human mind handles complexity. Essentially abstraction is "a simplified description, or specification, of a system that emphasises some of the system's details or properties while suppressing others" (Booch, 1991). Ideally the user would like to see the characteristics of an object which he finds important, while details which are irrelevant to him are effectively hidden.

***Encapsulation***

Encapsulation and Abstraction are complementary concepts. Encapsulation hides the implementation detail of a class from a user, whereas the essential behaviour of the class is visible to the user (abstraction).

***Modularity***

As programs grow in size, modularity is a useful means of reducing the ensuing complexity. In C++, modules are merely separately compiled files. Traditionally the module interface is placed in a file with a `.h` extension, while the implementation is placed in another file with a `.cpp`

### 7.3.4 Efficiency

Rodrigues (1990) defines efficiency as the bandwidth that can actually transport end-user data so that the system still remains stable. Unlike the FDDI architecture, the efficiency of the Metroring protocol does not degrade as the physical network size increases as the system bandwidth is independent of the number of nodes. This fact suggests that Metroring is a more suitable technology than FDDI for MAN environments requiring a high efficiency. Metroring features a high efficiency as the protocol overheads are relatively small. A simple comparison of the efficiency of FDDI, DQDB and Metroring as a function of the number of nodes is shown in Figure 7.8[8].

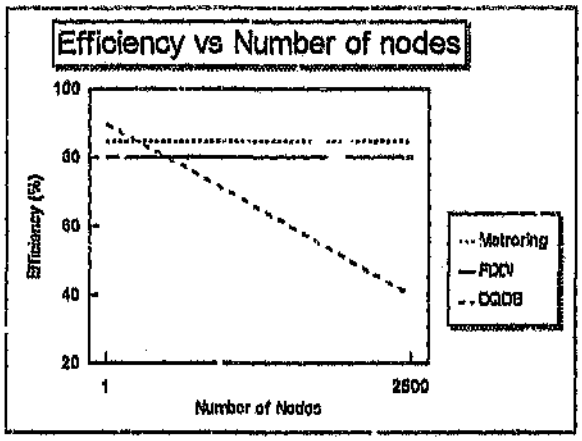



Figure 7.8: Efficiency vs. Number of Nodes

## 7.4 The Effect of Frame Size on Performance

Performance graphs for frame sizes of 100, 1k, 2k and 4k bytes are shown in Figure 7.9.

Figure 7.9(b) shows that the end-to-end delay is greater for larger frames. This trend is attributable to the fact that the transmission delay, which is a component of end-to-end delay, is

[8] FDDI and DQDB characteristics are taken from (Rodrigues, 1990).

re-transmissions of frames that have been discarded. In reality, however, discarded frames will result in an increased delay as shown by the dashed portion of the curve.

A measure of performance that is closely related to end-to-end delay is queue occupancy.

### 7.3.3 Queue Occupancy

Figure 7.7 shows the rapid growth in the queue size of a node as the incoming traffic rate surpasses the frame service rate. At high loads the queue size will continue to grow until the queue is filled to its capacity and any further incoming frames will be discarded. Figure 7.7 shows that this takes place at approximately 850fps. Consequently frames will begin to be discarded in greater numbers at this point. The plot of the number of frames discarded against the traffic arrival rate per node is also shown in Figure 7.7 and confirms this assertion.

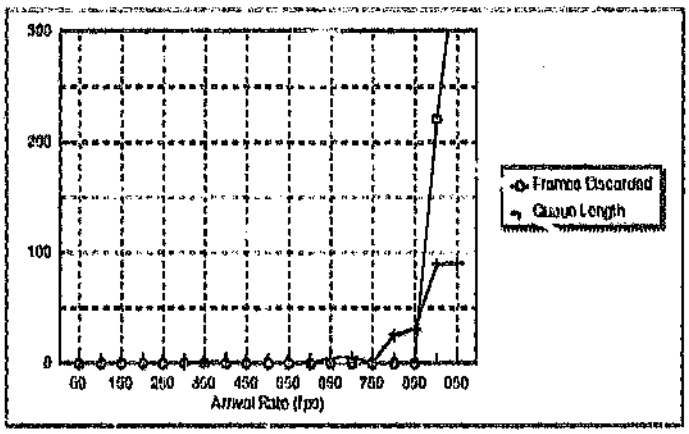



Figure 7.7: Queue Occupancy for a 5 node Metroring

The effects on performance of variations in packet size and node forwarding rates are examined using the simple topology shown in Figure 7.2(b).

node in the network is processing the maximum amount of data that it's physical processing power will allow. At a forwarding rate of 10kfps, the maximum possible throughput for any one node is 5.12 Mbps[6], hence the throughput curve flattens out at around this maximum throughput. Were the forwarding rate to increase, the value of traffic intensity per node at which this flexion appears would correspondingly increase. Note that the network **is not saturated.** At this value of intensity the total throughput is only 25 Mbps whereas the media capacity is 500 Mbps (100 Mbps per link multiplied by 5 links). The utilisation is therefore only 5% which does not even approach total utilisation of the medium. In order to achieve 100% utilisation of the medium a node would have to be capable of forwarding 100 Mbps on a single link because each Metroring node has a dedicated link to it's neighbouring nodes. However, the node will usually be forwarding frames to more than only one link, thus the forwarding rate may have to be as high as $100 \cdot x$ Mbps, where $x$ is the number of links. This translates into a forwarding rate of 200kfps or more. Currently, not even the dedicated mutiprotocol routers are able to achieve this sort of performance[7], let alone a non-dedicated computing device. Practically, then, the forwarding rate of the nodes will place an upper bound on the possible throughput rates as stated in section 7.1.

### 7.3.2 End-to-end Delay

Figure 7.6(b) shows the average end-to-end delay per frame plotted against the average traffic intensity per node. The delay begins to increase exponentially as the rate of frame arrivals reaches approximately 850fps. At this value of traffic intensity the incoming traffic rate becomes greater than the frame service rate, hence the queue size grows as shown in Figure 7.7, and the delay therefore increases as frames spend greater periods of time waiting in queues which are nearly full.

Figure 7.6(b) shows that the exponential increase in end-to-end delay tapers off after reaching 0.1 sec. The reason that this delay curve flattens out is that the simulation model does not model the

[6] 10000fps . 64bytes/frame 8bits/byte = 5.12 Mbps

[7] see Datamation, 1992, pp. 108-111

## 7.3 General Trends

Two criteria, **throughput** and **end-to-end delay**, are often used as performance metrics (Hammond, 1986) (Rodrigues, 1990). Section 5.2 defines these performance measures in detail. These two performance criteria are valuable when analysing the Metroring protocol. Throughput and end-to-end delay results are derived for a simple Metroring network and are discussed in this section. Simulation experiments are then performed to examine the effect that changes in certain parameters, for example frame size, have on important performance measures.

A simple Metroring network is shown in Figure 7.2(b). The system throughput and end-to-end delay plotted against the traffic arrival rate per node is shown in Figure 7.6(a) and (b) respectively.

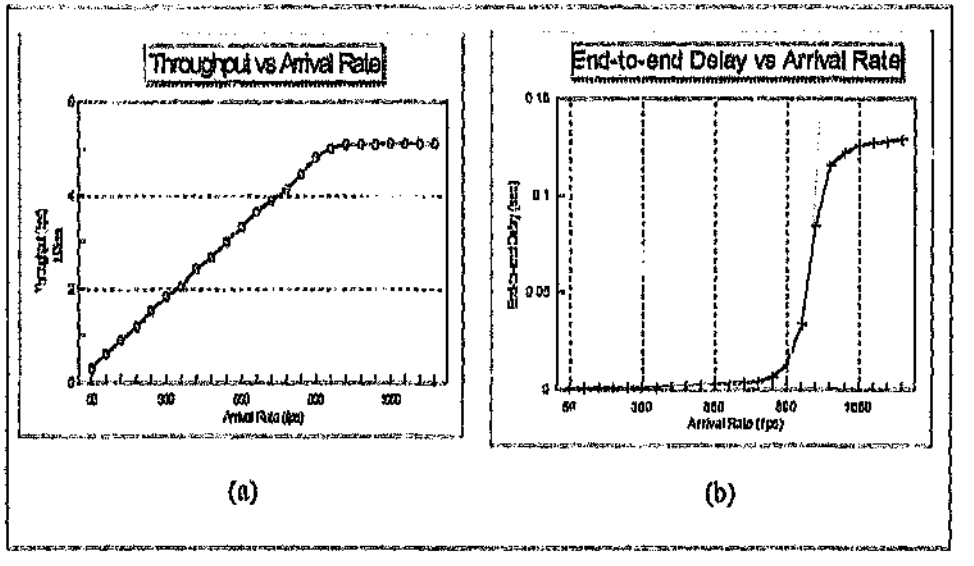



Figure 7.6: Performance Characteristics for a 5 node Metroring

### 7.3.1 Throughput

As traffic intensity increases, the throughput of the network begins to increase as seen in Figure 7.6(a). At approximately 800fps the throughput reaches a maximum value. At this value each

The Metroring simulator throughput results of the dual ring network shown in Figure 7.4(b) also compare favourably with those produced by COMNET. This can be seen in Figure 7.5. The two sets of results are not as similar as in the single ring case.

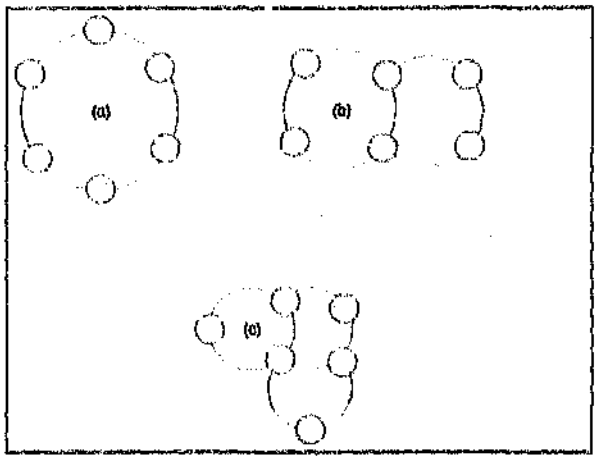



Figure 7.4: Metrorings with single, double and triple rings

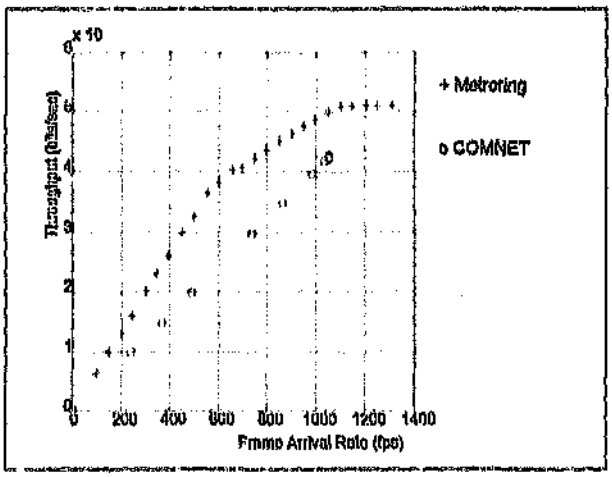



Figure 7.5: COMNET vs Metroring Simulator Throughput

was not able to produce end-to-end delay results as the delay times were too small to be shown in the COMNET output statistics.

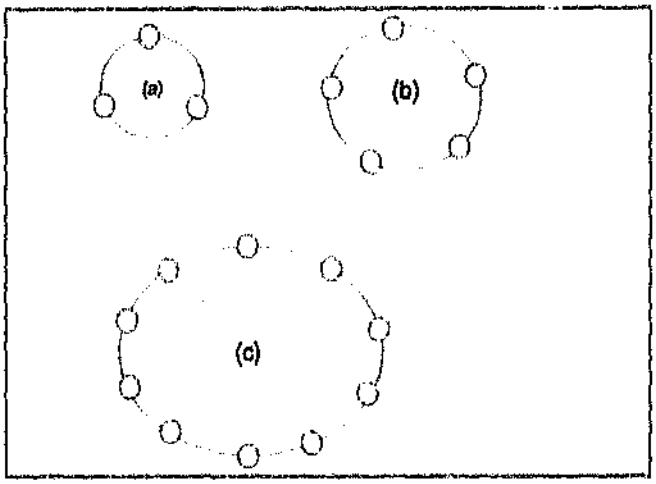



Figure 7.2: 3, 5 and 11 node Metroring topologies

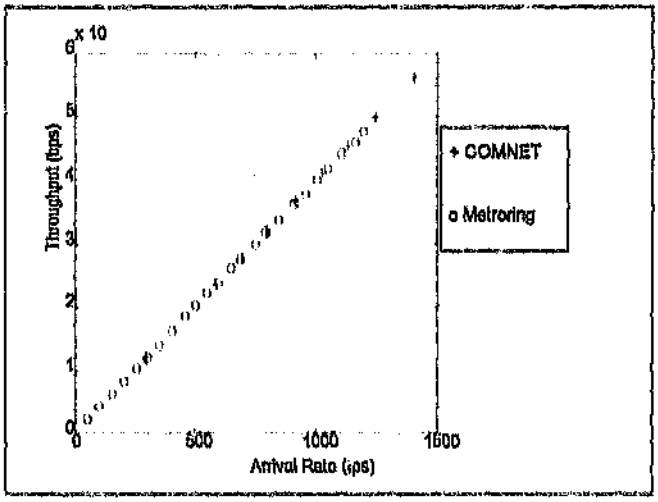



Figure 7.3: COMNET vs Metroring Simulator Throughput Statistics

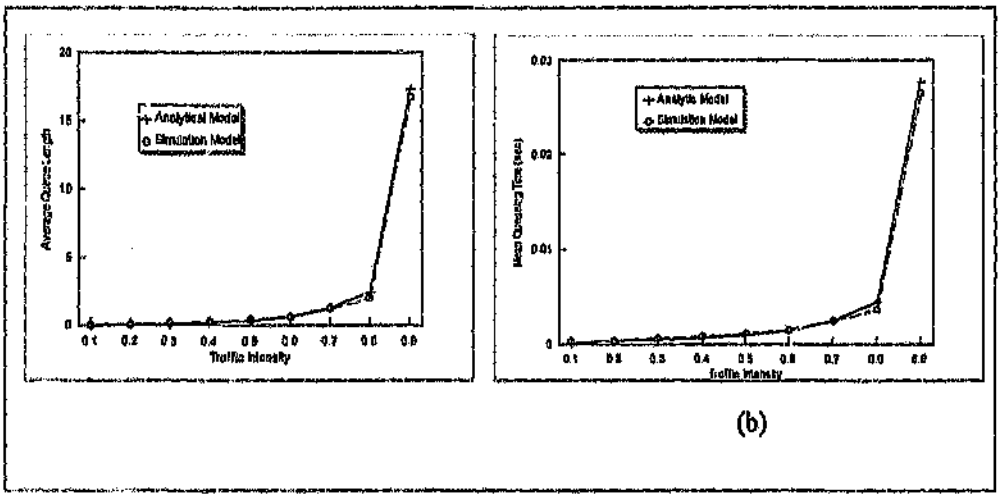



Figure 7.1: Theoretical vs Simulated Results for a single Metroring node

## 7.2.2 Comparison With Other Simulation

An internet simulation package was used to examine the effects observed in section 7.6.2. Results from the simulation package were compared to those of the Metroring simulator for validation purposes. The simulation package used was COMNET II.5 release 5.03 from CACI Products Co., with the permission of WEB Systems in Arcadia, Pretoria. The primary function of this simulation package is the modelling of internets and thus it was not ideally suited to the modelling of Metrorings which have very high data transfer rates. Although the package was running on a Sun workstation the simulation times were significantly high. In addition, COMNET models become extremely complex when trying to model Metrorings in which every node possesses the ability to communicate with every other node in the network. Thus the simulation of large Metrorings was prohibitively difficult.

The Metroring simulator and COMNET results for the three node Metroring depicted in Figure 7.2(a) are plotted in Figure 7.3. The two sets of simulation data are virtually identical. COMNET

The aim throughout these experiments is not to quantify precise values for the performance measures, but rather to use the simulation model to investigate the effect of different variables on the overall performance of the protocol.

## 7.2 Validation Experiments

### 7.2.1 Performance of a Single Node

As a validation test the simulation performance results of a single Metroring node were compared with a simple analytical model of a single node. The results are shown in Figure 7.1.

The M/M/1 queuing model described in Appendix C was used to model the Metroring node with random arrival and service times. The average queuing time is given by,

$$\omega_Q = \frac{\rho}{\mu(1-\rho)}$$

and the average queue length by,

$$L_Q = \frac{\rho^2}{1-\rho}$$

where $\rho$ is the traffic intensity, and $\mu$ is the service rate of a node as described in Appendix C.

$L_Q$ is divided by a factor of two in order to reflect the two queues in a Metroring ring node.

The performance results of a node with a forwarding capability of 10kfps when receiving frames of an average length of 500 bytes is depicted in Figure 7.1. Queuing length and queuing delay are plotted against traffic intensity, $\rho$, in Figure 7.1. It is evident from the figure that the results derived from simulation are almost identical to those predicted by the queuing model.

performance characteristics. The following conditions have been fulfilled throughout these simulations:

- All links are equidistant.
- All nodes in the network possess the same forwarding rate which is specified by the user.
- Each node in the network is equally likely to transmit or receive a frame.

**Forwarding rate**, as defined by Halsall (1992), is the rate at which frames are processed and forwarded for transmission from one port, or segment, to another. Forwarding rate is measured in frames per second (fps), or packets per second (pps). Vendors usually specify this rate using 64 byte frame sizes. Values for Metroring node forwarding rates were derived from router forwarding rates appearing in the literature (Datamation, 1992), (Halsall, 1992), (Bradner, 1991). Forwarding rates from the survey appearing in Datamation range from 500 pps (DEC WANrouter) to 65kfps (65000fps) (Cisco AGS+). The majority of simulations presented in this chapter use a forwarding rate of 10kfps.

The maximum forwarding rate of a node places an upper bound on the number of packets that can be processed per second. It is important to note that the maximum bandwidth achievable by even the fastest dedicated router appearing in the survey above does not remotely approach total utilisation of a fibre optic link. For example, in the case of Metroring where each link has an available bandwidth of 100 Mb, the Cisco AGS+ router is only able achieve a maximum utilisation of 33% with a forwarding rate of 65kfps. As dedicated routing hardware is unable to flood the medium used in Metroring it is fair to assume that a Metroring node, which is very often some form of workstation that is not dedicated to the task of routing, is also unable to achieve 100% utilisation of the medium.

The simulation experiments which follow assume a uniform link distance of 1000m, frame sizes distributed about a mean of 500 bytes, and a run length of 5 seconds of simulated time, unless otherwise specified.

# Chapter 7: Results

The simulation model described in the preceding chapters is used to examine the effect of certain parameters on the performance of the Metroring protocol. The simulation software executes a series of simulations using progressively higher traffic loading. Important measures of performance such as throughput and end-to-end delay are gathered for analysis purposes. These simulation experiments are presented in the following sections.

## 7.1 Simulation Environment

The model assumptions listed in chapter 5 are inherent in the simulation environment. Throughout many of the simulation experiments the average arrival rate of frames at nodes is regarded as the independent variable. The average arrival rate is a measure of traffic intensity. Traffic intensity is defined to be the rate of new traffic requests at each node in bits per second (Shanmugan et al, 1992). This parameter gives an indication of offered load. Traffic intensity is iterated over a wide range in order to simulate the entire spectrum of offered load.

In addition to the assumptions listed in chapter 5, a number of additional conditions are also satisfied in order to minimise the number of parameters that vary during a simulation run. By specifying these additional criteria the effects of the dependent variable are not obscured by secondary parameters. For example, when examining the effect of frame size on the performance of a particular network it is important that the link lengths, for instance, remain constant. If this were not the case performance trends could be attributable to either the variation in link lengths or the variation in frame size, or a combination of the two. This ambiguity is undesirable when analysing

The face validity of the output from the Metroring model is intuitive. Referring to the results in Chapter 7, Figure 7.6 shows curves for throughput and end-to-end delay which mirror those presented in Chapter 5, Figure 5.1. In addition, one can see that throughput increases with increasing frame size while end-to-end delay will increase exponentially.

The Metroring model was also validated by comparing simulation outputs to analytical models. Figure 7.2(b) shows a basic 5 node Metroring, all nodes being ring-nodes. These ring-nodes may be represented by queuing models. Section 7.2.1 compares simulation statistics taken from a node on the 5 node Metroring with those from a M/M/1 analytical model. The two correspond closely as one would expect (see Figure 7.1).

As validation is an ongoing process, the modeller must weigh up the possible, but not guaranteed, increase in model accuracy against the cost of increased validation effort. No model will ever be a perfect representation of the system and so the modeller must decide the point at which his model is sufficiently accurate enough before becoming inefficient from a cost perspective.

particularly useful step-through tool that was used extensively in debugging the Metroring Simulator software.

- Any measure of performance that can be computed analytically and then compared to its simulated counterpart provides another valuable tool for verification. If the simulation model is predicting one measure of performance correctly, the confidence in the model's predictive ability for other related measures is increased.

### 6.4.2 Validation

The verification and validation processes are usually conducted simultaneously by the modeller.

The most common method of validation is **calibration**. This is the iterative process of comparing the model to the real system and making the relevant changes to the model to yield a more accurate representation of the system. The comparison of the model with reality is carried out by subjective tests as well as objective tests. A necessary condition for the use of objective testing is that some version of the system under study already exists so that the system data can be collected to compare to the model predictions. As there is currently no Metroring in operation this method is not applicable for this study.

Whereas objective tests require data on the system behaviour, subjective tests have no such requirement. An important subjective aid in validation is to build a model with a high face validity. This step involves building a model that appears reasonable on its face to users and others who are knowledgeable about the system. One method of assessing the face validity of the model is by observing if the model behaves in the expected way when one or more input variables is changed. For example, if the arrival rate of frames increased, one would expect the utilisation and end-to-end delay time to increase as well. The modeller would have some notion at least of the direction of the change in the model output when an input variable is increased or decreased. As there are many variables, and therefore many possible tests, the modeller should choose only the most critical variables for testing.

a clearer pattern may emerge for deciding the optimum Metroring topology for a given application.

- Further research may be performed to clarify the distinctive inflections appearing in end-to-end delay curves for multi-ring Metroring networks. Some method of load sharing or "buffer balancing" to ensure optimum delay characteristics may be derived from this investigation.
- The performance characteristics of the Metroring protocol in a transient state could also be examined. The ability of the protocol to recover from nodal or link failure would be a major benefit emerging from this study.
- Research into the relationship between delay and parallelism in general internets as stated in 8.1.

## 8.4 Enhancements to the Simulation Model

A number of enhancements may be added to the Metroring model constructed as part of this research

- The ability to model larger networks by overcoming memory constraints imposed by the implementation phase.
- Implementation of a routing mechanism that is more representative of the real Metroring routing mechanism.
- Facilities to handle Fast Forward, and multi priority frames.
- Embedded GUI (Graphical User Interface) enabling easy construction of simulation models and graphical analysis of performance results.
- Embedded statistical analysis of performance measures.

- The results verify one of the major functions of the Metroring specification, viz. high throughput.

## 8.2 Applications

The performance characteristics presented in this dissertation help to identify the environments to which the Metroring protocol is best suited. Metroring's combination of fibre optic technology, spatial reuse and concurrent access, make it suitable for many applications and environments.

The Metroring simulation model may be used as an aid in the development of the protocol. The simulation model could be used to identify optimum design parameters. For example, experiments using different queue sizes might be performed to establish an optimum queue size and how this size should be adjusted in the case of SENs. The simulation model provides an insight into the behaviour of the protocol and could therefore uncover inconsistencies in design logic. The Metroring simulator's unique flexibility is well suited to the modelling of a protocol that is still undergoing development.

Performance results may also be used to define a distinct market for Metroring. The simulation model may be used to compare Metroring performance with that of competing protocols to aid in choosing a particular protocol to fulfil specific needs.

## 8.3 Future Research Regarding Performance Issues

The object of this research was to examine the basic performance characteristics of the Metroring protocol. There are several specific issues regarding Metroring performance analysis which may be examined in greater depth.

- Perhaps the most interesting area in which this research may be extended is in the analysis of the effect that topology has on performance. By modelling a wider range of topologies

# Chapter 8: Conclusion

The performance characteristics of the Metroring protocol were investigated in this dissertation. The simulation model constructed in the object-oriented paradigm provides a good tool for performance analysis of the Metroring protocol. Concurrent access and simultaneous transmissions increase effective throughput. The efficiency of the Metroring architecture does not decrease as the physical size of the network increases. High throughput to load ratios and high speed data transfers make Metroring ideal for networks exhibiting a high traffic loading. The Metroring protocol simulation model implementation has proved successful in altering the Metroring protocol specification (eg. page 77). This was one of the primary aims of this research.

## 8.1 Features of the Research

This research includes several features:

- Existing models could not be adapted to the Metroring requirements and therefore a Metroring simulation model has been developed and tested.
- The simulation model was implemented in the object-oriented paradigm. This results in a number of secondary features:
  - Flexibility.
  - Reusable modules are portable to other discrete event simulations.
  - The simulation model has the potential to model other protocols.
- The relationship between delay and parallel paths, shown in Figure 7.12(b), could find applicability in internets as hinted at in current research (Shankar, 1992).

will reach maximum utilisation before an equivalent multi ring Metroring. This aspect of Metroring performance requires a study in its own right. Many simulations need to be performed on Metrorings of much greater size to establish well defined criteria for choosing multi or single ring Metroring topologies.

1. A dedicated processor for every link in each node. This would distribute the traffic processing more or less evenly among processors ensuring that queues in the Metroring network filled at an approximately uniform rate. The commercial Metroring networks would implement this form of load balancing via dedicated processors on network interface cards.
2. Some form of buffer balancing where SENs would have queues that were larger than ring [illegible] a factor derived from the intensity of traffic at these two types of nodes. [illegible] SEN receiving, on average, three times the amount of traffic that a ring [illegible] receives would have an average queue size three times the size of the ring node queue. This method would be extremely difficult to implement and manage.

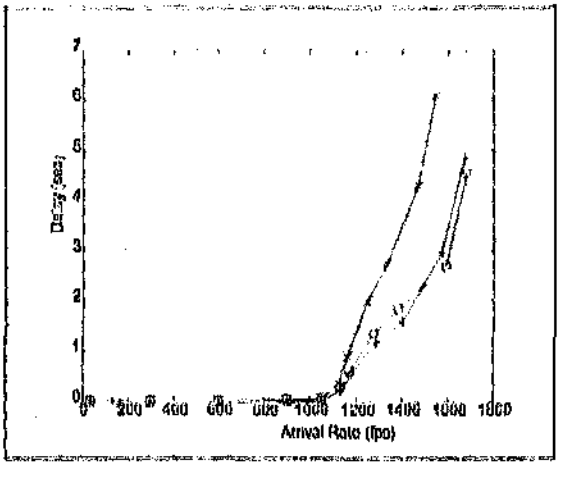



Figure 7.13: COMNET End-to-end Delay Results

The graphs in Figure 7.11 and 7.12 would tend to indicate that expanding a Metroring via additional segments, as opposed to additional nodes on a single ring, could lead to a less dramatic increase in end-to-end delay. However, the end-to-end delay would become significant sooner in a multi ring network than in a single ring network. From the graphs, the manner in which the Metroring is expanded will not impact the throughput significantly although a single ring, Metroring

The distinct shapes of the delay curves are attributable to the simulation software. The Metroring simulator discards frames once the queues within nodes become too large. This leads to a flattening out characteristic as discussed in section. It is evident from Figure 12(b) that the number of SENs present in the network corresponds to the number of exponential increases which the delay curve experiences. The dual ring delay curve has two distinct increases in end-to-end delay, while the delay curve for the Metroring containing three rings experiences three distinct increases in delay. It is interesting that the second exponential increase in the dual ring delay curve occurs at approximately twice the arrival rate of the first, namely at 1kfps compared with 500 fps. This reflects the symmetrical topology of the dual ring network.

Although the appearance of the end-to-end delay curves for multiple rings is due to the simulation model simplifications, one would expect similar characteristics when analysing the real world system. The end-to-end delay curves will experience an exponential increase when a node, or group of nodes, experience a greater traffic arrival rate than they are able to process. In the case of a multiple ring Metroring network, this will occur at several distinct intervals because the SENs will begin to discard frames sooner than the ring nodes[9]. Therefore one would expect a general trend of exponential increases in end-to-end delay at distinct intervals. In addition, one would expect these intervals to become less clearly defined as the number of rings, and consequently the number of SENs, increases. These assertions are confirmed by the end-to-end delay results produced by COMNET models of each of the topologies shown in Figure 7.4. The COMNET models were constructed using speeds far lower than those attainable by Metroring because of COMNET's lack of support for very high speed network modelling as mentioned in section 7.2.2. All links between nodes were defined to have a 100 kbps data transfer rate in the COMNET model. The delay characteristics of the COMNET model resemble those shown in Figure 7.12(b). The investigation by Shankar et al (1992) also exhibits a similar characteristic.

From the above discussion two methods of obtaining better end-to-end delay performance characteristic emerge:

[9] assuming a single processor in each node and uniformity of queue lengths throughout all nodes in the network.

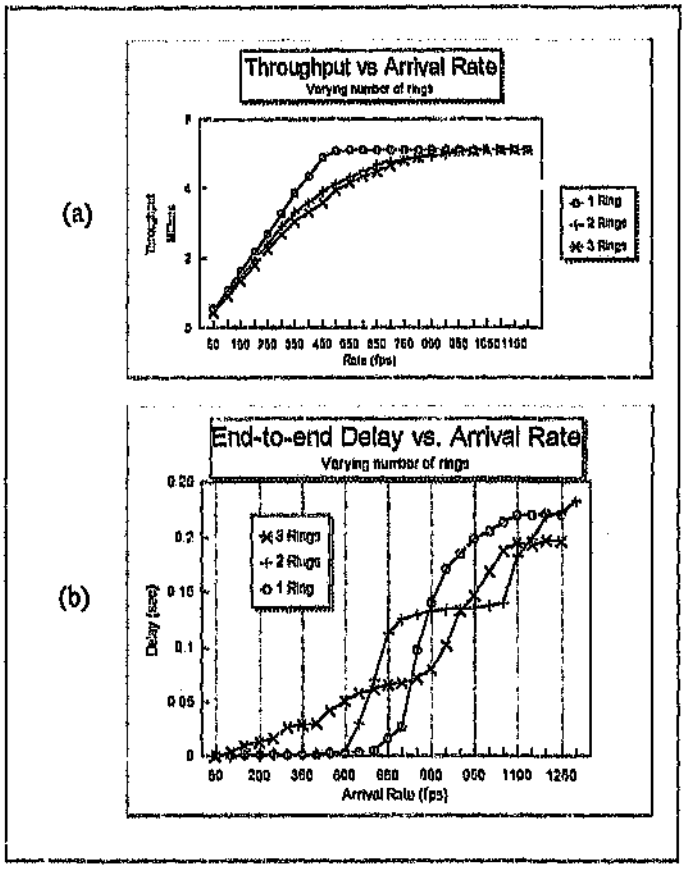



Figure 7.12: Performance Characteristics With Varying Number of Rings

From Figure 7.12(b), the average end-to-end delay in a single ring network only begins to proliferate at a much higher load than the two, three and four ring networks. This characteristic is attributable to waiting times within SEN node queues. A SEN may have three links or more, each acting as a source of traffic. This means that the SEN needs to service a greater volume of traffic than a ring node does. A SEN node's queues will therefore fill more rapidly than those in a ring node because the simulation model uses a single processor per node which is having to serve a greater number of requests in the case of a SEN. Consequently frames will experience greater delays when passing through a SEN than when passing through a two port node.

generate frames as in the case of the 5 node Metroring. This results in twice the throughput following the definition of throughput given in section 5.2.1.

The graphs also show that the rate at which Metroring networks reach their maximum throughput is dependant on the number of nodes in the Metroring. The 10 node Metroring reaches a maximum throughput value at around 500 fps while the 5 node network reaches its maximum throughput at approximately 850 fps. The more rapid increase in throughput in the larger Metrorings is due to the increased loading of the network. The greater number of stations cause this increase in traffic.

The rapid increase in network traffic in the larger Metrorings also results in larger end-to-end delays as shown in Figure 7.11(b). As in the case of throughput, the relationship between delay and number of nodes is roughly linear. End-to-end delays begin to increase rapidly at about 400 fps in the 10 node Metroring, while the 5 node Metroring only begi . to experience an increase in end-to-end delays at about 800 fps. Buffers in the 10 node Metroring are filled more rapidly than in the other two Metrorings because of the greater loading. This in turn leads to the rapid increase in end-to-end delay.

### 7.6.2 Number of Rings

A common way that a Metroring network can expand is via the addition of segments creating additional rings. It is important to evaluate the effect which this change in topology has on the system performance. This is achieved by keeping the number of nodes constant and varying the number of rings in the network model. The results are plotted in Figures 7.12.

Each of the four Metrorings contain 6 nodes. Hence their maximum throughput should be identical. This is shown in Figure 7.12(a). There is little difference in the throughput curves, however it is noticeable that the rate at which throughput increases does vary according to the number of rings in the Metroring. This is most noticeable when comparing the single ring network to the multi-ring networks.

### 7.6.1 Number of Nodes

The most simple manner in which a Metroring network can grow is simply by the addition of nodes to the ring. The three node, five node, and ten node network configurations shown in Figure 7.2 were simulated and their performance characteristics plotted in Figures 7.11.

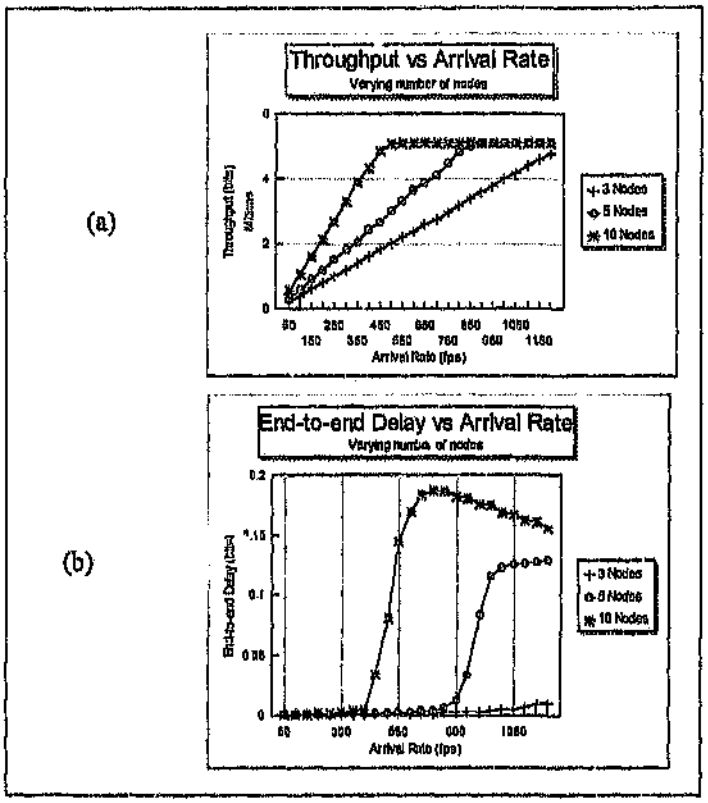



Figure 7.11: Performance Characteristics With Varying Number of Nodes

The variation in throughput with average arrival rate per node for the 3 Metroring topologies is depicted in Figure 7.11(a). It can be seen from the plotted data that throughput varies linearly with the number of nodes, i.e. the 10 node Metroring yields twice the throughput that the 5 node Metroring is able to deliver. In the 10 node Metroring there are twice as many nodes able to

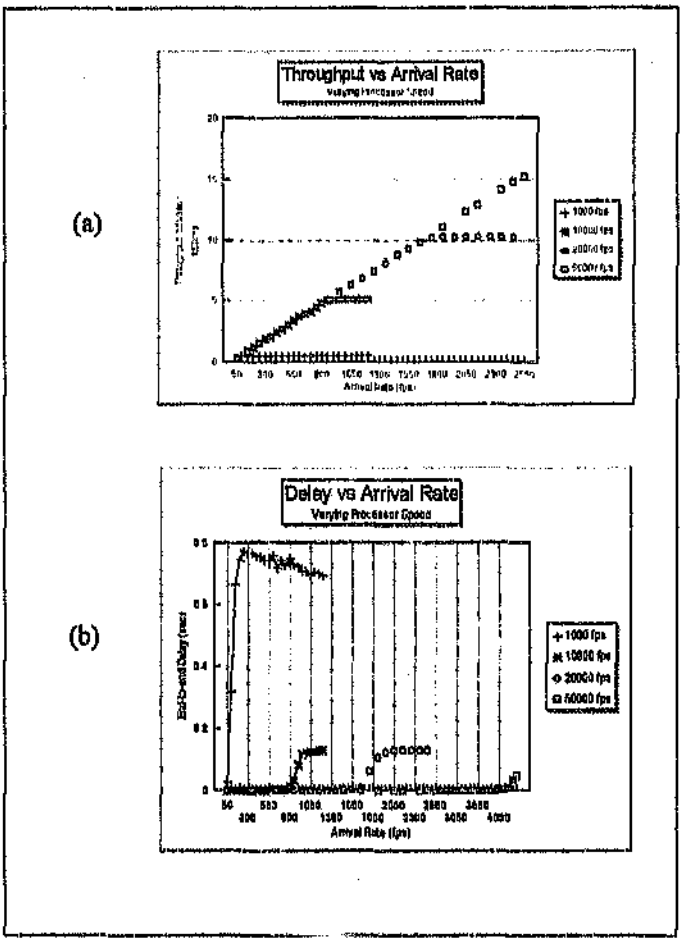



Figure 7.10: Effect of Processor Speed on Performance

## 7.6 The Effect of Topology on Performance

A Metroring may assume many different possible physical topologies. The Metroring simulation model is an important aid in appreciating the effect of different physical network topologies on the overall network performance. This section presents the results of simulation experiments carried out to illustrate these effects.

It is evident from Figure 7.9(a) that larger frames cause the Metroring to attain maximum throughput more rapidly than small frames do. In fact the Metroring does not reach maximum throughput within the simulated load range at frame sizes of 100 bytes. In order for the Metroring to reach maximum throughput an arrival rate in excess of 6kfps, using 100 byte frames, would need to be generated.

## 7.5 The Effect of Processing Rate on Performance

The forwarding rate of nodes has a marked effect on the performance of a Metroring network. Figure 7.10(b) depicts the end-to-end delay with nodes having a forwarding rate of 1k, 10k, 20k and 50kfps respectively. The Metroring containing the fastest nodes is able to withstand a much higher loading before delay times begin to increases dramatically. The Metroring with nodes having a 50kfps forwarding rate only experience an increase in the end-to-end delay at an average arrival rate of 4.2kfps compared with 1.7kfps and 700fps for nodes with a 20kfps and a 10kfps forwarding rate respectively. This characteristic is intuitive as nodes with a high forwarding rate can process frames faster than nodes with a lower forwarding rate and, therefore, will need a much higher traffic intensity before their queues begin to fill.

Note also that the increase in delay becomes steeper as the forwarding rate decreases. This is because the lower the node forwarding rate, the longer it will take to process a frame, hence frames will back up faster at nodes with a lower forwarding rate.

Data throughput for Metroring networks with nodes having 1kfps, 10kfps, 20kfps and 50kfps forwarding rates is plotted in Figure 7.10(a). As one would expect, higher forwarding rates translate into a higher throughput values. When each node has a 50kfps forwarding rate the total data throughput is around 25 Mbps compared with 5 Mbps for a 10kfps forwarding rate.

proportional to frame length (Sunshine, 1987). The additional time spent processing larger frames means that these large frames spend more time waiting in queues than short frames do. It follows that a series of short frames will pass through a node's buffers more rapidly than a similar series of longer frames will. From Figure 7.9(b) it is noticeable that there is no evident increase in end-to-end delay for 100 byte frames. This is attributable to the lack of queuing delays because the Metroring nodes are able to process the small frames faster than they arrive.

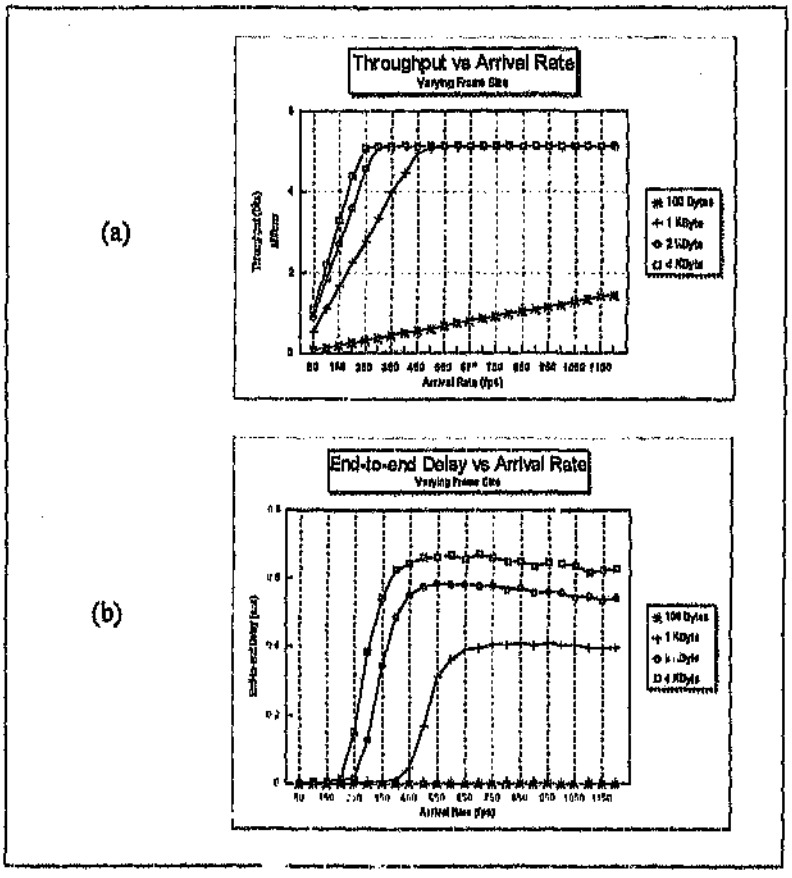



Figure 7.9: The Effect of Frame Size on Performance

$$L_Q = \frac{\rho^2}{1-\rho}, \text{ and}$$

$$\omega_Q = \frac{\rho}{\mu(1-\rho)}$$

it is important to note that the Poisson and exponential distributions are intricately linked. If the interarrival times are exponentially distributed and independent, then the number of arrivals by time $t$, say $N(t)$, meets the 3 assumptions listed below, and is therefore a Poisson process (Banks, 1984).

1. Arrivals occur one at a time.
2. Arrivals occur completely at random without any rush or slack times, i.e. the distribution of the number of arrivals in an interval of time depends only on the length of the interval, and not on the starting point of the interval.
3. The number of arrival occurring during nonoverlapping time intervals are independent random variables. Future arrivals occur completely at random, independent of the number of arrivals in past time intervals.

These assumptions are therefore inherent in the Metroring simulation model as exponential interarrival times are assumed.

To summarise using the corollary, if the arrival process of the customers in a queuing system follows the Poisson distribution, with arrival rate $\lambda$, then the interarrival distribution will be exponential with mean $\frac{1}{\lambda}$.

## C.4 The M/M/1 Queue

Service and interarrival times are exponentially distributed with means $\mu^{-1}$ and $\lambda^{-1}$ respectively. Thus $\lambda$ represents an arrival rate, for example 100fps. Similarly $\mu$ represents a service rate. The quantity $\frac{\lambda}{\mu} = \rho$, represents traffic intensity.

The parameters $L_q$ and $\omega_q$, representing average queue length and average queuing delay respectively may be computed as discussed in (Banks, 1984, pp.200-202). These parameters are given by,

variates from other distributions using these random numbers. This is the method used in this simulation.

A uniform random variable over the interval $[a, b]$ has a probability distribution function (pdf) given by,

$$f(x) = \begin{cases} \frac{1}{b-a}, & a < x \leq b \\ 0, & \text{otherwise} \end{cases}$$

and a cdf given by

$$F(x) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ 1 & x > b \end{cases}$$

The mean is given by, $\frac{a+b}{2}$, and the variance by, $\frac{(b-a)^2}{2}$

## C.3 The Exponential Distribution

The exponential distribution has been used to model interarrival times when arrivals are completely random and to model service times which are highly variable (Banks, 1984). The exponential distribution pdf is given by,

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

for $\lambda > 0$, where $\lambda$ is a rate, for example packets per second.

The cdf is given by,

$$F(x) = \begin{cases} 0, & x < 0 \\ \int_0^x \lambda e^{-\lambda t} dt = 1 - e^{-\lambda x}, & x \geq 0 \end{cases}$$

The mean and variance of the distribution are given by $\frac{1}{\lambda}$ and $\frac{1}{\lambda^2}$ respectively.

# Appendix C: Probability Distributions and the M/M/1 Queue

## C.1 The Poisson Distribution

The Poisson distribution is a discrete probability distribution. It describes many random processes well and has been widely used to model arrival distributions (Graybeal, 1980). The probability mass function (pmf) is given by

$$p(x) = \begin{cases} \frac{e^{-\alpha}\alpha^x}{x!}, & x = 0, 1, \ldots \\ 0, & \text{otherwise} \end{cases}$$

The cumulative distribution function (cdf) is given by,

$$F(x) = \sum_{i=0}^{x} \frac{e^{-\alpha}\alpha^i}{i!}$$

An important property of the Poisson distribution is that its expected mean value and variance are equal, i.e. $\alpha = \sigma^2$

## C.2 The Uniform Distribution

The uniform distribution is a continuous distribution. It is used to generate random numbers, uniformly distributed between 0 and 1. The uniform distribution plays a vital role in simulation as it provides the means to generate random events. In computer simulations it is customary to generate random numbers according to a uniform distribution and to then generate samples of random

## B.3 Flow Control

Typically some nodes in any network will transmit and process frames faster than others. Hence, nodes need some way of ensuring that they do not lose frames if they are receiving frames at a quicker rate than they are able to process them. Some form of flow control is essential in any networking protocol in order to avoid the situation described above.

The Metroring uses a number of queues and buffers in which to store frames. Each link has a buffer associated with it. This buffer is used for frame assembly when receiving frames and to store frames awaiting transmission. There are, in addition, three other queues in each node: a receive queue, a send queue and a transmitted queue. The receive queue is used to queue frames waiting to be processed by a node's CPU. All frames to be transmitted are first loaded into the send queue. These frames are then stored in the transmitted queue until they have been acknowledged. Should each link have a separate processor, it would correspondingly have the queues mentioned above associated with it for greater efficiency.

Metroring is made up of more than one segment, the SEN contains the addresses and shortest paths to all bridges attached to the region which that SEN belongs to.

The highest level in a Metroring routing hierarchy is the bridge table. Bridge node tables contains the same basic information that a SEN does, however, the bridge table will contain entries for every other bridge in the network.

### B.2.2 Routing Tables

A routing table contains a row for each destination, and a column containing the shortest path to that node or area. Table B.1 shows a Metroring routing table. The destination region and the destination node is represented by the alpha-numeric number appearing in the first column of the table. The link providing the quickest route to the destination is listed in the second column. The final column gives the cost of the route in terms of hop count. The routing tables used by the Metroring simulator are constructed in the same manner as the table depicted in Table B.1.

Routing tables are updated dynamically. Management frames notify nodes in the Metroring of the attachment or removal of nodes, node failure and link faults. Once a node has received this information, it is able to update its routing table accordingly.

| Destination | Port or Link | Hops |
|---|---|---|
| A0 | 0 | 1 |
| B1 | 1 | |
| C0 | 2 | 3 |
| C1 | 2 | |
| B0 | 1 | 3 |

Table B.1: A Metroring Routing Table

path between source and destination, and routes the frame along this path. The shortest path may be determined according to the shortest physical distance, the least number of node hops, or the lowest mean queuing and transmission delay times. The Metroring routing method employs frame propagation delay as a criteria for determining the shortest path to a destination. Although this method results in a low end-to-end delay time, its implementation imposes high overheads in terms of node processing power and storage capacity. For this method to be implemented in a Metroring, each node would need to store a routing table with entries for **every other node on the network**. This is obviously unreasonable in the case of a large Metroring consisting of many nodes and many regions.

Ideally, in the case of a Metroring network, we would like a method that results in the efficiency of a shortest path strategy while keeping the size of routing tables to a minimum. Hierarchy routing is an attempt to fulfil this requirement.

### B.2.1 Hierarchy Routing

A single level address space requires each node performing routing (in the case of a Metroring this is *every* node in the network) to know the correct route to every possible destination independently, while a hierarchical address space allows routing nodes to know correct routes only to destinations within the local "area" and to other areas (Schoemaker, 1978).

Hierarchy routing is implemented by providing a number of different classes of routing table. The most simple of these is the table contained within every two-port node. This table contains an entry for every node on that segment together with the shortest path to reach that node. The two-port node routing table also contains addresses for every SEN on that segment, as well as the shortest route to reach the SEN.

The next level in the hierarchy is the SEN table. This table contains entries for every node on every attached segment together with their respective shortest paths. The SEN table also contains the addresses and shortest paths to all other SENs within that particular segment. Lastly, if the

4. The receiving node sends a NACK to the transmitting node when an error is encountered. An error would be detected should the CS (check sequence) of the frame be wrong, or if the received frame's sequence number is out of sequence. The corrupted, or missing frame's sequence number is sent to the transmitting station together with a NACK.
5. Upon the reception of a NACK, the transmitting node will retransmit all frames with a sequence number equal to and greater than the corrupted frame's sequence number. This retransmission strategy is called **go-back-N**.
6. All frames with a sequence number less than the corrupted frame's sequence number are removed from the transmitting station's queue as they have been received correctly by the receiving node.

Although the continuous RQ scheme offers improved utilisation, it is at the expense of larger transmission queues due to the retention of frame copies.

## B.2 Routing

Shaikh (1989) defines the primary function of a routing algorithm to be the determination of a path that frames must follow from a source to a destination in such a way that the end-to-end delay is minimised.

Two basic methods of routing exist: adaptive and non-adaptive methods. **Non-adaptive** or **static**, routing methods do not react to network topology changes or routes that are congested with traffic. A route is computed in advance and that route is fixed. **Adaptive** or **dynamic** routing methods, on the other hand, seek to continually include changing traffic and topology changes in their routing decisions. The inflexibility of the static routing methods renders this method inadequate for the high speed, highly efficient Metroring network. Although the Metroring protocol uses dynamic routing, the Metroring simulator employs static routing for simplicity.

There are a number of adaptive routing schemes that vary in complexity, from the shortest queue, or "hot potato" method to the shortest path method. The latter method computes the shortest

# Appendix B: The Metroring Protocol

## B.1 Error Control

When a computer transmits a message to another computer over a potentially unreliable medium, it will normally retain a copy of that message for retransmission in case of failure of the transfer (Sauer, 1983). The retransmission will be triggered by one of two events: a time-out period transpires or either a positive or negative acknowledgement is received by the sending node. This process must take place automatically, and must be transparent to the user. The method of error control described above is known as automatic repeat request and there are two types in practise. The first is known as **idle RQ** (repeat request). The idle RQ scheme requires that the transmitting node wait for a positive or a negative acknowledgement (ACK or NACK respectively) before transmitting another frame. For a detailed explanation of this method of error control the reader is referred to (Halsall, 1992). The second method of automatic repeat request error control is called **continuous RQ**. This strategy does not require an acknowledgement from the receiving station before transmitting the next frame, and thus a continuous stream of information may be transmitted.

The Metroring protocol uses a type of continuous RQ (repeat request) error control strategy. This error control scheme has a much better link utilisation than the idle RQ scheme. The operation of this method proceeds as follows:

1. The transmitting station sends frames continuously without waiting for an acknowledgement from the receiving station.
2. Each transmitted frame is marked with a unique sequence number.
3. A copy of each transmitted frame is retained in the transmitting station.

The LCP Layer ensures the correct, sequential transfer of frames between Metroring nodes. Error control and flow control are two major functions of the LCP Layer. These functions are described in Appendix B.

The RCP Layer is responsible for routing. The RCP is a hierarchical routing method that selects the shortest path, in terms of frame propagation delay, between source and destination. The routing method adopted by Metroring is also discussed further in Appendix B.

The Management Protocol Layer (MP), is responsible for administering all aspects of node interactions within the Metroring network. The most important Metroring functions are:

- Controlling the method by which a new node attaches to the Metroring network.
- Informing all relevant nodes of any node and media faults and of the correction of these faults.
- Ensuring node routing tables accurately reflect the current Metroring network configuration.

For a more detailed discussion of the Metroring protocol stack, the reader is referred to (Tjirkalli, 1993).

result. The control of such congestion is another function of the network layer (Tanenbaum, 1988).

- *Layer 4: The Transport Layer.* Provides an interface between the networking and the application oriented layers. It essentially provides the higher level applications with a network-independent, reliable message interchange service.
- *Layer 5: The Session Layer.* Is responsible for initialising and clearing a dialogue channel between two peer application entities.
- *Layer 6: The Presentation Layer.* This layer is concerned with the syntax of the data during a dialogue session between two application entities.
- *Layer 7: The Application Layer.* Provides the user with a number of information services. These include file transfer access and management, and electronic mail.

## A.2 A Comparison of OSI and Metroring

A comparison of the Metroring and OSI Reference Model protocol layers is shown in Figure 4.4.

From the figure, the most distinct feature of the Metroring protocol stack is the combination of the Network and Data Link Layers of the OSI stack into a single layer, named the Path Layer. It is also evident from the figure that the Path layer incorporates functions of the networking layer of the OSI model. This reflects the routing function that Metroring nodes perform.

The Path Layer is divided into the Metroring Media Access Control Layer (MRMAC), the Link Control Protocol Layer (LCP), the Route Control Protocol Layer (RCP) and the Management Protocol Layer (MP).

The MRMAC layer is responsible for access control to and from the physical media. As there are no media contention issues in the Metroring protocol, the MRMAC Layer is significantly simplified from corresponding MAC Layers in other protocols such as Ethernet. Point-to-point, full duplex links also simplify the MRMAC Layer. The MRMAC Layer controls the input output buffers and keeps neighbouring nodes informed of buffer availability.

# Appendix A: OSI and Metroring

## A.1 The OSI Seven Layer Reference Model

Computing equipment supplied by many different vendors may coexist on any single network. In order for these devices to communicate with one another successfully, some form of universal data exchange protocol is needed. The ISO Reference Model for Open Systems Interconnection (or the seven layer model) was developed to fulfil this need. The model provides a vendor-independent standard concerned with structuring of the communications software that is needed to provide a reliable communication service to support a wide range of applications (Halsall, 1988). Layers 1 to 3 of the reference model form the network dependent layers, while layers 4 to 7 constitute the application oriented layers.

- *Layer 1: The Physical Layer.* The physical layer consists of all the mechanical and electrical components necessary to link computing equipment. This layer is concerned with transmitting raw data bits over a communication medium.
- *Layer 2: The Data Link Layer.* The function of this layer is to provide a reliable information transfer facility to the rest of the network. The Data Link Layer is split into two sublayers - the MAC and LLC layers. The MAC layer ensures fair access to the physical medium by nodes attached to the network. The LLC layer is concerned with the provision of a reliable link for data communications on the network. The major functions of the Data Link Layer are the fragmentation of data into frames, error control and flow control.
- *Layer 3: The Network Layer.* This layer is responsible for the establishment of a logical link between two nodes on the network. A key function of this layer would be frame routing. If too many frames are present in any section of the network, "bottlenecks" will

| | |
|---|---|
| getEvent | retrieve event from future events list |
| initSimulation | initialise simulation parameters |
| eventListHandler | procedure to handle various events |

Class: **RingNode**

Description: Class defining a Metroring ring node

Base Class: BaseNode

Methods:

| | |
|---|---|
| setupTable | setup a routing table for the node |
| lookUp | find a frame's destination address |

Class: **SEN**

Description: Class defining a Metroring Segment End Node

Base Class: BaseNode

Methods:

| | |
|---|---|
| setupTable | construct a routing table for the node |
| lookUp | find a frame's destination address |

Class: **Simulator**

Description: Simulation engine

Methods:

| | |
|---|---|
| putEvent | place event in future events list |

Base Class: MetroringObject

| | |
|---|---|
| receiveFrame | process for receiving a frame |
| transmitFrame | process for transmitting a frame |
| getFirstItem | returns the first item in the queue |
| getLength | returns the length of the queue |

Class: **Region**

Description: Class representing a Metroring region

Base Class: MetroringObject

Methods:

| | |
|---|---|
| addNode | adds a node to the region |
| setupNodes | initialises the nodes in a region |
| getNumberNodes | returns the number of nodes in the region |

Class **Random**

Description: Class to generate a Random Integer between 0 and r-1. This generator uses the linear congruental random number generator.

Methods:

| | |
|---|---|
| uniform | returns a random uniform number |
| exponential | returns an exponentially distributed random number |

+----------+

Class: **MetroringObject**

Description: Abstract Base Class for Metroring Objects

Base Class: Object

Methods:

| | |
|---|---|
| nameOf | returns the name of the class |
| printOn | returns contents of object |

+----------+

Class: **Network**

Description: Defines Metroring network

Base Class: MetroringObject

Methods:

| | |
|---|---|
| reset | resets all variables associated with the network |
| setupRegions | creates necessary regions in the network |
| getNode | returns any node object in the network |

+----------+

Class: **NodeQueue**

Description: Class providing structure for queues in Metroring nodes

Methods:

| | |
|---|---|
| isEqual | tests whether event is equal to another |
| isLessThan | tests whether event is less than another |
| getTime | return the time associated with an event |

+----------+

Class: **Frame**

Description: Class defining the Metroring frame

Methods:

| | |
|---|---|
| printOn | displays object contents |
| getDestNode | returns destination node address |
| getLength | returns the frame length |
| getTime | returns time that frame was generated |

+----------+

Class: **Generator**

Description: Generates frames and events

Methods:

| | |
|---|---|
| generateFrame | generates a new frame |
| generateEvent | generates a new event |
| generateExpon | generates an exponentially distributed random number |

# Appendix F: Metroring Simulator Class Reference

The main classes in the Metroring simulator are listed with a description and several important methods in each:

Class: **BaseNode**

Description: Class providing structure for a Metroring node.

Base Class: MetroringObject

Methods:

| | |
|---|---|
| setupQueues | creates and initialises the queues |
| setupTable | initialises routing table from an ASCII file |
| lookUp | finds destination in routing table |
| receiveEvent | handles frame arrival |
| transmitEvent | handles frame transmission |

---

Class: **BaseEvent**

Description: Structure for events in the future events list

Base Class: Sortable

3. **Library Classes.** C++ provides many useful library classes. Entities such as queues are available as compiled classes for the user to incorporate into his software. The existence of these class libraries offsets, to some degree, the benefits gained by a special purpose simulation language. Some basic building blocks provided by languages such as SIMSCRIPT and SLAM are provided by C++ and others may be obtained from third party sources, or alternatively, they may be easily written using the components provided by C++.

# Appendix E: Justification For Using C++ as the Implementation Language of Choice

The object-oriented model described in Chapter 6 was implemented using Borland C++ version 3.1. C++ has evolved from its predecessor, C, an extremely popular and powerful programming language. C++ remains compatible with C while providing all the additional functionality necessary to support the object-oriented paradigm. Thus C++ supports the elements discussed in Section 6.1.1. C++ is a very efficient language (Jordon, 1990) offering performance advantages over other languages. It is also widely available and has an abundance of tutorial material which aids in the learning process. Major factors important in the selection of C++ are:

1. **Inheritance.** C++ supports both single and multiple inheritance. This is an extremely important benefit of C++ and is used extensively in the Metroring simulation, for example in the derivation of a Metroring SEN, a Metroring frame and the random statistical distributions, to name but a few.
2. **Polymorphism.** Polymorphism provides a generic interface defined by a base class so that objects can be manipulated uniformly though they may be instances of either the base class or any derived class (Jordon, 1990). An example of polymorphism is found in the `Network` class of the Metroring Simulator. In this class all nodes are stored in an array. The simulation actually allows two types of Metroring node: *ring* nodes and Segment End Nodes. These nodes are defined slightly differently, however, polymorphism allows them to be handled in the same manner. This feature presents tremendous flexibility, and, if properly utilised, could enable the simulation of different protocols on the Metroring simulator.

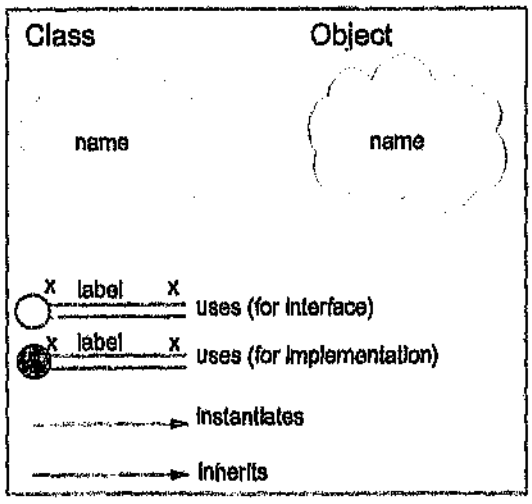



Figure D.1: Class and Object Diagrams

(1991) explains, an object diagram "may thus represent a time-lapse snapshot in time of an otherwise transitory event."

An object in an object diagram denotes some instance of a class. Therefore the operations used in an object diagram must have corresponding definitions in the class diagram. For example, if a diagram depicts object O sending a message M to a second object S, the operation M must be defined in S's class. Class diagrams document the key abstractions in a system, and object diagrams highlight the important interactions among a collaboration of instances (Booch, 1992).

The icon used to represent an object is also shown in Figure D.1. The object icon is similar to the icon for a class except that solid lines are used to denote an object. A solid line is used to represent a path through which one object can send messages to another. An arrow on the line indicates the direction of the message.

## D.4 The Booch Notation

### D.4.1 Classes

In the Booch notation, the existence of classes and the relationships between these classes is shown by a class diagram. Figure D.1 shows a class icon (the symbol used to represent a class), as well as the possible relationships that may exist between classes. A number of class diagrams may be needed to express the entire structure of a system.

The notation supports a variety of relationships which may exist between classes, including inheritance, using, instantiation, and metaclass relationships (the metaclass relationship will be ignored as it is not supported by C++). The icon for each relationship in Figure D.1 may include a label.

The using relationship is indicated by a double line with a circle at one end. A closed circle indicates that the implementation of one class uses the resources of another class. An open circle indicates that the interface of one class uses the resources of another class. The cardinality between classes which use one another may also be expressed in Booch's notation. For example, if we have two classes, A and B, a symbol 1 next to A and a symbol 2 next to B indicates that for every instance of A we have 2 instances of B, and for every instance of B we have exactly 1 instance of A.

Instantiation is indicated by a dashed line and inheritance by a solid line. The arrow points to the class being instantiated or inherited.

### D.4.2 Objects

An **object diagram** represents the existence of objects and the relationships between them. As in the case of class diagrams, a number of these diagrams may be necessary in the design of a system. Whereas classes are largely static in nature, objects are more transitory. Thus object diagrams are used to capture the dynamic semantics of operations (Booch, 1991). As Grady Booch

## D.2 Encapsulation

Encapsulation is the process of hiding all of the details of an object that do not contribute to its essential characteristics (Booch, 1991). To use an example from the Metroring simulator to illustrate this concept, consider a `Network` object. A user of the object may, for example, add node objects to the network, however, the manner in which these nodes are added and the structure in which they are stored are invisible to the user because of encapsulation.

One can see how encapsulation can allow the change of a class' implementation without affecting that class' clients. For example, the array of nodes can be changed to a linked list of nodes in the case of the `Network` class without affecting any of `Network`'s client classes.

C++ is particularly flexible when it comes to the visibility of member objects and functions. C++ makes use of what is known as intelligent encapsulation. The disruption upon a program is also kept to a minimum when algorithms which are prone to change during the system life cycle are encapsulated. Hence, encapsulation played an important part in the design of the Metroring simulator. Each class may have a **public**, **protected** or **private** part. By placing a member from a user in the private section of a class we fully encapsulate that member, limiting visibility to the class itself. Commonly that data which defines the state of an object is placed in the private part of its class definition. An example of this type of visibility would be the statistics associated with each node. We only want the Node object associated with those statistics to have access to them in order to maintain their integrity. The second type of visibility is protected. This allows the class and all its subclasses visibility to the protected member. Finally, a member may be placed in the public part of a class. This member would not be encapsulated as it would be visible to all client classes.

## D.3 Modularity

Modularity is the property of a system that has been decomposed into a set of cohesive and loosely coupled modules (Booch, 1991).

# Appendix D: Object-Oriented Concepts

## D.1 Abstraction

An abstraction denotes the essential characteristics of an object that distinguishes it from all other kinds of objects and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer (Booch, 1991).

Because an abstraction focuses on the outside view of an object, it separates the behaviour of the object from its implementation (Booch, 1991). The manner in which the object behaves forms the outside view of the object, while the way in which the object is implemented is an internal detail which is hidden from the outside world. In C++ member functions typically describe the manner in which an object behaves.

The main problem in developing an object model is to define the major abstractions. Often, these abstractions directly parallel the vocabulary in the problem domain. So, in a Metroring for example, a node would form one key abstraction. This process is closely linked to the process of developing a simulation model as discussed in Chapter 6. We encapsulate all our knowledge about the abstraction into a **class**. Generally, implementation details are hidden in a **private** section of the class, while the behaviour of the class forms the **public** section of that class. Classes are really just abstractions which we have invented to describe the problem domain, but when building a model we need tangible objects upon which we may operate. We therefore create an instance of a class to form an object. If we have a class called Node, which describes the universal behaviour of any Metroring node, we might create a number of instances of this class, say *Node One*, *Node Two* and *Node Three*, to form a ring.

41. Smythe, C., "Networks and their architectures", Electronics and Communication Engineering Journal, 1991, pp. 18-28.

42. Spragins, J.D., Hammond, J L. and Pawlikowski, K., "Telecommunications Protocols and Design", Addison Wesley Publishing Company, 1991

43. Stroustrup, B., "The C++ Programming Language", Addison-Wesley, 1991

44. Sunshine, C.A., "Transport Protocols for Computer Networks", In: Stallings, W. ed., "Tutorial: Computer Communications, Architectures, Protocols, and Standards", 2nd ed., Computer Society of the IEEE, 1987, pp. 308-313.

45. Sunshine, C A., "Network Interconnection and Gateways", IEEE Journal on Selected Areas in Communications, Jan. 1990, Vol.8, No.1, pp. 4-11.

46. Sventek, J. et al, "Token Ring Local Area Networks: A Comparison of Experimental and Theorectical Performance", 1984, In: Kim, B.G. ed., "Current Advances in LANs, MANs and ISDN", Artech House, 1989, pp. 139-147

47. Tanenbaum, A.S., "Computer Networks", Prentice-Hall, 1988

48. Taylor, D., Oster, D. L. and Green, L., "VLSI Node Processor Architecture for Ethernet", In: "Advances in Local Area Networks", 1989, pp. 49-61.

49. Teja, E.R., "Router roundup: Tools for network segmentation come of age", Data Communictions, Sep. 21, 1989, pp. 35-40.

50. Tjirkalli, N., "The Definition of the Metro Ring Network Protocol", Dissertation submitted for MSc., 1993

51. Todd, T.D., and Bignell, A.M., "Traffic Processing Algorithms for the SIGnet Metropolitan Area Network", IEEE Transactions on Communications, Vol.40, No.3, March 1992, pp.568-576.

52. Venna, P.M., "Performance Estimation of Computer Communication Networks", Computer Science Press, 1989

53. Woodside, C.M., "The effect of buffering strategies on protocol execution performance", IEEE Transactions on Coummunications, 1989, pg. 545.

54. "Product Guide: A Selected List of Multiprotocol Routers and Brouters", Datamation, Oct. 1, 1992, pp. 108-111.

27. Phillips, C.I., and Cuthbert, L.G., "Concurrent Discrete Event-Driven Simulation Tools", IEEE Journal on Selected Areas in Communications, Apr. 1991, Vol.9, No.3, pp. 477-485.

28. Press, W.H., et al, "Numerical Recipes in C", Cambridge University Press, 1988, pp. 204-224.

29. Qu, Y., Landweber, H., and Livny, M., "Parallelring: A Token Ring LAN with Concurrent Multiple Transmissions and Message Destination Removal", IEEE Transactions on Communicat' ns, April 1992, Vol.40, No.4, pp. 739-745.

30. Rodrigues, M.A., "Evaluating Performance of High-Speed Multiaccess Networks", IEEE Network Magazine, May 1990, pp. 36-41.

31. Roth, P.F., Ilyas, M., and Mouftah, H.T., "SIMULA'TION: A powerful tool for prototyping telecommunications networks", Simulation, Feb. 1992, pp. 78-81.

32. Sauer, C.H., "Simulation of Computer Communication Systems", Prentice-Hall, 1983

33. Sedgewick, "Algorithms in C++", Addison Wesl y, 1992

34. Seila, A.F., "SIMTOOLS: A software tool kit for discrete event simulation in Pascal", Simulation, March 1988, pp. 93-99.

35. Schoemaker, S., "Computer Networks and Simulation", North-Holland, 1978

36. Shaikh, M.A. and Althouse, E. L., "Efficient Simulation Experiments for Comparing Communication Network Algorithms", Simulation, 1989, pp. 233-236.

37. Shankar, A.U., et al, "Performance Comparison of Routing Protocols under Dynamic and Static File Transfer Connections", Computer Communication Review, vol. 22, no. 5, Oct. 1992, pp. 39-52.

38. Shanmugan, K.S., Frost, V.S. and LaRue, W., "A Block-Oriented Network Simulator (BONeS)TM", Simulation, Feb. 1992, pp. 83-94.

39. Shearn, D.C.S., "PASSIM - A Pascal discrete event simulation program generator", Simulation, July 1990, pp. 31-39.

40. Shen, S., et al, "Simulation and Theoretical Results on Cluster Management and Directory Management in Dynamic Hierarchical Networks", IEEE Transaction on Communications, Feb. 1992, Vol.40, No.2, pp. 312-324.

12. Eldredge, D.L., McGregor, J.D., and Summers, M.K., "Applying the object-oriented paradigm to discrete event simulations using the C++ language", Simulation, Feb. 1990, pp. 83-91.

13. Frost, V.S., et al, "A tool for local area network modeling and analysis", Simulation, Nov. 1990, pp. 283-297,

14. Gerla, M., Kleinrock, L., "Congestion Control in Interconnected LANs", IEEE Network, Jan. 1988, Vol.2, No.1, pp. 72-85.

15. Graybeal, W. and Pooch, U. W., "Simulation: Principles and Methods", Winthrop Publishers, 1980

16. Halsall, F., "Data Communications, Computer Networks and OSI", Addison-Wesley, 1988

17. Halsall, F., "Data Communications, Computer Networks and Open Systems", 3rd ed., Addison-Wesley, 1992, pp. 354-369

18. Hammond, J.L. and O'Reilly, P.J.P., "Performance Analysis of Local Computer Networks", Addison-Wesley, 1986, pp. 144-159.

19. Harris, N., "Course Notes Switching and Data", University of the Witwatersrand, 1990, pp.

20. Jordan, D., "Implementation Benefits of C++ Language Mechanisms", Communications of the ACM, Sep. 1990, vol. 33, no. 9, pp. 62-64.

21. Kettenis, D.K., "COSMOS: A simulation language for continuous, discrete and combined models", Simulation, Jan. 1992, pp. 33-38.

22. Klessig, R.W., "Overview of Metropolitan Area Networks", IEEE Communications, Jan. 1986, vol. 24, no. 1, pp. 9-15.

23. Law, A.M., "Simulation Modeling and Analysis", McGraw-Hill, 1982

24. Lippman, S.B., "C++ Primer", 2nd ed., Addison-Wesley, 1992.

25. Mollamustafaoglu, L, Gurkan, G., and Ozge, A.Y., "Object-Oriented Design of Output Analysis Tools for Simulation Languages", Simulation, Jan. 1993, pp. 6-14.

26. Pflug, G.C., and Proinaska, M., "The entity - connection approach to modelling and simulation", Simulation, Oct. 1990, pp. 226-235.

# References


1. "A Management Briefing on Extending the Company Network", General DataComm, 1993
2. Banks, J. and Carson, J.S., "Discrete-Event Syste n Simulatio:.", Prentice-Hall, 1984
3. Boggs, D.R., Mogul, J.C and Kent, C.A., "Measured Capacity of an Ethernet: Myths and Reality", 1989, In: Kim, B.G. ed., "Current Advances in LANs, MANs and ISDN", Artech House, 1989, pp. 91-103.
4. Booch, G., "Object Oriented Design with Applications", The Benjamin/Cummings Publishing Co., 1991
5. Booch, G., "The Booch Method: Notation", Computer Language, Oct. 1992, vol. 9, no. 10, pp. 40-54.
6. Bradner, S., "Cisco Systems Addendum to Router Tests V.4", Cisco Systems, 1991
7. Burmeister, N.J. and Harris, N., "LAN Protocol Overheads and Performance Assessment", Elektron, 1992, pp. 27-29.
8. Chiarawongse, J. et al, "Performance Analysis of a Large Interconnected Network by Decomposition Techniques", IEEE Network, July 1988, Vol.2, No.4, pp. 19-27.
9. Cidon, I. and Ofek, Y., "MetaRing - A Full-Duplex Ring with Fairness and Spatial Reuse", IEEE Transactions on Communications, Jan. 1993, vol. 41, no. 1, pp. 110-120.
10. Cobb, R., Mansfield, E.R., and Mellichamp, J.M., "Development of Design Guidelines for Local Area CSMA/CD Networks", Simulation, Apr. 1992, pp. 270-276.
11. Cooper, R.B., "Introduction to Queueing Theory", Second Edition, Elsevier North Holland, 1981

COMNET end-to-end delay results

| 3 Rings | | 4 Rings | | 5 Rings | |
|---|---|---|---|---|---|
| Rate (fps) | Delay (sec) | Rate (fps) | Delay (sec) | Rate (fps) | Delay (sec) |
| 1950 | 0.01 | 1950 | 0.01 | 1950 | 0.01 |
| 17890 | 0.01 | 17888 | 0.01 | 17887 | 0.01 |
| 35912 | 0.01 | 35913 | 0.01 | 35912 | 0.01 |
| 54088 | 0.02 | 54082 | 0.02 | 54075 | 0.02 |
| 63119 | 0.04 | 63127 | 0.03 | 63124 | 0.04 |
| 67788 | 0.28 | 68020 | 0.23 | 67575 | 0.38 |
| 70429 | 0.61 | 70779 | 0.55 | 69753 | 0.9 |
| 76584 | 1.36 | 77278 | 1.18 | 74731 | 1.99 |
| 82748 | 1.85 | 83850 | 1.6 | 79916 | 2.74 |
| 95407 | 2.71 | 94286 | 2.97 | 87993 | 4.29 |
| 101080 | 4.54 | 100333 | 4.85 | 2444 | 6.09 |

# Appendix G: Results

Results are produced in graphical form in Chapter 7. Data from two sets of these graphs have been reproduced in tabular form to enable easier comparison.

Theoretical vs simulated results for a single Metroring node

| Traffic Intensity | Queue Length (Analytical) | Queue Length (Simulated) | Mean Queueing Time (Analytical) | Mean Queueing Time (Simulated) |
|---|---|---|---|---|
| 0.1 | 0.01 | 0.010386 | 0 | 0.00027 |
| 0.2 | 0.03 | 0.045812 | 0.000219 | 0.000628 |
| 0.3 | 0.08 | 0.111481 | 0.000374 | 0.001066 |
| 0.4 | 0.16 | 0.22458 | 0.000574 | 0.001634 |
| 0.5 | 0.28 | 0.367631 | 0.00085 | 0.002136 |
| 0.6 | 0.6 | 0.610135 | 0.001442 | 0.002926 |
| 0.7 | 1.23 | 1.228807 | 0.002513 | 0.004954 |
| 0.8 | 2.4 | 2.006718 | 0.00442 | 0.007246 |
| 0.9 | 17.23 | 16.700741 | 0.027674 | 0.053064 |