

172 1122

# **SIMULATOR FOR INTERCONNECTED LAN'S**

*Dissertation submitted to The Jawaharlal Nehru University  
in partial fulfilment of the requirements  
for the award of the degree of*  
**MASTER OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE**

**B VENKATA RAMANA**

**SCHOOL OF COMPUTER AND SYSTEMS SCIENCES  
JAWAHARLAL NEHRU UNIVERSITY  
NEW DELHI-110 067  
JANUARY 1994**

# CERTIFICATE

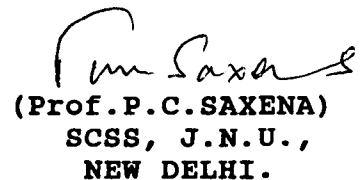
This is to certify that dissertation entitled " **SIMULATOR FOR INTERCONNECTED LAN'S** ", being submitted by B.VENKATA RAMANA to **Jawaharlal Nehru University** in the partial fulfillment of the requirement for the award of degree of **Master of Technology** in Computer science is a record of original work done by him under the supervision of Prof.P.C.SAXENA, Professor, School of computer and systems sciences, Jawaharlal Nehru University during the year 1993 monsoon semester.

The results reported in this dissertation have not been submitted in part or full to any other University or Institute for the award of any degree or diploma.



Handwritten signature of Prof. K.K. Bharadwaj, dated 4-1-94.

(Prof.K.K.BHARADWAJ)  
Dean, SCSS, J.N.U.,  
NEW DELHI.



Handwritten signature of Prof. P.C. Saxena.

(Prof.P.C.SAXENA)  
SCSS, J.N.U.,  
NEW DELHI.

*to*

*my parents*

## ACKNOWLEDGEMENTS

I feel pleasure to express my heartfelt gratitude to my guide **Prof P.C. SAXENA** for his uncompromising guidance, constant supervision and constructional help. This effort would not have succeeded without the valuable discussion, encouragement to pursue my thoughts in my own way, apt criticism and excellent guidance.

I extend my sincere thanks to Dean **Prof. K.K. BHARADWAJ** , for providing me the opportunity to undertake this project. I would like to thank the staff and authorities of our school for providing me the necessary facilities to complete my project.

I would like to thank some of my friends, Rama Rao, Manoj, Ravi, Govardhan, Sanjay, who helped me in completing this project successfully. Last but not the least , I would like to thank my classmates, for the encouragement they gave me in completing this project.

*Ramana*  
( **B. VENKATA RAMANA** )

# TABLE OF CONTENTS

## CHAPTER 1 : INTRODUCTION

1.1	Local Area Networks	1
1.2	Some Media Access Protocols in LANs	2
1.3	Interconnected LANs	7
1.4	Performance: Measures, Analysis and Simulation	8
1.5	Available Analytical Results	10
1.6	Motivation	11
1.7	Design Objectives	12

## CHAPTER 2 : OVERVIEW OF DISCRETE EVENT SIMULATION

2.1	Definitions	16
2.2	Discrete Event Modelling Approaches	16
2.3	Time Flow Mechanism	18
2.4	Special Purpose Simulation Languages	19
2.5	Model Validation	20
2.6	Stochastic Variate Generation	21
2.7	Simulation Output Analysis	23
2.8	Variance Reduction Techniques	26
	Appendix: Methods of Output Data Collection	30

### **CHAPTER 3 : MODELLING AND PROGRAMMING**

3.1	Modelling Assumptions	33
3.2	General Modelling Approach	34
3.3	Modelling of The MAC Protocols	38
3.4	Modelling of The Bridges/Routers	47
	Appendix: Programming Details	49

### **CHAPTER 4 : EXPERIENCE WITH RUNNING THE SIMULATOR**

4.1	Individual Network Simulations	63
4.2	Effects of Initial Transients	66
4.3	Simulation of The Interconnected Networks	66
4.4	CPU Time Requirements	68
	Appendix	83

### **CHAPTER 5 : CONCLUSIONS**

	<b>REFERENCE</b>	<b>88</b>
--	------------------	-----------

## INDEX OF FIGURES

1)	Fig.1.1	Ring Topology	13
2)	Fig.1.2	Bus Topology	14
3)	Fig.1.3	Ring/Bus Architecture	14
4)	Fig.2.1	Event Process and Activity	29
5)	Fig.4.1	Performance Comparison of Token Ring and Ring/Bus	70
6)	Fig.4.2	Token Ring Performance Under Different Service Disciplines	71
7)	Fig.4.3	IEEE 802.5 Token Ring Performance	73
8)	Fig.4.4	Performance of Ring/Bus with Two Priority Classes	74
9)	Fig.4.5	Performance of Ring/Bus with Eight Priority Classes	75
10)	Fig.4.6	Performance Comparison of IEEE 802.5 Token Ring and Ring/Bus with Priority	76
11)	Fig.4.7	Performance of CSMA/CD	77
12)	Fig.4.8	Confidence Intervals For CSMA/CD	78
13)	Fig.4.9	Three Interconnected Networks - Case 1	80
14)	Fig.4.10	Three interconnected Networks - Case 2	81

## INDEX OF TABLES

1)	Table 1.1	IEEE 802.3 Standards	15
2)	Table 4.1	Delay Performance of the Token Ring and the Ring/Bus	69
3)	Table 4.2	IEEE 802.5 Token Ring Simulation Results	72
4)	Table 4.3	Effects of Transients in Token Ring	79
5)	Table 4.4	Effects of Transients in CSMA/CD	79
6)	Table 4.5	Three Interconnected Networks- Case1A	80
7)	Table 4.6	Three Interconnected Networks- Case1B	80
8)	Table 4.7	Three Interconnected Networks- Case 2	81
9)	Table 4.8	CPU Time Requirements on VAX/VMS System	82



A general purpose simulator SIMNET for the performance study of individual and interconnected networks has been developed. The event scheduling approach has been used to model various MAC protocols. The simulator software has been written in C . The simulator allows the characteristics of every user (in the network ) to be specified in an asymmetric network. For a symmetric network, where all users are identical, the characteristics of only one user need to be specified. Individual nodes may be placed at any desired location in the network or they may be placed at uniformly distributed points on the communication medium.

SIMNET supports the Token Ring (simplified version as well as the IEEE 802.5 standards, with or without priority), the CSMA/CD (ETHERNET, IEEE 802.3 standards) and the Ring/Bus protocol (an improved version to the Token Ring method) with/without priorities. Individual networks may be specified to have any of the above media access protocols. Various parameters of the access method may be separately specified or the appropriate IEEE 802.3/802.5 standards may be used for the simulation runs. Simulation of individual networks may be run with a limit of 100 nodes per network.

SIMNET allows the simulation of interconnected networks here upto 10 component networks - each with upto 100 users - may be separately specified. The component networks may be specified

with any of the access protocols mentioned above. This may be done in complete detail as for the individual networks mentioned above. Any pattern of interconnections between the component networks may be simulated. Traffic components flowing from individual one network to another may be separately specified.

SIMNET provides complete statistics on the mean packet delays and maximum buffer size requirements at each node. Statistics averaged over all the nodes in each network is also provided. The overall delay throughput behaviour of the network may also be studied. For interconnected networks, SIMNET provides details on the behaviour of the bridge/router between individual networks, i.e. its buffer requirements and the delay going through the bridge/router; this information is provided along with the details of the behaviour of each component network and that of each node on these networks.

# CHAPTER 1

## INTRODUCTION

Local area networks (LANs) are becoming an integral part of many working environments. As the number of computers in offices, universities, factories etc. has increased, the need to interconnect them has also increased rapidly. The networking of computers allows sharing of information and expensive resources among the users.

### 1.1 Local Area Networks

A local area network is a geographically confined communication system. The participating nodes share a common transmission medium. Access to the medium is provided by a distributed protocol called the Medium Access Control (MAC) protocol.

The pattern of interconnection used among the various nodes of the network is termed the network topology. The ring and the bus are the two common topologies used by LANs. In the ring topology, a packet generated by a station is passed from node to node along unidirectional links so that it passes by its destination and eventually returns to the source node. In the bus topology, packets flow away from the originating node in both

directions to the ends of the bus. The destination node reads the packet as it passes by. The network topology, performance, cost, reliability and communication medium are the factors which dictate the choice of an access protocol.

The Token Ring and the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) are the two commonly used access protocols for the ring and bus topologies respectively. These two access protocols and their respective topologies offer different advantages and disadvantages. In the following section, we briefly review Token Ring and CSMA/CD protocols along with their IEEE 802 recommendations. A new hybrid Ring/Bus protocol for networks with the ring topology is also reviewed. These are the MAC protocols, whose modelling and performance simulation are the objectives of the SIMNET simulator which has been developed.

## **1.2 Some Media Access Protocols in LANs**

### *1.2.1 Token Ring*

In a Token Ring, the nodes are placed on a ring network as shown in Fig.1.1. The right of access to the network is regulated by a special bit pattern called the *token*. The token keeps circulating around the ring whenever all the nodes are idle. When a node wants to transmit its packet(s) it waits for the token to arrive, converts the token to a connector (busy token), sends its packet(s) and then forwards the token to the next downstream node. Each node introduces at least a one bit

delay which allows the node to read and regenerate the incoming bit before sending it on to the next node. Logically, the token ring behaves like a polling system [1].

Many variations exist in the detailed operation of the token ring. Each node could be restricted to sending a limited number of packets (Limited Service) or a node could empty its queue of waiting packets before releasing the token to the next node (Exhaustive service). A transmitting node may release the token just after transmission of its packet(s) or may wait for the packet to return around the ring.

#### IEEE 802.5 Token Ring Standards [2]

The 802.5 Token Ring protocol standards extend the basic token ring scheme to incorporate an elaborate eight level priority service. The 802.5 Token Ring operates at a transmission speed of 4 Mb/s. A 24-bit token is used instead of normal 8-bit token. A one byte access control (AC) field within the token and the header of every transmitted packet contains three priority bits (PPP) and three reservation bits (RRR). When a node wants to transmit a priority  $n$  packet, it must wait until it can capture a token whose priority is less than or equal to  $n$ . A station may hold the token for the Token Holding Time (THT) which is 10 msec unless the installation specifies a different value. After a node has claimed the token, the node transmits packets that are at or above the present ring service priority level. This continues until it has completed

transmission of all such packets or when the transmission of another packet would exceed THT. If the node has not received back the header of last transmitted packet, it must wait for that before releasing the token.

A feature called *stacking* is used to control the distributed raising and lowering of the ring priority, based on the priority and reservation bits. A stacking station is one which has raised the priority of the ring, and it uses two stacks to remember both the former priority of the ring (receive stack) and the new priority (transmit stack). The new priority needs to be remembered because another station may raise the ring priority even higher. The receive stack is used at some later time to restore the ring priority to its former (lower) priority; more than one station at a time may be the stacking stations.

### 1.2.2 CSMA/CD

A CSMA/CD network is organized as a bus as shown in Fig.1.2. Under the CSMA/CD protocol, every node wanting to transmit a packet must listen to the transmission medium in order to find out whether any transmission is in progress. If the medium is busy, the node must wait. In spite of carrier sensing, packet collisions cannot be completely avoided because of the non-zero propagation delay of the bus. Upon detection of a collision, transmission is aborted and the node transmits a brief jamming signal to ensure that all stations know that there has

been a collision. The colliding transmissions are then rescheduled for a random time in future, at which point they attempt to transmit again.

Several CSMA/CD variants exist for the way they schedule transmission of packets arrived during channel busy period. In the *non-persistent* scheme, ready nodes sensing the channel busy wait a random time before trying again. In the *1-persistent* scheme, nodes are allowed to transmit as soon as the channel is sensed idle. In the *p-persistent* scheme, ready nodes sensing the channel busy wait for a geometrically distributed time (beyond the point where channel is sensed idle) before trying again.

### IEEE 802.3 CSMA/CD Standards [3]

Here, 1-persistent CSMA/CD is chosen as the access protocol. The parameter values for a 10 Mb/s baseband implementation are given in the Table 1.1. The scheduling of retransmissions is determined by a controlled randomization process called the *truncated binary exponential backoff*. At the end of enforcing a collision, i.e., transmitting a jamming signal, the retransmission is delayed by an integral multiple of the worst-case round trip propagation delay called a "slot time". The number of slot times to delay before the  $n^{\text{th}}$  retransmission attempt is chosen a uniformly distributed random integer  $r$  in the range  $0 \leq r < 2^k$ , where  $k = \min(n, 10)$ . If the number of

retransmissions exceeds some pre-specified upper limit, the packet is discarded.

### *1.2.3 Ring/Bus*

This new scheme for the ring networks operates in a manner very similar to the token ring system. Physically, the Ring/Bus is a linear or tree-shaped cable onto which the stations are attached. Logically, the stations are organized into a ring, shown in Fig. 1.3., with each station knowing the address of the station to its "left" and "right". When the logical ring is initialized, the highest numbered station may send the first frame. After it is done, it passes permission to its immediate neighbour by sending the neighbour a special control frame called a *token*. The total time for a signal to go around the ring is called the Ring Latency Time (RLT). The RLT of the ring network employing this scheme consists only of the propagation delay around the ring. All the nodes other than the transmitting node passively monitor the transmission medium. To maintain token circulation when none of the nodes have any packet to send, an active repeater with a 1-bit delay must be placed somewhere in the ring from considerations of signal loss in propagation. With this, the ring may also be viewed as a bus with two sections connected through a 1-bit delay, hence the name Ring/Bus [4].

The basic Ring/Bus approach has been extended to support multi-priority traffic [5]. For  $n$  priority classes of service, a  $[8+(n-1)]$  bit token is used. The  $[n-1]$  lower significant bits of the busy token indicate the priority of the



on going transmission. This scheme allows pre-emption of lower priority transmissions of other nodes by a node having a high priority packet. Therefore, a node waiting to transmit a priority  $n$  packet will have to wait till an idle token or busy token of priority less than  $n$  is received. A node will know whether its transmission has been pre-empted or not, when it receives back the busy token. If the received busy token has the same priority as that of the transmitted packet, the node can continue the transmission of the remaining part of the packet otherwise it stops its transmission and switches to a delayed-PR mode. In the delayed-PR mode, the node introduces a latency equal to the length of the token. It will remain in the delayed-PR mode until it succeeds in transmitting a packet without pre-emption.

### **1.3 Interconnectd LANs**

Many situations exist in which a single organization may be having multiple LANs. Ownership autonomy, cost, traffic management, long physical distance between the nodes reliability may be some of the reasons for operating different LANs. In itself, we have two CSMA/CD networks plus three token ring networks. One CSMA/CD network is owned by Computer Centre and other by ERNET project group. This is an example of situation where ownership autonomy considerations brought multiple LANs into existence.

The reasons behind the need to network the computers can logically be extended to establish the need of connecting different LANs. To interconnect different LANs, a node must convert packets from one protocol to another. This node is generally called a *gateway*. In particular a gateway connecting two networks at the network layer is called a *router*, while a gateway which interconnects networks working under identical MAC layers is referred to as a *bridge*. Many problems arise in connecting two LANs. Differences in MAC protocol, packet format, maximum packet size and transmission speed are the factors which should be considered while designing a bridge/router [6].

#### **1.4 Performance: Measure, Analysis and Simulation**

The most common performance measures used in LANs are the average throughput,  $S$  and mean packet transfer delay,  $D$ .  $S$  is defined as the limit as  $T \rightarrow \infty$  of the average proportion of time between 0 and  $T$  that the channel uses to successfully carry packets. The maximum achievable throughput is called the *capacity*. The mean packet delay  $D$  of a packet is defined as the time interval between the generation of a packet at a node and the start of its successful transmission from that node. This means that the packet delay consists of the queuing and access delay at the sending node. For the internet packets, packet delay is defined as the time interval between the generation of a packet at a node in the source network and the start of its successful transmission in the destination network. Therefore, the packet delay in this case also includes propagation and

transmission delays in all the networks of the path except that in the destination network. Buffer requirements, buffer overflow probabilities (particularly at the bridges/routers) etc. may be other performance measures of interest.

In order to design LANs and their interconnections efficiently, it is necessary to have good performance evaluation techniques. With adequate performance modelling, it is possible to anticipate and correct inadequate network performance while avoiding cost of altering an existing network. Performance of the network can be evaluated using three different approaches. One is to actually measure the performance of an operating network. Another possibility is to use analytical calculations and third is to simulate the network using a computer model. The first approach is not very attractive as performance can be evaluated only after network has been set up. The analytical approach requires less effort and time to calculate performance measures. However, it makes various simplifying assumptions in order to allow mathematical analysis. The analytical models often assume a symmetric network i.e., a network in which each node has the same packet arrival rate, same mean packet length and distribution. Sometimes they also restrict the model to have a particular placement of the nodes. The performance measured under these simplifying assumptions may not truly reflect the performance of the actual network. The computer simulation approach allows modelling of networks with desired degree of complexity ; hence, more exact performance results can be

obtained. The major disadvantages of simulation are the time required to write, debug, and execute a sophisticated simulation, the efforts necessary to validate the simulation, and the analysis required of the statistical simulation output [7].

## 1.5 Available Analytical Results

In this section, we briefly review the analytical results available for performance evaluation of the MAC protocols discussed in Sec 1.2 and for interconnected LANs.

In the Token Ring case exact results for an Asymmetric token ring with exhaustive service are available [8], [9]. Exact results of other service disciplines are available only for the symmetric case. An approximate performance analysis of the 802.5 token ring priority access mechanism is given in [10]. It makes the assumption that the node issuing a token knows the level of the highest priority packet waiting for transmission anywhere in the system. This is not the case in the real system. The lowering of ring priority is also not accurately modeled.

For the 1-persistent CSMA/CD protocol only approximate results are available [11], [12]. Most of the analytical models assume a slotted CSMA/CD where all the nodes are synchronized into slots of *Slot Time*. Here, if only one node transmits in a slot Time, all the other nodes will detect this transmission and will not use subsequent slots until entire packet is completed. They also assume the buffer size to be unity i.e., new arrivals

are discarded until the previous arrival is successfully transmitted.

For the Ring/Bus protocol without priority, results for the token ring can easily be extended. However, for the Ring/Bus with Priority Scheme only approximate results for packet delays for a symmetric network are available [5]. The Analytical model makes the pessimistic assumption that any node which has one or more low priority packets will be in the delayed-PR mode.

Several performance studies on interconnected systems have been published in literature for token ring networks [13] and for CSMA/CD networks [14]. However, these results are approximate. The basic problem in analyzing interconnected systems is that of characterizing the output process of a multiuser random access communication system i.e., the departure process of successfully transmitted packets. The output process of a multiuser random access communication system depends on the protocol that has been deployed. Description of that process is a difficult task, and only approximations based on special assumptions have been attempted [15].

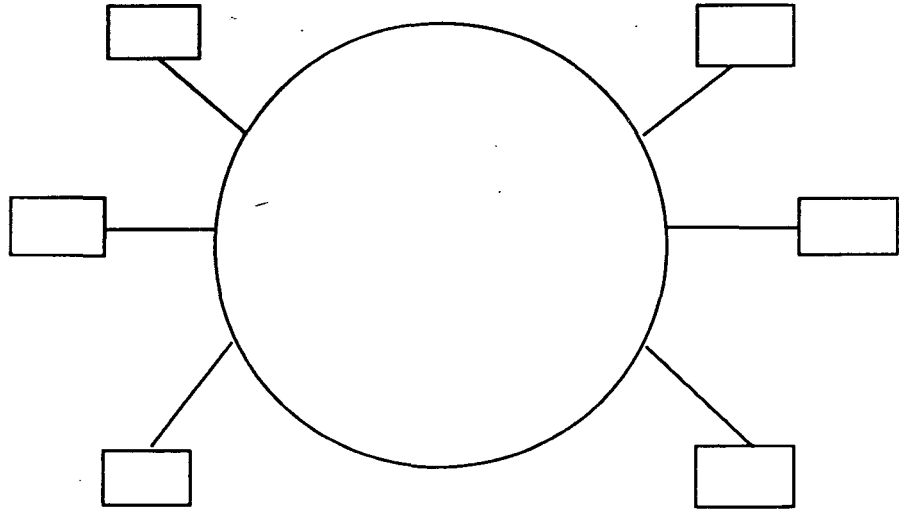
## **1.6 Motivation**

The unavailability of exact analytical results for the performance evaluation of real-life Local Area Networking situations motivated me to use a simulation approach. I developed a simulator SIMNET using which the throughput-delay

performance of isolated as well as interconnected LANs can be evaluated for a number of MAC protocols.

## **1.7 Design Obejctives**

The simulator should be a generalized one in the sense that system details can be specified at the node level. For interconnected LANs, no restriction should be placed on the choice of the MAC protocol of the component LANs and the way LANs, are interconnected. Simulation outputs should have reasonable statistical validity. CPU Time requirements should be as small as possible. User-interface should be as user friendly and simple as possible.



**Fig.1.1 : Ring Topology**

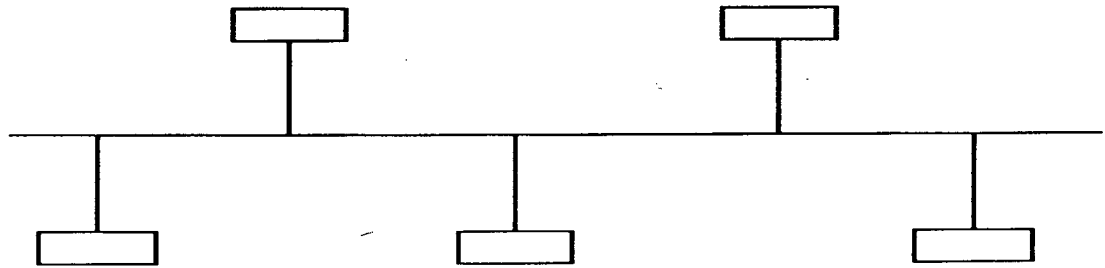


Fig 1.2 : Bus Topology

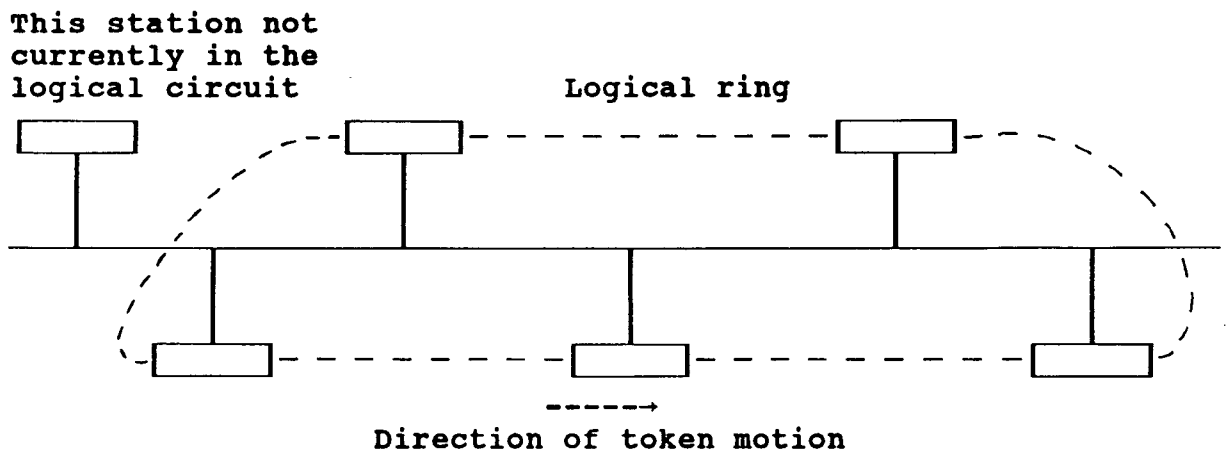


Fig 1.3 : Ring/Bus Topology



---

Parameters	Values
Slot Time	512 bit times
Inter Frame Gap	9.6 $\mu$ s
Attempt Limit	16
Backoff Limit	10
Jam Size	32 bits
Max Frame Size	1518 bytes
Min Frame Size	512 bits
Address Size	48 bits

---

Table 1.1 IEEE 802.3 Standards

# CHAPTER 2

## OVERVIEW OF DISCRETE EVENT SIMULATION

### 2.1 Defintions

A *system* is a collection of entities which operate in some interrelated manner. Each entity is characterized by its attributes. The vector representing the values of the attributes of the system entities is called *state* of the system. An event denotes a change in the state of a system entity. If changes in the states of the system entities occur at discrete moments in time, the sytem is called a *discrete event system*.

### 2.2. Discrete Event Modelling Approaches [16].

The concepts of event, process, and activitiy are important when building a model of a system. As already defined, an event signifies a change in the state of an entity. A process is a sequence of events ordered on time. An activity is a collection of operations that transform the state of an entity as shown in Fig.2.1. These three concepts give rise to three alternative ways of building discrete event models, namely the event scheduling approach, the activity scanning apporoach and the process interaction approach.

In the event scheduling approach, events are scheduled in advance. Whenever an event is scheduled, a record identifying the event and the time at which it is to occur is filed in a special list. The simulation progresses as follows. The list of scheduled events is searched to find the event with the earliest scheduled time. The simulation time is advanced to this scheduled time, state changes and scheduling of new event(s) associated with the occurrence of this event are performed. After that, the list is searched again for the next event and the cycle is repeated.

The activity scanning approach calls for the modelling of the activities. Events are not scheduled in this approach. Each system entity that changes state has a clock. The simulation progresses as follows. Clocks of all the system entities are checked to determine the time and type of the next event, the simulation time is advanced to this time, the steps necessary with this event are performed and then the cycle is repeated. In effect, the activity scanning approach substitutes logical tests in the model for the "event scheduling and next event selection features" of the event scheduling approach. The essential feature of this approach becomes evident when there are several activities. Then, whenever time is advanced to the next event, all activities are scanned to determine which can be begun or ended.

The process interaction approach combines the event scheduling and activity scanning approaches. It maintains a list of future events but it also carries out, at each event time

a scan of all the activities and finds out which one can be begun or ended as in the activity scanning approach. The process interaction approach to modelling stresses the interaction between processes in describing a system. Rather than modelling changes in the system state, this approach describes the progress of entities, referred to as transactions, through the system. It could be noted that the selection of a particular modelling technique depends on the system being modeled.

### **2.3 Time Flow Mechanism**

All the three modelling approaches require that a means must be provided for causing the events in the simulation to occur at proper time and in proper sequence. This is often referred to as the Time Flow Mechanism (TFM) of the simulation. The modelling approaches which use event scheduling, maintain a doubly linked linear list of events. Events are ordered on scheduled time of occurrence. A doubly linked list consists of data and pointer to both the next item and preceding item of the list. Using a linked list, selection of the next event is quite simple since it is the event at the top of the list. The scheduling of new events can however be time consuming because the list must be searched to find the proper location for the event in order to maintain the linear order.

The algorithm chosen to insert a new event in the list greatly affects the simulation's execution time. Many algorithms

exist for event insertion in the list. One algorithm begins the search for proper location from the bottom (high time values) of the list, another algorithm starts from the top (low time values) of the list. A third algorithm uses an additional pointer. By maintaining a pointer to the middle event of the list, it is possible to determine whether the new event could be in the top or bottom half of the list. However, once this is determined, the question of the direction to scan that half arises leading to four different algorithms. Use of middle pointer expedites insertion of events in a long list. A comparative evaluation of these algorithms along with some other algorithms can be found in [17]. It has been found that the middle pointer method which begins all insertions from the top of the half in which the new event belongs, gives an overall good performance.

## **2.4 Special Purpose Simulation Language**

The three modelling approaches described above, have led to the development of three types of special purpose simulation languages. For example, SIMSCRIPT uses event scheduling approach, CSL uses activity scanning approach, and GPSS uses process interaction approach. These special purpose simulation languages offer many convenient facilities such as automatic generation of pseudo-random numbers for any desired statistical distribution; automatic data collections; their statistical analysis and report generation; automatic handling of queues;

etc... However, use of these languages has not become as popular as one would expect from the advantages that these languages offer. The three major reasons are (1) That these languages are not usually available (2) Users generally not familiar with these languages, (3) And the fact that these languages are highly problem oriented. On the other hand general-purpose high-level languages such as C have the advantages of easy availability, user familiarity, portability, better efficiency in terms of execution time, and the freedom of choosing the modelling approach. Therefore, general-purpose high-level languages have been and are being used to simulate discrete event systems.

## **2.5 Model Validation**

Model validation is an important stage in simulation experiment before simulation results can be accepted. The objective of the validation stage is to ensure that the simulation model is a proper representation of the system being studied. What is meant by the validation of a model, is a difficult question. Since no simulation model will duplicate the given system in every detail, therefore it cannot be said to be a true model of the system. However, simulation studies are usually done with a model which represents the real system adequately for the purpose of the study for which it is used.

The validation efforts can be grouped into two parts; validation of the abstract model itself and the validation of its implementation. The validation of the abstract model is highly

subjective. Testing the validation of an implementation is a more objective and easier task. The first step in validating an implementation is to check the programming logic. It ensures that the model has been correctly implemented. The next step of validating the implementation is the use of sample problems. The sample inputs should be chosen such that corresponding outputs can be obtained analytically. If results match, then the model is a valid model [18].

## 2.6 Stochastic Variate Generation

To simulate (discrete) stochastic systems such as queueing networks, a source generating random variables is required. Random variables for a wide variety of probability distributions can be obtained, provided only that a sequence of uniformly distributed random variables can be generated. Therefore, the first step in preparing the input for simulation is to have a deterministic algorithm that produces uniform random variables. Such deterministically generated numbers which appears to be random are called pseudo-random numbers. Many statistical tests have been devised to test the randomness properties of the generated sequence. The primary criteria for the generated random sequences are that they be uniform and independent.

TH-5P30

### 2.6.1 Uniform Variate Generation

Numerous methods exist for generating uniform random deviates. These methods are usually based on some recurrence relation. Each new number is generated from the previous one by applying some *scrambling operation*. A fast, and the most commonly used method is the so-called *multiplicative congruential generator*. It consists of computing

$$x_i \equiv cx_{i-1} \pmod{m}, \quad i = 1, 2, \dots, m-1 \quad (2.1)$$

where  $x_i$  is the  $i^{\text{th}}$  pseudo-random number,  $x_{i-1}$  is the previous pseudo-random number, and  $c$  is a constant multiplier. The value of  $x_0$  is called the *seed* of the random number generator. The value  $m$  is usually chosen to be equal to the largest number that can be represented on the computer system. If the multiplier,  $c$ , is a primitive root modulo  $m$ , then the generator will have a maximal period of  $m-1$ .

The random number generated by this method have been tested extensively for uniformity and independence. It has been found experimentally that  $m = 2^{31}-1$  and  $c = 16807$  give much better results than almost all of the many other values that were investigated [19].

### 2.6.2 Non-Uniform Variate Generation

The uniform random number generator can be used to generate random numbers from any distribution using the following property; if  $x_i$  is a sequence of random numbers which are uniformly



distributed in the interval (0,1) and if  $y$  has the probability density function  $f(y)$  and cumulative distribution function  $F(y)$  then the sequence of random numbers  $y_i$  generated by the operation

$$y_i = F^{-1}(x_i) \quad (2.2)$$

has the density function  $f$ . For example, for an exponentially distributed sequence,

( Here in this formulas  $\delta$  indicates  $\lambda$  )

$$f(y) = (1/\delta)e^{-y/\delta}$$

and  $F(y) = 1 - e^{-y/\delta}$

Then (2.2) can be used to get,

$$y_i = -\delta \log(1-x_i)$$

or  $y_i = -\delta \log x_i \quad (2.3)$

Thus (2.3) will generate the desired sequence.

## 2.7 Simulation Output Analysis [20].

In a simulation experiment, the mean  $\mu_x$  of an analyzed process is estimated from the sequence of collected observation  $x_1, x_2, \dots, x_n$  by calculating the average as follows:

$$X(n) = \sum_{i=1}^n x_i / n \quad (2.4)$$

The sample mean  $X(n)$  calculated from (2.4) is itself a random variable. According to the *central limit theorem*, the distribution of  $X(n)$  approaches the normal distribution with variance  $\sigma_x^2/n$  and mean  $\mu_x$  as the sample size  $n$  increases, where  $\sigma_x^2$  is the variance of the analyzed process.

Since  $X(n)$  is a random variable, a measure of its intrinsic reliability or precision is required. The *confidence interval width* and the *confidence level* jointly give this measure of precision. The confidence level  $(1-\alpha)$  is defined by the probability,

$$P( X(n) - \delta_x \leq \mu_x \leq X(n) + \delta_x ) = 1 - \alpha \quad (2.5)$$

where,  $\delta_x$  is the half width of the confidence interval for the  $X(n)$ , and  $0 < \alpha < 1$ . The half-width of confidence interval is given by

$$\delta_x = Y_{1-\alpha/2} \sigma[X(n)] \quad (2.6)$$

where,

$$\sigma^2[X(n)] = 1/n(n-1) \sum_{i=1}^n (x_i - X(n))^2 \quad (2.7)$$

is the unbiased estimator of the variance of  $X(n)$  and  $Y_{1-\alpha/2}$  is the  $(1-\alpha/2)^{th}$  fractile of normal distribution with zero mean and unit variance and is defined as,

$$1/\sqrt{2\pi} \int_{-\infty}^{Y_{1-\alpha/2}} e^{-z^2/2} dz = 1 - \alpha/2 \quad (2.8)$$

Qualitatively, the confidence interval and the confidence level can be interpreted as follows. If the simulation experiment were repeated a number of times, then the interval  $(X(n) - \delta_x, X(n) + \delta_x)$  would contain the unknown population mean  $\mu_x$  in  $100(1-\alpha)\%$  of cases and would not contain that in  $100\alpha\%$  of cases.

The equation (2.7) for the estimator of the variance of  $X(n)$  has been written with the assumption that the observations are realisation of independent and identically distributed (iid)

random variables  $X_1, X_2, \dots, X_n$ . However if observation  $X_1, X_2, \dots, X_n$  cannot be regarded as realizations of iid random variables, then the estimator of the variance of  $X(n)$  is given by

$$\sigma^2[X(n)] = [R(0) + 2 \sum_{k=1}^{n-1} (1-k/n)R(k)]/n \quad (2.9)$$

where,

$$R(k) = 1/(n-k) \sum_{i=k+1}^n [x_i - X(n)][x_{i-k} - X(n)], \quad 0 \leq k \leq n-1$$

is the estimator of the autocovariance of order  $k$ .

Any variance analysis disregarding correlation among the observations would lead either to an excessively optimistic confidence interval of  $\mu_x$ , in the case of positively correlated observations or to an excessively pessimistic confidence interval for  $\mu_x$ , in the case of negatively correlated observations. Several methods of data collection and analysis have been proposed to overcome the theoretical problems that arise from the correlated nature of observations collected during the simulation. These methods either attempt to weaken or even remove statistical dependencies among observations or take actual correlation among observations into consideration. A review of these methods is given in the appendix at the end of this chapter.

Problem of Initial Transients: After initialization, any (stochastic) dynamic system passes through a transient phase before reaching the steady-state. During the transient phase, the (stochastic) characteristics of the system vary with the time. Therefore, data collected during the transient phase of

the system do not characterize the steady-state behaviour of the simulated system. The influence that the initial transient data can have on the final results is a function of the strength of the autocorrelation of collected observations. With no restriction imposed on the length of the simulation run, this influence can be arbitrarily weakened by running the simulation program sufficiently longer. If all initial output data are retained, the bias of the point estimator  $X(n)$  is greater, than if they were deleted. Contrary opinions on the usefulness of deletion are caused by the fact that it increases the variance of the point estimator, and, in effect, can increase its mean square error.

## 2.8 Variance Reduction Techniques[7]

The larger the variance of the variable being estimated more samples will be required to achieve a specified level of confidence; this can be seen from the (2.6). This corresponds to a longer simulation run. Several methods have been developed to improve the efficiency of the sampling procedure of input random variables in the simulation, and they are usually referred to as *variance-reduction techniques*. These techniques essentially exploit the statistical properties of the simulation model to reduce the uncertainty in the output. Some of the commonly used variance-reduction techniques are described here.

### 2.8.1 Antithetic Sampling

This technique works by running two simulations in parallel with random number streams that are complementary i.e., if one simulation run uses the uniform random sequence  $u_1, u_2, \dots$ , then the other simulation is made to use the sequence  $(1-u_1), (1-u_2), \dots$ . If these complementary streams produce two simulation outputs that are negatively correlated, then the average of two outputs will be a more efficient estimator for true mean than either of the individual outputs. However, it has been observed that queue interactions in a typical computer communication network are too complicated for antithetic sampling to be effective.

### 2.8.2 Control Variates

This variance-reduction technique is based on the concept of control variates. A control variate is a random variable that is correlated with the performance measure of interest and whose expected value is known. For example, let the random variable  $X$  represent the performance measure to be estimated via simulation and  $Y$  be another random variable with  $E(Y)$  known and that  $Y$  is correlated with  $X$ . Then an alternate estimate for the performance measure is  $X_C$  where,

$$X_C = X - a(Y - E[Y]) \quad (2.10)$$

The variance of the control variate estimate of  $X, X_C$  is given by

$$\text{Var}[X_C] = \text{Var}[X] + a^2\text{Var}[Y] - 2a\text{Cov}[X, Y] \quad (2.11)$$

The variance of  $X_C$  will be less than the variance of the brute force simulated value of  $X$ , if there is enough correlation between  $X$  and  $Y$  and the variance of  $Y$  is small enough. The

optimum constant "a", is chosen through a linear regression using the simulation data to minimize the variance of the estimator  $X_C$ . This technique has been found effective in queueing network performance simulations. The disadvantages of using control variates are the difficulty of finding a good control variate, and the effort required to calculate correlation from simulation data.

### *2.8.3 Importance Sampling*

The idea of importance sampling, is to use a sampling procedure which deliberately distorts the original input distribution so that a large proportion of samples is drawn from the interval of importance. Then an appropriate weighting function is used to multiply the sampled data which makes correction for the distortion introduced. A new importance sampling technique has been proposed in [7]. This technique has been developed by combining importance sampling with concepts from regenerative simulation theory. In this technique, importance sampling is accomplished by biasing (e.g. increasing) the arrival rate at the queue. Increasing the arrival rate will concentrate the simulation efforts on regenerative cycles that have a large number of entities. The regenerative cycles with the large number of entities are also the cycles that have the greatest effect on the mean waiting time. This technique has been applied to [M/M/1] and [M/G/1] queues and is found to be effective. However, extension of this technique for the simulation of queueing networks requires further research.

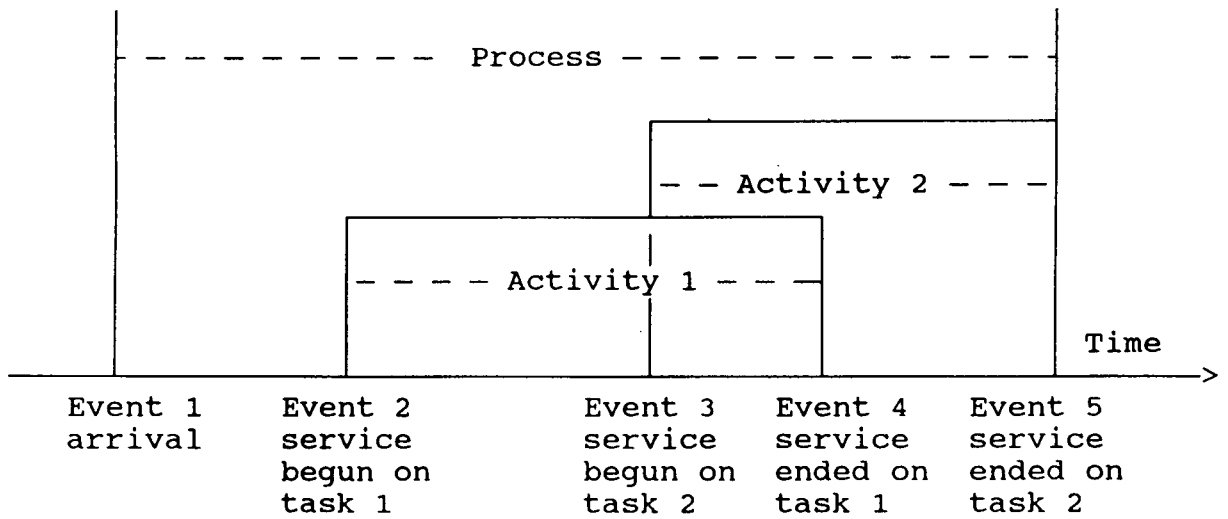


Fig 2.1 : Event , Process and Activity.

# APPENDIX

## Methods of Output Data Collection

Some of the methods of output data collection are reviewed here. A more detailed account of these methods along with some other methods may be found in [20]. The method of replications require that the simulation is repeated a number of times, each time using a different, independent sequence of random numbers. The mean value of observations collected during each run is computed. These means are used in further statistical analysis as secondary, evidently independent and identically distributed output data. There is a trade-off between the number of replications and their length for achieving a required accuracy of estimators. It has been found that it is better to keep replications longer than to make more replication. This method gives better accuracy of the point estimator measured by its mean square error. However, it is more sensitive to the nonstationarity of the observations collected during the initial transient period than methods based on single simulation run.

In the method of batch means, the recorded sequence of  $n$  original observations  $x_1, x_2, \dots, x_n$  is divided into a series of non-overlapping batches  $(x_{11}, x_{12}, \dots, x_{1m}), (x_{21}, x_{22}, \dots, x_{2m}) \dots$ , of size  $m$  and batch means  $X_1(m), X_2(m), \dots, X_k$  are used to calculate the variance of output data. This approach is based on the assumption that observations more separated in time are less correlated. Thus, for sufficiently long batches of observations,



batch means should be almost uncorrelated. Selection of a batch size that ensures uncorrelated batch means is the main problem associated with this method. Though, the problem of selecting a suitable batch size has not yet been satisfactorily solved, this method generally behaves better than the method of replication.

In the method of overlapping batch means, each new observation starts the next batch of observations. If same batch size  $m$  is used for the previous method and this method, then this method gives asymptotically more stable confidence intervals. This method uses the fact that, to generate short and stable confidence intervals an estimator of variance  $\sigma^2[X(n)]$  should have a small variance itself. It has been found that the variance of variance estimator can be reduced by introducing batches of observations. An important feature of this technique is that this makes it possible to increase the size of batches within a given length of a simulation run without decreasing the number of batches.

While the methods of batch means try to obtain less correlated data, discarding some observations between consecutive batches should be a natural and efficient way to obtain an additional decrease in the correlation between the batch means. An extreme solution is applied in the method of uncorrelated sampling in which only single observations, each of them  $k$  observations apart, are retained and all other observations are discarded. The distance between the consecutive retained observations should be selected large enough to make the correlation between them negligible. When this is done, the

sequence of  $K$  retained observations contain the realization of almost iid random variables. No results on the effectiveness of this method are available.

In the regenerative method observations are also grouped into batches, but the batches are of random lengths, determined by successive instants of time at which the simulated process starts afresh, i.e., at which its future state transitions do not depend on the past. Such instants of time are called regeneration points, and states of the process at these points are called regeneration states. The special nature of the process behaviour after each regeneration point causes batches of observations to be collected during different regeneration cycles; these would be statistically independent and identically distributed. The means of these batches would also be statistically independent and identically distributed. As a sequence of the identical distributions of output data collected within consecutive regeneration cycles, the problem of initialization vanishes if a simulation experiment commences from a selected regeneration point. This method requires special estimators because of the random length of the batches. These estimators are usually in the form of a ratio of two random variables. The ratio estimators are not very good as these are biased estimators and their asymptotic normality is questionable even for a relatively large number of samples. Two methods, one based on spectral analysis and other based on autoregressive representation, use variance estimators which actually take into account the correlation of observed sequence.

# CHAPTER 3

## MODELLING AND PROGRAMMING

This chapter gives the details of modelling and programming of the individual and interconnected networks. A more detailed information of global variables and subroutines used is given in the Appendix at the end of this chapter. In the following section various modelling assumptions have been listed.

### 3.1 Modelling Assumptions

Following are the assumptions that have been used in the performance modelling of the networks:

- (1) The transmission medium is assumed to be completely error free.
- (2) The packet arrival processes at the nodes other than the bridges/routers are assumed to be Poisson.
- (3) The packet lengths can either be fixed or exponentially distributed.
- (4) The propagation delay is assumed to be 5 km cable length.
- (5) The First Come First Served (FCFS) service strategy has been assumed.
- (6) The effects of various error conditions and recovery procedures on the delay performance of the network have been excluded.
- (7) In the CSMA/CD, delays at the repeaters have been neglected. However, these delays can be taken care of by specifying a media length greater than the actual physical media length.

(8) The bridges/routers have been assumed to be two-way. The operation of a bridge/router in each direction has been assumed to be independent of operation in the other direction.

(9) The processing and other delays at the bridges/routers have been neglected.

(10) A packet is said to have arrived at the bridge/router when the last bit of the packet has been received.

## **3.2 General Modelling Approach**

The event scheduling approach has been used for the performance modelling of the networks. The selection of this modelling approach is natural as networks have many arrivals and fewer activities. For example, consider a network having one hundred nodes and each node having eight classes and priority traffic. It requires a total number of eight hundred arrival event clocks to be scanned before advancing the simulation time to next event time, if the activity scanning approach were used. The process interaction approach also does not offer any additional advantage and programming-wise the event scheduling approach looks more simpler.

### *3.2.1 Event List Structure*

A two layer event list structure has been used. Each component network of an interconnected system has its own future event list. The top events of each component network have been used to construct a super event list. The super event list as well as individual networks event lists have been implemented in a doubly linked linear data structure. For insertion of the new

events in the individual networks event lists, the middle pointer method (with direction of scan from the top of half to which new event belongs) has been used. For insertion of new top event of a network in the super event list, a simple top to bottom scan method has been used as the size of super event list is very small. The size of super event list may reach a maximum of events in the case of an interconnected system having component networks.

The use of a two layered event structure makes new event insertion quicker as only a sub list of events corresponding to a particular network needs to be searched. The extra overheads associated with the maintenance of the super event list are less expensive than, if a single event list were used for all the component networks having a large number of events.

The individual networks event lists are stored in a two dimensional array  $evtime(i,j)$ , where the index  $i$  indicates network number and  $j$  indicates event number. The event number signifies the type of event in an encoded manner. For example, in a token ring with  $N$  nodes, event numbered 1 corresponds to token arrival at a node and event numbered  $k$  ( $k \leq N+1$ ) corresponds to packet arrival event at node number  $K - 1$  (For the complete details of event numbering for each MAC protocol, see Appendix at the end of this chapter). An array  $itop(i)$  is used to store the top event number of each component network. The index  $i$  indicates the network number. The super event list is stored in an array  $topeve(n)$ . The array index,  $n$ , indicates the network

number. A variable called *m<sub>top</sub>* points to the top of the super event list.

### *3.2.2. Selection of Next Event.*

The selection of next event is a two step procedure. The variable *m<sub>top</sub>* gives the network number *n*, and *i<sub>top</sub>(n)* gives the top event number *k* of network number *n*. These two values *n* and *k* are used to select event *evtime(n,k)*. The simulation time is advanced to this event time and *m<sub>top</sub>* and *i<sub>top</sub>(n)* are updated. An appropriate MAC protocol subroutine is called with parameters *n* and *k*. The details of these MAC protocol subroutines are given in the next section. After return from the subroutine, the new top event of network *n* is placed in the super event list at the appropriate place, and the cycle is repeated. The length of the simulation run is calculated from the average packet arrival rate and the average number of packet arrivals per node to be simulated.

### *3.2.3 Packet Arrival and Packet Transmission Completion Events*

We are discussing these two events separately here as these events are common to all MAC protocol models. Whenever an arrival occurs, packets present count is incremented and the arrival time of packet is stored in a circular buffer *atbuff*. The pointer indicating the location of the latest arrival in the buffer is updated. The packet arrival instant recorded now, is later used to calculate delay experienced by this particular packet. If the buffer is full, the packet is discarded. A random interarrival time from an exponential distribution (with parameter  $\delta$  equal to the corresponding packet arrival rate) is

generated and the next packet arrival is scheduled at a time equal to current time plus interarrival time.

Whenever a node gets a chance to transmit its packet(s), time to transmit a packet of length  $L$  is calculated and the packet transmission completion event is scheduled after that much time. Packet length  $L$  may be fixed or exponentially distributed. The packet transmission start time is recorded. When the packet transmission completion event occurs, packets present count is decremented. A pointer to *atbuff* indicates oldest packet not yet served. The packet arrival instant stored at that location in *atbuff* and packet transmission start time are used to calculate the delay experienced by this particular packet. The oldest packet not yet served pointer is advanced to the next location of the buffer.

The destination network of a packet is determined by tossing an  $N$ -faced biased coin in the source network. The probability of each face is pre-specified. The face probabilities sum to one. Each component network has its own "coin". This  $N$ -faced biased coin is implemented using a uniform random number generator. The range  $(0,1)$  of uniform distribution is divided in  $N$  subranges. The lengths of these subranges are such that probability of random number falling into a particular range is equal to the corresponding face probability. If the packet is for outside the network, an arrival at an accepting traffic for that network is scheduled. The scheduled time of the packet arrival at the bridge corresponds to the time when the bridge will receive the last bit of the transmitted packet.

## 3.3 Modelling of The MAC Protocols

### 3.3.1 Token Ring Modelling

The future event list of the Token Ring consists of the following events: (1) The token arrival event at the next node, (2) The next packet arrival events for each node, (3) and the packet transmission completion event. When the event (1) is present in the list, event (3) will not be present in the list and vice versa. The arrival events at the nodes other than the bridges/routers are unconditional events, therefore these events will always be in the list.

An integer array token is used to keep the current position of the token in the ring and the current token owner node number. A zero in the token owner field indicates an idle token. The token circulation is simulated as follows: suppose the token is currently at the node  $i$ , then after a passage of simulation time equal to the walk-time between the nodes  $i$  and  $i+1$  token is moved to the node  $i+1$  and the token position field is updated.

When an idle token arrives at a node  $i$  and if that node has one or more packets, then the token owner field is changed from zero to  $i$  and packet transmission completion event is scheduled. If node  $i$  has no packet, then the arrival of token at the next node is scheduled. When the packet transmission completion event occurs then the service limit of the networks is checked. If the token owner node has one or more packets and the service limit has not been reached, then the transmission completion event of the next packet is scheduled. Otherwise, token owner field is



changed to zero and arrival of token at the next node is scheduled.

### 3.3.2 IEEE 802.5 Token Ring Modelling

The future event list of the 802.5 token ring has the same type of events as the simple Token Ring. Since 802.5 Token Ring has a priority operation, the future event list contains next packet arrival events for each priority class at each node. The bridges/routers have only the lowest priority packet arrival events as internet traffic is assigned the lowest priority.

The token is circulated in the same way as in the simple token ring. However, unlike the simple token ring, the token is circulated as a Starting Frame Sequence (SFS) once around the ring after a node has got right to transmit and packet transmission completion event has been scheduled. The circulation of SFS allows other nodes to make reservations for the priority of the next token to be issued.

The *token* array contains three more fields in addition to the token owner and the token position fields. The priority field keeps current token priority. The reservation field keeps reservation request. The packet priority field contains the priority of the packet being transmitted if any transmission is in the progress. Whenever a node *i* raises the token priority from  $P_1$  to  $P_2$  and the node number *i* is stored in the stacking node field of *ipstack* array and the old priority  $P_1$  is stored in the old priority field of *ipstack* array  $P_2$  is used as the pointer to these locations in the array. An array *ihstpr* is used to store the priority of the highest priority packet available at each of the nodes.

When an idle token or SFS arrives at a node  $i$ , actions taken are explained with the help of following pseudocode. In the following pseudocode,  $\rightarrow$  represents a pointer and '.' is used to specify a particular field of a array.

```
if (token.owner = 0) then /*token is idel*/
    if (token.priority > ihstpr(i)) then
        if (token.priority->ipstack.stacking node = i) then
/*node i has raised the priority to current priority level*/
            token.priority->ipstack.stacking node = 0
            if (token.reservation > token.priority->ipstack.old prio-
rity) then
/*reservation request is greater than old priority*/
                token.reservation->ipstack.old priority
                    = token.priority->ipstack.old priority
                token.reservation->ipstack.stacking node = i
                token.priority = token.reservation
                token.reservation = 0
            else
                token.priority = token.priority->ipstack.old priority
/* old priority restored */
            endif
        elseif (token.reservation < highest priority packet at node i)
            then
                token.reservation = ihstpr(n,i)
            endif
        elseif (number of of packets at node i  $\geq$  1) then
            token.owner = i
```

```

        token.packet priority =ihstpr(i)
        schedule packet transmission completion events;
    endif
    schedule arrival of token.SFS at next node;
elseif (token.owner#i)then
    /*SFS sent by some other node*/
    if (token.reservation<ihstpr(i))then
        token.reservation =ihstpr(i)
        schedule arrival of SFS at the next node;
    endif
else
    /* SFS sent by node i has been received back */
    do nothing;
endif

```

When the packet transmission completion event occurs, the delay statistics for the priority class  $j$  ( $j=\text{token.packet priority}$ ) at the node number  $i$  ( $i=\text{token.owner}$ ) are updated. The  $\text{ihstpr}(i)$  is modified, if necessary. If the service limit has not been reached and node as packet(s) of priority greater than the token priority, transmission completion event for the next packet and SFS arrival event at next node are scheduled. Actions taken otherwise are explained again with the help of pseudocode.

```

    Token.owner =0 /* token is made idle*/
    if ((token.priority ≥ max(token.reservation,ihstpr(i)))then
        Token.reservation =max(token.reservation,ihstpr(i))/*token
priority is raised*/

```

```

else
    temp=token.priority
    token.priority =max(token.reservation,ihstpr(i))
    token.rservation = 0
    token.priority....>ipstack.stacking node-i
/*node i becomes stacking station*/
if (node i was not stacking station before OR
    temp->ipstack.stacking node#i)then
token.priority->ipstack.old priority = temp
else
    temp->ipstack.stacking node=0
    token priority->ipstack.old priority
    =temp->ipstack.old priority
endif
endif
endif

```

### *3.3.3 Ring/Bus Modelling*

The modelling of the Ring/Bus without priority is essential same as the simple Token Ring. Since walk-time in this scheme consists of only propagation delay between the two nodes, the same model of Token Ring with nodal delays set to zero is used to simulate this scheme. The delay introduced by the active repeater is taken into account by suitably increasing the walk-time between the two nodes which keep an active repeater in between them.

#### Ring/Bus with Priority Modelling

The future event list of the Ring/Bus contains the same

type of events as the 802.5 Token Ring. Only the token priority field, in addition to the token owner and the token position fields of the token array is used. The token reservation field is not used as there is no provision for reservations. Since the priority of the token itself indicates the priority of the packet being transmitted, packet priority field is also not used. Whether a node  $i$  of the network  $n$  is transmitting or not is indicated by flag  $ntxf(n,i)$  and whether the node  $i$  is in delayed-PR mode or not is indicated by a flag  $dprf(n,i)$ . The priority of the highest priority packet available at node  $i$  is kept in  $ihstpr(n,i)$ .

When an idle token or a busy token arrives at a node  $i$ , actions taken arrives at a node  $i$ , actions taken are explained with the help of following pseudocode.

```

If(number of packets at node  $i=0$ ) then
    schedule arrival of busy/idle token at the next node;
elseif (token.owner  $\neq i$ )then
    if ( $ntxf(i) = \text{set}$ ) then
        /*transmission of node  $i$  has been pre-empted*/
         $ntxf(i) = \text{reset}$ 
        if ( $dprf(i) = \text{reset}$ ) then
             $dprf(i) = \text{set}$ 
    increase the walk-time between node  $i$  and  $i+1$  by token length;
    endif
    schedule arrival of busy/idle token at the next node;
endif
if(token.priority< $ihstpr(i)$ )then

```

```

        if(token.owner #0) then /*token is not idle*/
            delete packet transmission completion event;
            /*Transmission of other node is pre-emp[ted */
endif
    token.owner = i
    token.priority = ihstpr(i)
    ntxf(i) = set
    schedule packet transmission completion event;
endif
else
    /*Transmission of node i has not been pre-empted*/
endif

```

When the packet transmission completion event occur, delay statistics for priority  $j$  ( $j=\text{token.priority}$ ) at node  $i$  ( $i=\text{token.owner}$ ) are updated. The  $ihstpr(n,i)$  is modified, if necessary. If  $dprf(n,i)$  is set, it is reset and the walk-time between the nodes  $i$  and  $i+1$  is again reduced to propagation delay only. If the service limit has not been reached and node  $i$  has one or more packets to send, the packet transmission completion event for the next packet is scheduled. The arrival of busy/idle token at the next node is also scheduled.

### 3.3.4 CSMA/CD Modelling

The future event list of the CSMA/CD consists of the following events: (1) channel idle at a upstream node event, (2) channel idle at a downstream node event, (3) collision backoff period completion events for the nodes which are backlogged ( A node is called backlogged if its packet transmission attempt had

resulted in a collision and it has not succeeded in transmitting that packet yet), (4) Next packet arrival event for each node, and (5) packet transmission completion event.

The events (1) and (2) are scheduled every time after a collision and a successful transmission. Two variables *csups* and *csdns* keep the node numbers of upstream and downstream node respectively. A carrier sense flag *csf(n,i)* is used to indicate the channel status (idle/busy) as seen by the node *i* of the network *n*. Whenever the event (1) or (2) occurs carrier sense flag of the node given by *csups* or by *csdns* is reset. If that node does not attempt to acquire the channel, the channel idle event for the next upstream/downstream node is scheduled.

A variable *turn* keeps the number of the node which is currently transmitting. The packet transmission attempt count and backoff count for a node *i* are stored in *nattempt(n,i)* and *nbackoff(n,i)* respectively. These counts are used by backoff algorithm to schedule backoff periods. A node will attempt to acquire the channel in the following conditions.:

- (1) it has just sensed the channel idle end,
  - (a) It is not backlogged and it has one or more packets to transmit.
  - (b) It is backlogged and its backoff period has ended already
- (2) If the channel is sensed idle and its backoff period completes.
- (3) If the channel is sensed idle, the node is not backlogged and an arrival has just taken place.

Whether the channel acquisition attempt results in a success or a collision is simulated as follows: Whenever a node  $i$  attempts to acquire the channel for its packet transmission, the node state information (node is backlogged or not and the number of packets present at the node) and list of scheduled events associated with every node other than the node  $i$  are checked to find the possibility of a collision. We define collision window between the node  $i$  (attempting to transmit) and a node  $j$  as the current simulation time plus propagation delay between the two nodes. In the following conditions, a collision will result.

- (1) If any other node also simultaneously attempt to acquire the channel.
- (2) If any other node  $j$  will have a packet arrival event within the collision window and the node  $j$  is not backlogged.
- (3) If any other node  $j$  which is backlogged and whose collision backoff period will be completed within the collision window.

If the exhaustive checking at every node shows that the packet transmission attempt by the node  $i$  will result in a collision then the farthest upstream and downstream nodes participating in the collision are found out. The collision detection time and subsequent transmission stop times for these nodes are calculated. From these times, the time when the channel will again start appearing idle is calculated and channel idle events (1) and (2) are scheduled in the event list. The backoff periods for all the colliding



transmissions are generated using the collision backoff algorithm and these times are scheduled in the event list. If this exhaustive checking shows that no collision will take place then the packet transmission completion event is scheduled in the event list. When the packet transmission completion event occurs then the channel idle event list. When the packet transmission completion event occurs then the channel idle events are scheduled to occur after a delay equal to the interframe gap.

### **3.4 Modelling of The Bridge/Routers**

A Bridge/router connecting the two networks  $i$  and  $j$  is modeled as a pair of nodes. One node of the pair  $n_i$  belongs to network  $i$  and the other node  $n_j$  belongs to network  $j$ . Whenever the network  $i$  ( $j$ ) has a packet which need to be passed through this bridge/router, it schedules an arrival at the node  $n_j$  ( $n_i$ ). When a network has a packet for a network which is not directly connected through the bridge/router, the packet is sent to a network which will forward the packet further to reach its destination. This path information is kept in a routing table. To send a packet from a network  $i$  and network  $j$ , this routing table is searched to find the node number  $n_k$  of a network  $k$  at which a packet arrival should be scheduled. The packet destination  $j$  and the packet lengths are stored in the buffer at the bridge/router corresponding to node  $n_k$ . The two sides of the bridge/router have independent buffers to store destination and packet length information. The routing table is constructed

using the interconnected information before the start of simulation. The path between the two networks can either be a minimum hop path (calculated using Dijkstra's algorithm with cost of each hop equal to one.) or any other possible path specified by the user.

# APPENDIX

## Programming Details

### EVENT NUMBERS

Events are numbered assuming number of nodes equal to NN. If a network has a priority operation then M classes of priority are assumed.

MAC PROTOCOL	EVENT NUMBER	EVENT TYPE
Token Ring & Ring/Bus Without Priority	1 $1 < k < NN+2$	Token Arrival at a node Packet Arrival at the node (k-1)
	NN+2	Packet Transmission Completion
IEEE 802.5 Token Ring & Ring /Bus with Priority	1 $1 < k < (NN*M+2)$	Token Arrival at a node Packet Arrival of priority j at the node i , where $j = \text{mod}(k-1, M)$ (if j=0 then j=M) $i = (k-1) / M$ (if j=0 then i=i+1)
	(NN*M+2)	Packet Transmission Completion

CSMA/CD	1	Channel idle at a upstream node
	2	Channel idle at a downstream node
	$2 < k \leq (NN+2)$	Backoff period completion for the node k-2
	$(NN+2) < k < (2*NN+3)$	Packet arrival at the node (k-(NN+2))
	NN+3	Packet Transmission Completion

### Dummy Events

If the number of unconditional events in a network are less than five, then for maintaining the event list structure, dummy events are introduced in the list. These dummy events are numbered beyond the range defined above and are scheduled to occur at the times beyond the maximum simulation clock time. Therefore, these events never occur.

## GLOBAL VARIABLES

*nofnet*: number of networks in the system: integer

*mtransped*: maximum of the individual networks' transmission

speeds: real

*mxpktsiz*: maximum packet size allowed in the system: integer  
*p(i,j)*: fraction of the traffic generated in the network i routed to the network j: real  
*noflinks*: number of the bridges in the system: integer  
*nic(i,j)*: network interconnection from i to j flag. 1 = connected, 2 = unconnected: integer.  
*ipopt*: path option flag. 1 = minimum hop path, 2= user specified path: integer  
*iroutetab(i,j,k)*: k contains the routing information for the packets from the network i to the network j. The field of k are (1) network number, (2) bridge number of network, i, and (3) node number of the network given in the field (1): integer  
*noisbrf(i,j)*: If the node j of the network i is a bridge/router, then it contains the bridge number. Otherwise it contains the zero: integer  
*btonmap (i,j)*: It contains bridge to node mapping information for the bridge j of the network i: integer  
*plbuff(i,j,k)*: k is a packet length information buffer at the bridge j of the network i: integer  
*pdbuff(i,j,k)*: k is a packet destination information buffer at the bridge j of the network i : integer  
*ppdelay(i,j,k)*: propagation delay from the node j to the bridge k of the network i: real

*ipkt(i,j)*: number of the internal packets routed to the bridge j to network i: integer

*tapdelay(i,j)*: average transmission and propagation delay experienced by the internal packets routed to the bridge j of the network i: real

*btbpatd(i,j,k)*: average propagation and transmission delay experienced by the external packets arriving at the bridge j and being routed to the bridge k of the network i: real

*ntype(i)*: MAC protocol type for the network i: integer

*tonf(i)*: type of the network i flag 1 = symmetric, 2 = asymmetric: integer

*nofpr(i)*: number of the priority classes in the network i: integer

*nn(i)*: number of the nodes in the network i: integer

*nofbr(i)*: number of the bridges in the network i: integer

*medialen(i)*: media length of the network i: real

*pdelay(i,j)*: propagation delay from node j to the next node of the network i: real

*parf(i)*: packet arrival rate flag for the network i,  
1 = same for all the priority classes at all the nodes ,  
2 = same for each priority class at all the nodes, and  
3 = different for each priority class at each node: integer

*mplf(i)*: mean packet length flag for the network *i*, defined in the same way as the *parf(i)*: integer

*npf(i)*: node placement flag. 1=user specified, 2 = uniformly distributed: integer

*pldf(i)*: packet length distribution flag. 1 = constant , 2 = exponentially distributed: integer

*mnpktsz(i)*: minimum packet size for the network *i*:integer

*overhead(i)*: overheads per packet in the network *i*: integer

*sysclk(i)*: simulation clock of the network *i*: double precision

*transped(i)*: transmission speed of the network *i*: real

*scalef(i)*: simulation clock scale factor for the network *i*: real

*ptscalk(i)*: latest packet transmission start time in the network *i*: double precision

*iboflow(i)*: number of packets discarded in the network *i* due to buffer overflow: integer

*npdcarded(i)*: number of the packets discarded due to collisions in the network *i*:integer

*ipl(i)*: length of the last packet transmitted in the network *i*:integer

*tosf(i)*: type of service flag for the network *i*. 1 = exhaustive , 2 = limited, 3 = fixed time: integer

*slimit(i)*: service limit in the network *i*:integer

*pscnt(i)*: packets transmitted after acquiring the token by a node in the network *i*:integer

*mtht(i)*: maximum time for which token can be hold in the network i:real

*ttht(i)*: timer holding token for the network i:real

*ridctime(i)*: ring latency time in the network i:real

*ndelay(i)*: nodal delay in the network i:integer

*toklen(i)*: token length in the network i:integer

*rotf(i)*: release of the token flag for the network i. 1=just after the packet(s) transmission, 2= after transmitted packet(s) returned back integer

*token(i,j)*: j contains token descriptor of the network i. The field of j are (1) token owner, (2) token priority, (3) reservation priority, (4) token position, (5) priority of the packet being transmitted: integer

*narep(i)*: number of the active repeaters in the network i:integer

*iapos(i,j)*: j contains active repeater positions in the network i:integer

*dprf(i,j)*: flag indicating the mode of the node j of the network i. 0=normal mode, 1 = delayed-PR mode:integer

*ntxf(i,j)*: transmission status of the node j of the network i . 0 = not transmitting, 1 = transmitting: integer

*ihstpr(i,j)*: highest priority packet available at the node j of the network i:integer

*ipstack(i,j,k)*: priority stacking information for the network i.k contains the number of the node which has raised the priority to the level j, and the original priority level before raising to the level j:integer



*stprcnt(i,j)*: number of the priorities stacked at the node j of the network i:integer.

*slotime(i)*: slot time in the network i:integer

*jmseq(i)*: jam sequence size in the network i:integer

*mattempt(i)*: transmission attempt limit in the network i:integer

*mbackoff(i)*: backoff limit in the network i:integer

*csf(i,j)*: carrier status flag for the node j of the network i. 0 = channel idle, 1 = carrier present:integer

*csdns(i)*: upstream node of the network i which is going to sense the channel idle:integer

*txnn(i)*: number of the node currently transmitting in the network i:integer

*nblogf(i,j)*: backlog status of the node j of the network i. 0 = unbacklogged ; 1 = backlogged:integer

*nattempt(i,j)*: current value of the backoff count at the node j of the network i:integer

*rnode(i,j,k,l)*: l contains packet arrival rate and mean packet length for the priority class k at the node j of the network i:real

*inode(i,j,k,l)*: i is the descriptor for the priority k at the node j of the network i. The fields of the l are (1) number of packets present, (2) total number of packets transmitted successfully: integer

*icnode(i,j,k)*: k is the descriptor for the node j of the network i. The field of k are (1) maximum number of packets present, (2) number of packets present: integer

*atbuff(i,j,k,l)*: l is the buffer

containing arrival times of the packets for the priority  $k$   
 at the node of  $j$  of the network  $i$ : double precision  
*ioldpkt(i,j,k)*: pointer for the atbuff pointing to the  
 arrival time of the oldest packet of the priority  $k$  at the  
 node  $j$  of the network  $i$  not yet transmitted. integer  
*ilatpkt(i,j,k)*: pointer for the atbuff pointing to the arrival  
 time of the latest packet of priority  $k$  at the node  $j$  of the  
 network  $i$ :integer  
*snode(i,j,k,l)*:  $i$  contains mean and the second moment of the  
 packet delay for the priority  $k$  at the node  $j$  of the network  
 $i$ :integer  
*topeve(i)*: top event time of the network  $i$ :double precision  
*mnptr(i)*: It contains the index of the following event of  
 the event in the super event list :integer  
*mpptr(i)*: it contains the index of the preceding event of  
 the event  $i$  in the usper event list:integer  
*mtop*: It contains index of the top event in the super event  
 list: integer  
*evtime(i,j)*: event time of the event  $j$  of the network  $i$ .  
 $0$  = presently not in the event list,  $1$  = presently in the  
 event list:integer  
*inptr(i,j)*: next event from the event  $j$  pointer:integer  
*itop(i)*: event number of top event in the network  $i$ :integer  
*imid(i)*: middle event pointer in the event in the network  
 $i$ :integer  
*imid(i)*: middle event pointer in the event list of the  
 network  $i$ :integer

*ilast(i)*: highest event number in the network *i*:integer

*luhalf(i)*: size of the upper half of the event list of the network *i*:integer

*llhalf(i)*: size of the lower half of the event list of the network *i*:integer

## SUBROUTINES AND FUNCTIONS

*arrival(n,i,j,k)*: This subroutine schedules the next arrival the priority *j* at the node *i* of the network *n*, *k* is the event number.

*avarate(rnode, nofnet)*: This function calculates average arrival rate per node, excluding the bridges/routers over all the networks.

*biexpboff(n)*: This function generates a random integer *k* in the range  $0 \leq k < 2^n$ .

*cdetect(n,j,cowindow, pkttxst,cf)*: This subroutine checks if the node *j* of the network *n* will attempt to transmit its packet in the *cowindow*. The flag *cf* indicates the result of this checking, 0 = no collision, 1 = collision. If *cf* = 1, then *pkttxst* contains the time of its packet transmission start.

*chacquire(n,i)*: This subroutine is called when a node attempts to acquire the channel, *n* is the network number and *i* is the node number. It calls *cdetect* subroutine and checks the returned flag *cf*. If all the calls to *cdetect* returned *cf* flag reset, it calls a subroutine *pkttxt* to schedule packet transmission completion event. Otherwise, it calls a subroutine *chidle* to schedule channel idle events.

*chidl(n, fnupstx, fndnstx, txsbyfun, txsbyfnd)*: This subroutine determines a point on the media on which both the upward going signal and the downward going signal will reach at the same time. This point is determined using the farthest node upstream *fnupstx* and downstream *fndnstx* transmitting and their transmission stop times, *txsbyfun* & *txsbyfnd* passed by *chacquire*. With reference to this point channel idle events at the upstream and the downstream nodes are scheduled. The collision backoff period for all the nodes participating in the collision are also scheduled. *n* is the network number.

*csmacd(n,k, msysclk)*: This is the main subroutine simulating the CSMA/CD protocol. *n* is the network number, *k* is the event number and the *msysclk* is the master simulation clock time.

*destination(n,i)*: This subroutine generates the destination network number for a packet being transmitted from the node *i* of the network *n*. If the destination is not equal to *n*, it schedules an arrival at the appropriate node of the destination (intermediate) network.

*discard(n,i)*: If the transmission attempt limit has reached for a particular packet of the node *i* of the network *n*, that packet is discarded from the *atbuff*.

*fileread(valid)*: This subroutine reads the input from a file and checks their validity. If the file contains valid data, the flag *ivalid* is set to zero otherwise, it is set to 1.

*filwrite*: This subroutine writes input data into a file.

*getpath(nofnet, nic, iroutetab)*: This subroutine gets paths from the user for the networks not connected through bridge/router and prepares the routing table.

*ihstpr(n,i,j)*; This subroutine is called after a priority *j* packet transmission to update the highest priority packet available at the node *i* of the network *n*.

*inetdcal*: This subroutine calculates the packet delays between the each pair of the networks of a system.

*linklist(n)*: This subroutine prepares the future event list for the network *n*.

*mptradj(n)*: This subroutine adjusts middle event pointer of the future event list of the network *n*.

*mtinsert(n)*: This subroutine inserts the top event of the network *n* in the super event list.

*netdata(n)*: This subroutine gets input data from the keyboard for the network *n*.

*nodedata(n)*: This subroutine gets the individual node data from the keyboard for the network *n*.

*nstate(n,i,j)*: This subroutine updates node descriptors *inode* & *icnode* after the arrival event of priority *j* at the node *i* of the network *n*.

*pathsearch(i,j)*: This subroutine schedules packet transmission completion event of the priority *j* at the node *i* of the network *n*.

*printstat*: This subroutine calculates the result of the simulation and prints them on the display and optionally in a file.

*rnunf()*: This function generates a uniform random number in the range(0,1).

*rnset(iseed)*: This subroutine initializes the seed of the uniform random number generator.

*ranrnode(n)*: This subroutine places the nodes at the uniformly distributed distances on the medium.

*ringbuswp(n,k,msysclk)*: This is the main subroutine simulating the Ring/Bus with Priority protocol. *n* is the network number, *k* is the vent number and *msysclk* is the master simulation clock time.

*simulator*: It is the main simulator routine

*idelete(n,i)*: This subroutine deletes the event *i* from the future event list of the network *n*.

*tinsert(n,i)*: This subroutine inserts the event *i* in the future event list of the network *n*.

*tokenring(n,k,msysclk)*: This is the main subroutine simulating the Token Ring with priority and the Ring/Bus without priority, *n* is the network number, *k* is the event number and *msysclk* is the master simulation clock time.

*wtimestat(n,i,j)*: This subroutine updates mean and the second moment of the packet delay after the priority *j* transmission completion from the node *i* of the network *n*. It also updates node descriptors *inode* & *icnode*.



## CHAPTER 4

# EXPERIENCE WITH RUNNING THE SIMULATOR

In this chapter some of the sample results produced using the SIMNET simulator are presented. Wherever available, analytical results are given along with the simulation results. All the sample cases should be assumed symmetric unless specified otherwise. Almost all the results have been obtained after simulating more than 3000 packets on an average per node. This simulation length approximately gives 10% confidence interval widths at the 90% confidence level. We have not taken into consideration the correlation among successive output samples while generating confidence intervals. We have assumed the effects of transients on the means to be negligible as the simulation run length is long enough.

### 4.1 Individual Network Simulations

#### *4.1.1 Token Ring and Ring/Bus without Priority*

The throughput-delay performance of a Token Ring and a Ring/Bus without Priority with 100 nodes operating at 1 Mbps are summarized in the Table 4.1 along with the analytical results. The packet length is exponentially distributed with mean equal to 1000 bits. As can be seen from the table, the simulation results match very closely with the exact analytical results over the entire range of the offered load.

In Fig.4.1, the mean packet delays (normalized by the packet duration) for the Token Ring and the Ring/Bus with the parameters same as above (except that the packet lengths are constant) are compared. The Ring/Bus shows improvement over the Token Ring. In Fig 4.2, throughput-delay performance of a token ring with 10 nodes and operating at 4 Mbps under two service disciplines; exhaustive and limited-1 service are compared.

#### *4.1.2 IEEE 802.5 Token Ring*

In [10], authors have given an approximate analysis of the 802.5 Token Ring Priority Access Mechanism. In that paper simulation results for a 4Mbps Token ring with 16 nodes and two priorities are given along with the analytical results. The packet lengths were assumed exponentially distributed with mean length equal to 1024 bits. I repeated the same experiment with the SIMNET simulator. Results obtained from the two simulations are summarized in the Table 4.2 The results obtained in [10] are listed in the column Simulation 1 and those obtained from SIMNET are listed in the column Simulation 2. The closeness of the two simulation results should be noted.

In Fig. 4.3, simulation results for a IEEE 802.5 Token Ring with 10 nodes and 8 priority classes are given.

#### *4.1.3 Ring/Bus with Priority*

Fig 4.4., shows the throughput-delay performance of a Ring/Bus with 100 nodes and two priority classes. The simulation results are compared with the analytical results obtained using

the expressions given in [5]. The analytical and simulation results for priority 2 closely match as the assumptions used in the analysis do not come into the picture. For priority 1, simulation results show that Ring/Bus will exhibit better delay performance than obtained from the analytical results.

In Fig.4.5 the throughput-delay performance of the Ring/Bus with 10 nodes and 8 priority classes is plotted. The priority access mechanisms of the IEEE 802.5 Token Ring and Ring/Bus with priority are compared in Fig.4.6. The curve shows that the priority handling in the Ring/Bus is better than in IEEE 802.5 Token Ring [21].

#### *4.1.4 CSMA/CD*

The throughput-delay performance of a CSMA/CD network with 100 nodes operating at 1 Mbps is given in Fig.4.7. The delays obtained using the approximate analytical calculations are of the same order as that obtained from the simulation. The simulation results show better delay performance than that obtained from the approximate calculations. In Fig.4.8, I have plotted the upper and lower confidence limits of the mean delay along with the mean delay itself for a CSMA/CD network with 100 nodes and operating with IEEE 802.3 standard parameters. These have been plotted as a function of the length of the simulation run. This figure shows the expected effect of reducing the confidence interval as the simulation length improves. From figures such as these, I was led to believe that the confidence interval produced by our simulations is actually substantially better than the value computed using the calculated variance.

## 4.2 Effects of Initial Transients

To study of the effects of the initial transients on the mean delay in the Token Ring (with 10 nodes) and the CSMA/CD network (with 10 nodes), I conducted the following experiment. A simulation run length of 3000 packets per node was chosen. The packet delay samples were collected after discarding initial X% observations. The simulation length has been appropriately elongated to make up for the initial discarded observations. For the Token Ring, the experiment was repeated at three different traffic load values, 0.1 (low), 0.5 (moderate), and 0.9 (high). For the CSMA/CD, the experiment was performed for two traffic values, 0.1 (low) and 0.5 (high). Results for different values of X are summarized in the Table 4.3 and Table 4.4. From the table, it can be seen that the transients do not affect the means in the token ring for all the traffic values. In the case of CSMA/CD, transients introduce slight error at the high load values. It should be noted that this error will reduce with lower traffic per node. For the high traffic per node, longer simulation run length should be chosen.

## 4.3 Simulation of the Interconnected Networks

To study the errors introduced by assuming the arrival process at the bridge to be a Poisson process, I conducted the following experiment.

(a) Two identical Token Rings each with 50 nodes and connected through a bridge were simulated. Individual ring transmission speeds were kept at 4 Mbps and the packet lengths equal to

1000 bits. All the nodes other than the bridge were identical. Total internal traffic generated was kept at 0.5. A 0.2 fraction of the internal traffic was routed through the bridge to the other Token Ring. The queueing and access delays at the bridge were found to be 231.84  $\mu$ sec (confidence interval 7.36  $\mu$ sec).

(b) A single Token Ring with the same parameters as in the part(a) was simulated. The node acting as the bridge earlier now acts as a simple node with a packet arrival rate same as at the bridge. However, arrival process is now Poisson. The packet delay at that node is now found to be 259.9  $\mu$ sec (confidence interval 8.8  $\mu$ sec.)

This experiment shows that assuming arrivals at the bridge to be Poisson will give pessimistic delays.

Another interconnected system with three networks (2 Token Rings and 1 CSMA/CD, each with 50 nodes) was simulated with two different interconnection patterns. In the case 1, Fig.4.9, Two Rings are connected to the CSMA/CD through bridges. The Packet Delays experienced at moderate loads (case 1A) in each network are summarized in Table 4.5 In the case 1B, the backbone CSMA/CD network is heavily loaded, as shown by the drastic increase in the delays in the Table 4.6 In the case 2, one additional bridge between the two Rings has been introduced. Load conditions are approximately equal to that in case 1A. Improvement in the delay performance for the packets going from one Token Ring to another Token Ring can be seen in the Table 4.7.

## 4.4 CPU Time Requirements

Most of the simulation runs were carried on the VAX/VMS system. The CPU time requirements for different cases of individual networks are tabulated in the Table 4.8.

These times are given for a run length equal to 3000 packets per node and for three different load values;  $p=0.1$   $p=0.5$  and  $p=0.9$ . The CPU time requirements increase linearly with the number of packet per node for a particular simulation. The CPU time requirements with increasing number of nodes or decreasing traffic load increases at a faster rate than linearly.

Offered Load	Packet Delay ( $\mu$ sec)			
	Token Ring		Ring/Bus without Priority	
	Analytical	Simulation	Analytical	Simulation
0.10	169.53	168.77	113.89	116.45
0.20	315.60	317.33	253.10	257.31
0.30	503.57	509.46	432.14	438.90
0.40	754.16	760.08	670.80	675.30
0.50	1105.00	1112.85	1005.00	1005.21
0.60	1631.25	1641.87	1506.25	1520.93
0.70	2508.30	2531.86	2341.67	2384.17
0.80	4262.50	4276.04	4012.50	4025.72
0.90	9619.90	9113.49	9029.25	9028.61
0.95	20461.20	20404.92	19050.48	18416.32

Table 4.1 Delay Performance of the Token Ring and The Ring/Bus

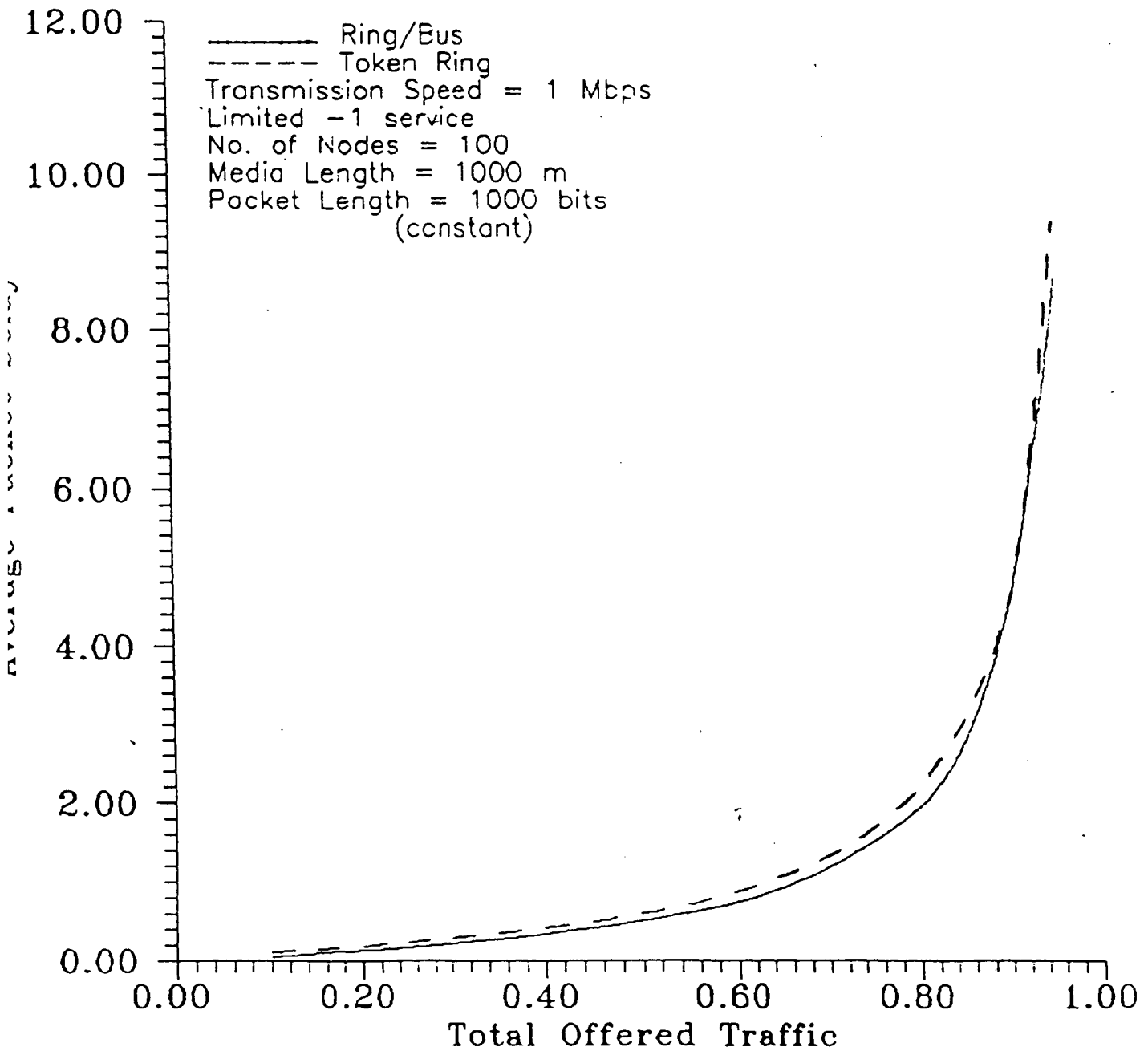


Fig. 4.1: Performance Comparison of Token Ring and Ring/Bus.  
 (Delays normalized by packet duration.)



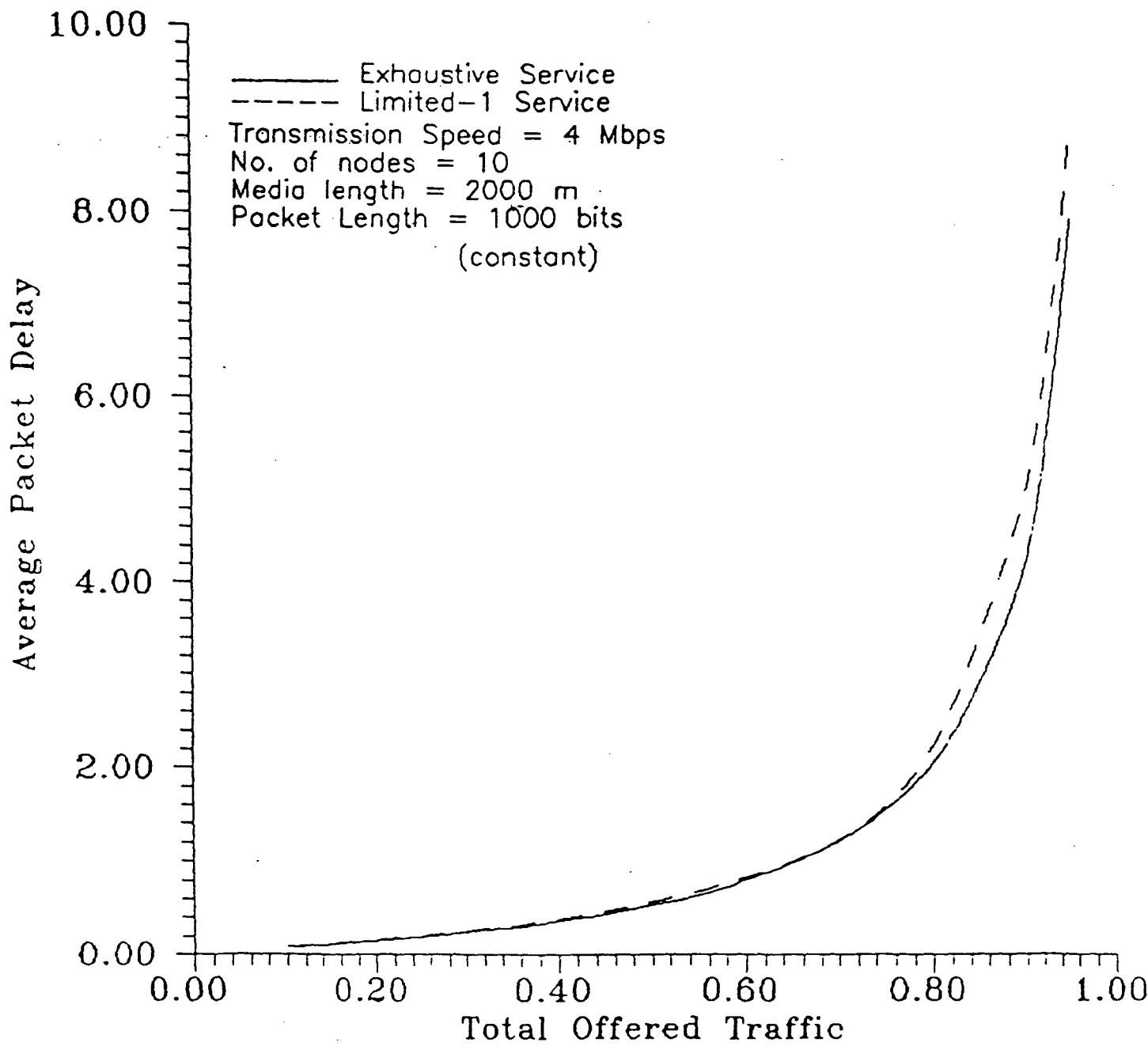


Fig.4.2: Token Ring Performance Under Different Service Disciplines.  
 (Delays normalized by packet duration.)

Offered Load	Packet Delay ( $\mu$ sec)			
	Simulation 1*		Simulation 2	
	Priority 1	Priority 2	Priority 1	Priority 2
0.2	81.00	81.00	744.96	70.36
0.3	108.00	108.00	129.62	110.44
0.4	216.00	162.00	326.78	212.15
0.5	351.00	216.00	326.78	212.15
0.6	540.00	270.00	540.09	279.38
0.7	919.00	324.00	915.17	360.31
0.8	1730.00	486.00	1971.55	485.08
0.9	4757.00	648.00	4550.58	653.25

Table 4.2 IEEE 802.5 Token Ring Simulation Results

(\*from [10])

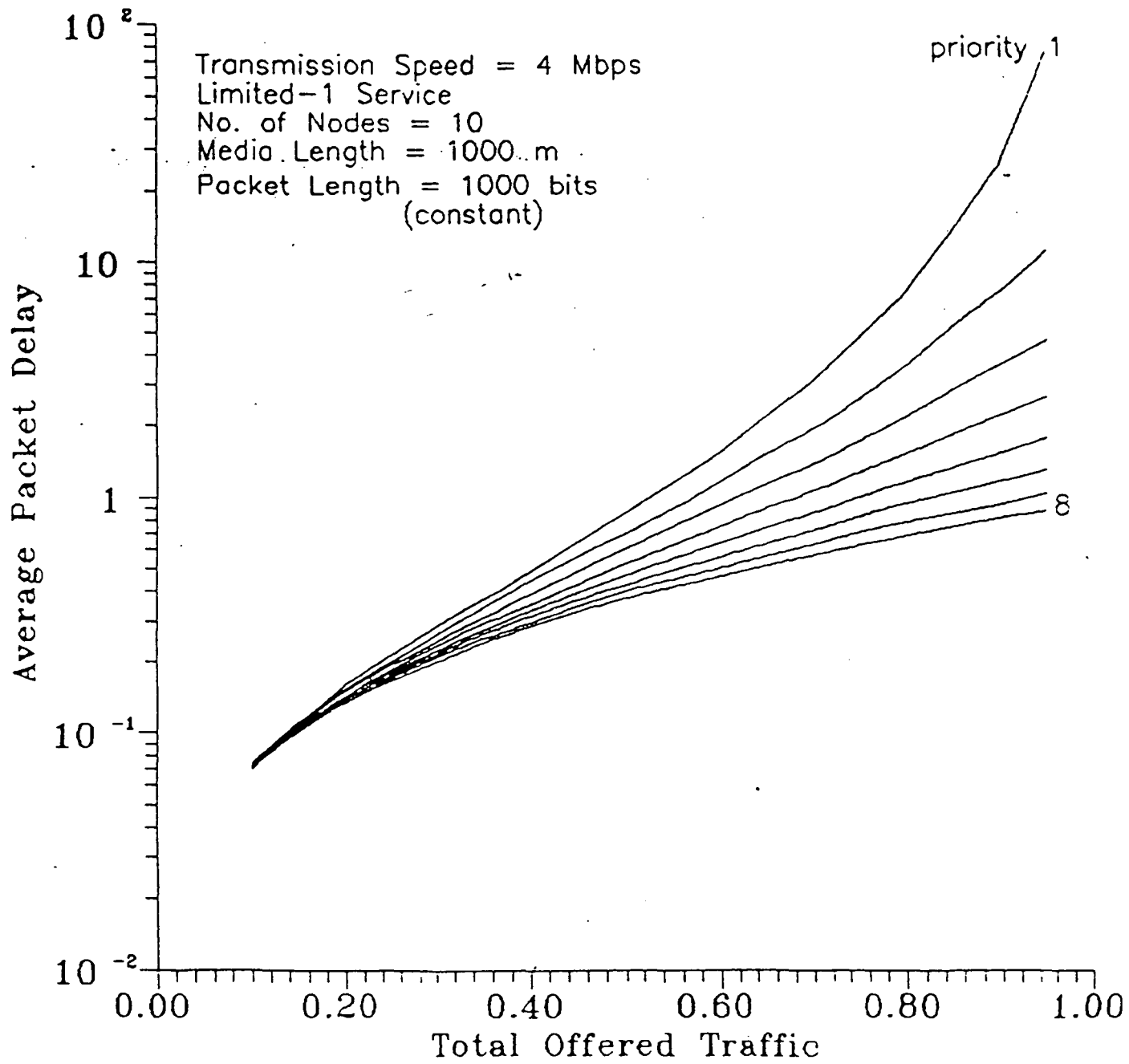


Fig. 4.3: IEEE 802.5 Token Ring Performance.  
 (Delays normalized by packet duration.)

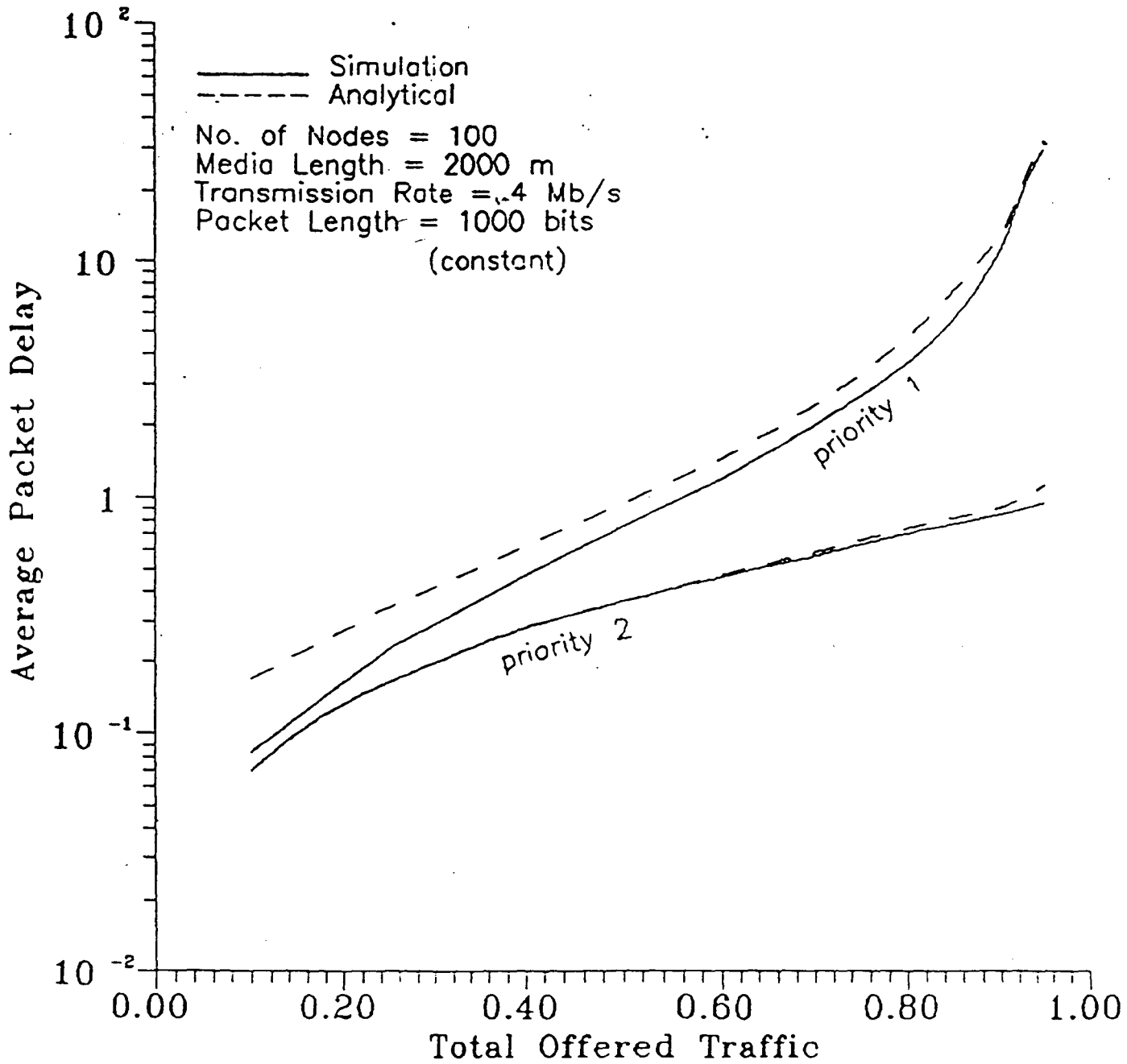


Fig.4.4: Performance of Ring/Bus with Two Priority Classes.  
 (Delays normalized by packet duration.)

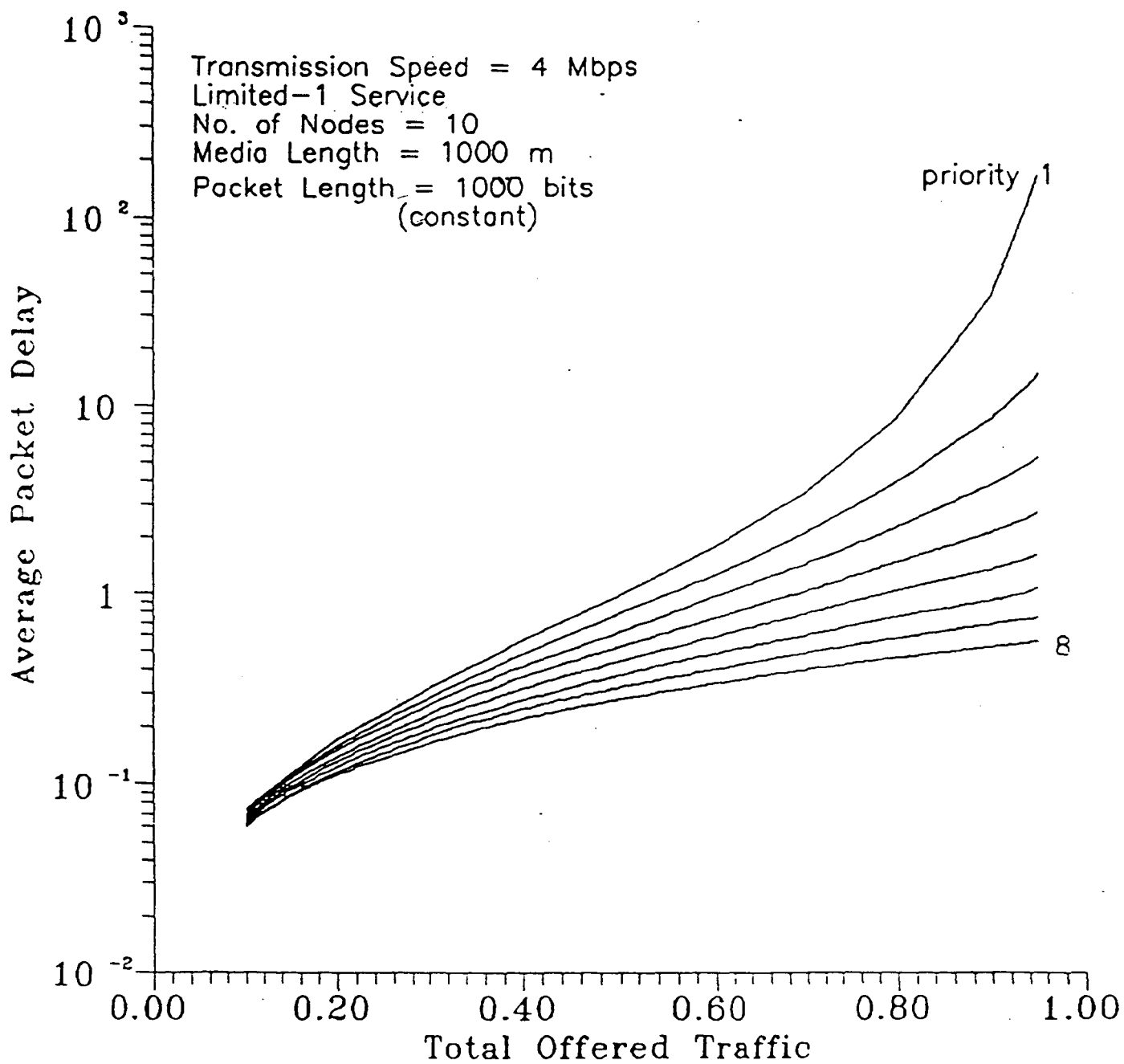


Fig 4.5: Performance of Ring/Bus with Eight Priority Classes.  
 (Delays normalized by packet duration.)

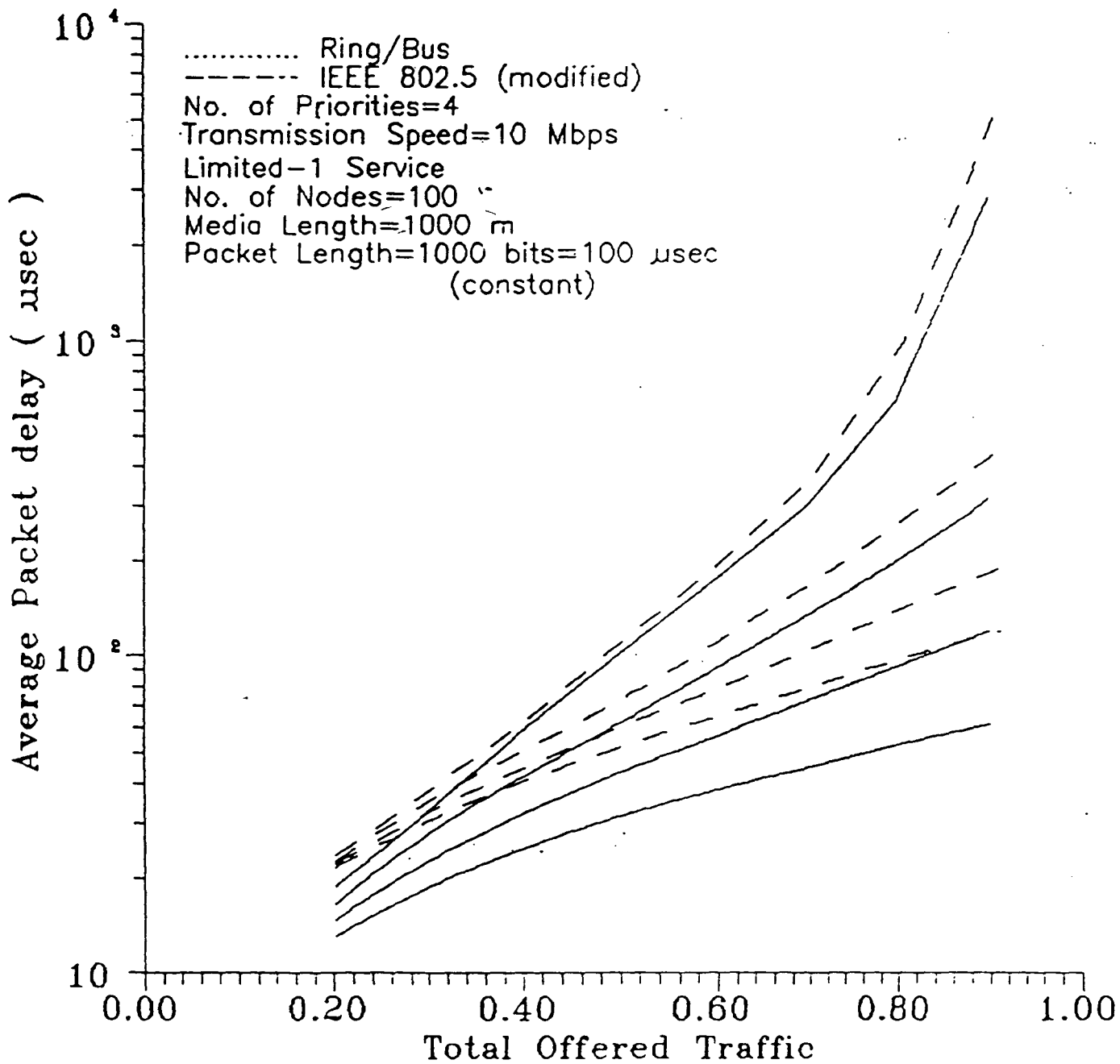


Fig.4.6: Performance Comparison of IEEE 802.5 Token Ring and Ring/Bus with Priority.

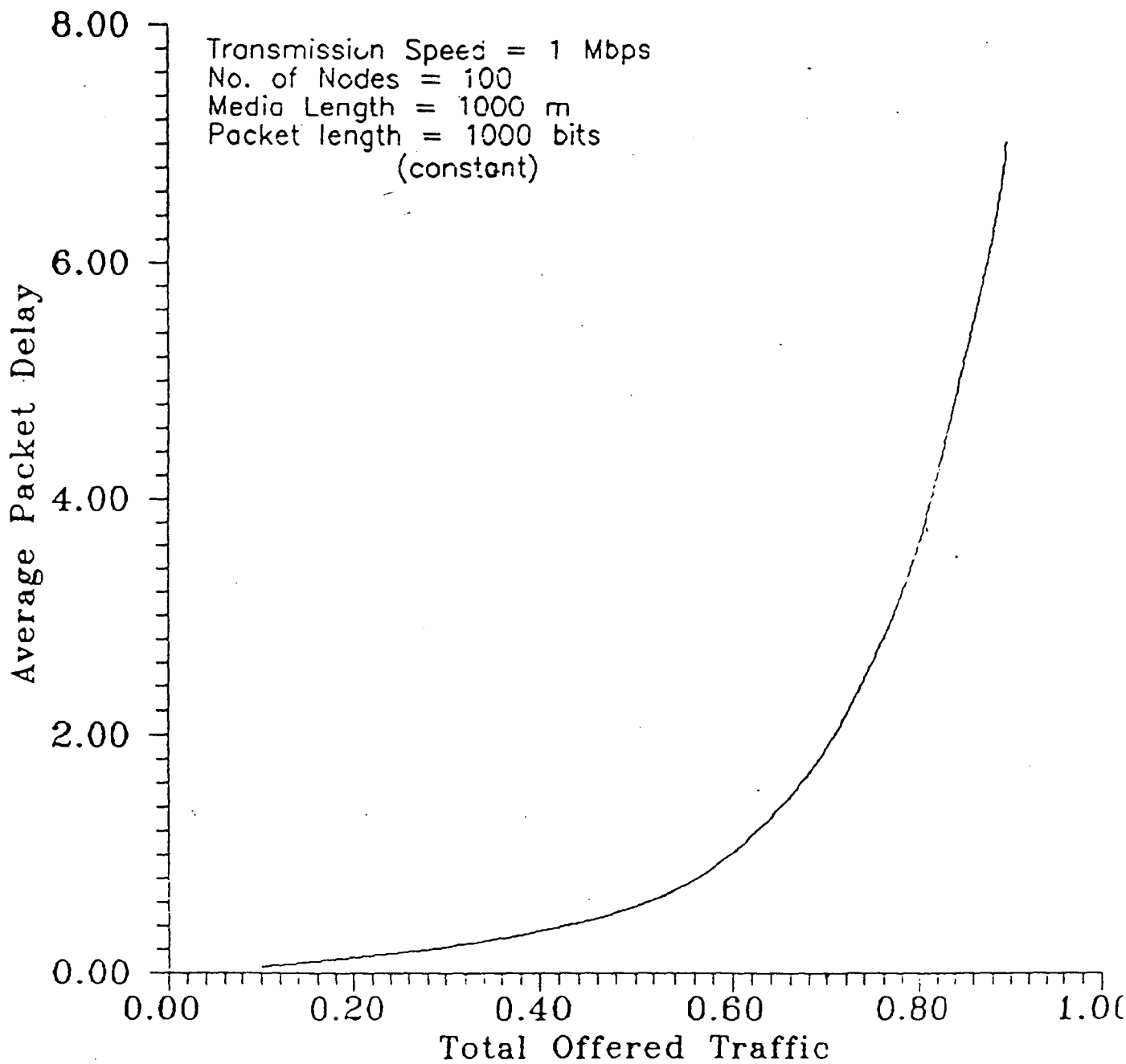


Fig.4.7: Performance of CSMA/CD.  
 (Delays normalized by packet duration.)

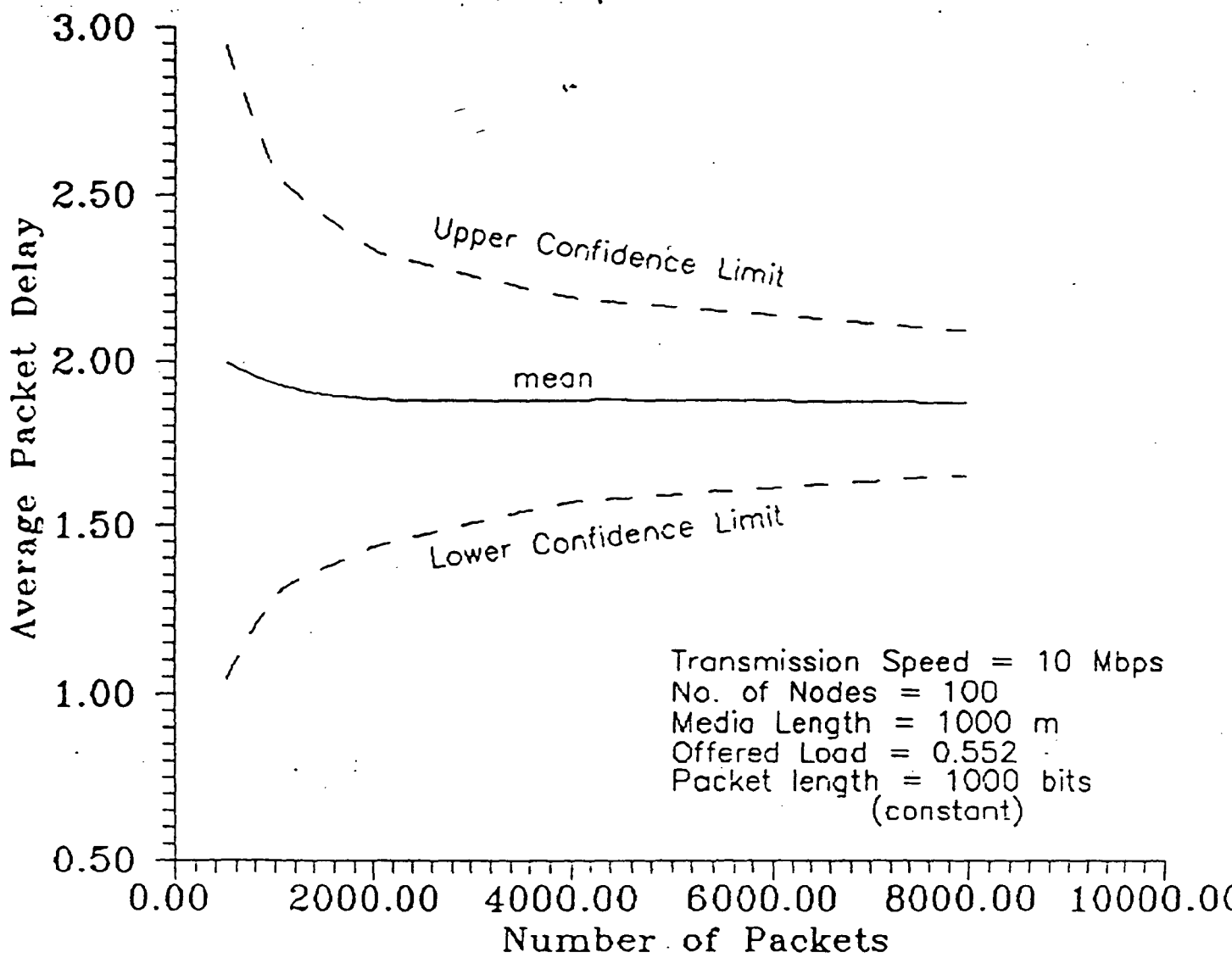


Fig.4.8: Confidence Intervals For CSMA/CD.  
 (Delays normalized by packet duration.)



Offered Load	Packet Delay ( $\mu$ sec)			
	0%	1%	5%	10%
0.1	20.60	20.56	20.46	20.58
0.5	140.73	140.35	140.31	141.54
0.9	1319.13	1311.29	1299.72	1319.02

Table 4.3 Effect of Transients in Token Ring

Offered Load	Packet Delay ( $\mu$ sec)			
	0%	1%	5%	10%
0.1	9.22	9.26	9.35	9.38
0.5	441.62	467.10	470.88	460.86

Table 4.4 Effect of Transients in CSMA/CD

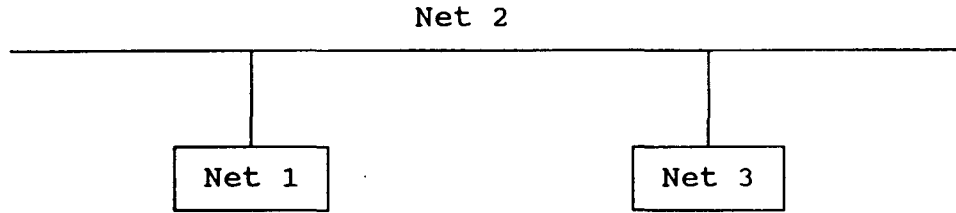


Fig 4.9 : Three Interconnected Networks - case 1.

Network	MAC	Transmission Speed	Offered Load	Packet Delay( $\mu$ sec)		
	Protocol			Net 1	Net2	Net3
Net1	Token Ring	4 Mbps	0.574	768.67	1526.90	3022.87
Net2	CSMA/CD	10 Mbs	0.406	1721.23	255.65	1750.80
Net 3	Token Ring	4 Mbs	0.574	2986.50	1529.06	749.92

Table 4.5 Three Interconnected Network - Case 1A

Network	MAC	Transmission Speed	Offered Load	Packet Delay( $\mu$ sec)		
	Protocol			Net1	Net2	Net3
Net1	Token Ring	4 Mbps	0.652	908	68320	70929
Net2	CSMA/CD	10 Mbs	0.751	28471	25984	28596
Net 3	Token Ring	4 Mbs	0.652	55281	52802	9052

Table 4.6 Three Interconnected Networks-Case 1B

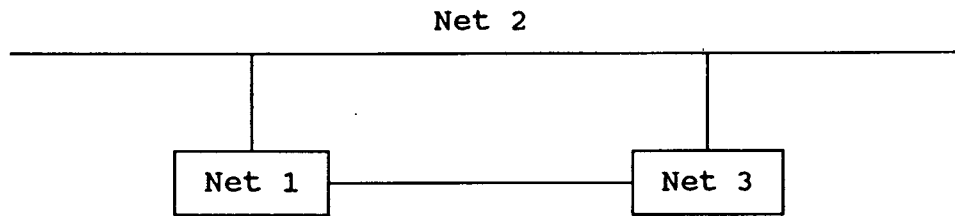


Fig 4.10 : Three Interconnected Networks - case 2.

Network	MAC	Transmission	Offered	Packet Delay( $\mu$ sec)		
	Protocol	Speed	Load	Net1	Net2	Net3
Net1	Token Ring	4 Mbps	0.624	811.57	1490.21	2340.30
Net2	CSMA/CD	10 Mbs	0.362	1535.50	183.99	1530.17
Net3	Token Ring	4 Mbs	0.624	2346.21	1492.24	792.30

Table 4.7: Three Interconnected Network - Case2

---

MAC Protocol	Number of Nodes	Number of Priorities	Time (in min)		
			$\sigma=0.1$	$\sigma=0.5$	$\sigma=0.9$
Token Ring	100	-	84.17	46.08	11.17
Ring/Bus without Priority	100	-	72.00	42.15	9.80
CSMA/CD	100	-	23.04	21.45	42.00
IEEE 802.5 Token Ring	10	8	3.42	3.38	3.07
Ring/Bus with Priority	10	8	4.03	4.03	3.34
Ring/Bus with Priority	100	2	77.27	66.23	55.20

---

Table 4.8: CPU Time Requirements on VAX/VMS System

## APPENDIX

Following are the abbreviations used in the output listing produced by SIMNET.

NN : Number of Nodes  
PARE : Packet Arrival Rate in packets/sec  
MPL : Mean Packet Length in Kilobits  
AWT : Average Waiting Time in microseconds  
VARIANCE : Variance of the AWT  
PACNT : Packet Arrival Count  
MPAN : Maximum Packets At Node  
(Over the simulation period)  
UCL : Upper Confidence Limite  
LCL : Lower Confidence Limit  
CI : Confidence Interval  
APAB : Average Packets At Bridge  
MPAB : Maximum Packets AT Bridge  
(over the simulation period).

# CHAPTER 5

## CONCLUSIONS

A general purpose simulator SIMNET for the performance simulation of individual and interconnected networks has been developed. The event scheduling approach has been used to model various MAC protocols. The simulator software has been written in C. The simulator developed is completely portable. It has been run successfully on VAX/VMS. The salient features of SIMNET are summarized below.

(1) SIMNET can simulate individual networks as well as a system of interconnected networks. A system may have up to 10 component networks. The number of nodes per network is limited to 100. The choice of these limits is not completely arbitrary considering the requirements they put on memory and CPU time. However, simulations of fairly complex interconnected networks are possible within these limits.

(2) The individual networks may have any of the following MAC protocols.

- (a) Simple Token Ring
- (b) Ring/Bus without priority
- (c) IEEE 802.5 Token Ring
- (d) Ring/Bus with Priority
- (e) CSMA/CD (IEEE 802.5 standards)

3. Each of the four ring protocols may have any of the following service disciplines.

(a) Exhaustive

(b) Limited-K service,  $K \geq 1$

For the IEEE 802.5 Token Ring one additional service discipline i.e., Maximum Service Time Fixed may be specified.

(4) The individual networks may have different transmission speeds, different size packet overheads and different service disciplines.

(5) Each priority class at each of the nodes may have different packet arrival rates.

(6) All the nodes of a network other than the bridges/routers have the same packet length distribution. The packet length may be constant or exponentially distributed. Different priority classes at each node may have different mean packet lengths.

(7) The nodes may be placed anywhere as per the user's specifications or at uniformly distributed points on the communication medium.

(8) Any pattern of interconnections between the component networks may be specified. However, multiple bridges/routers between two networks are not allowed.

(9) If multiple paths exist between the two networks which are not directly connected then either the minimum hop path or any other path may be chosen. In the latter case, all the networks on the path follow the same path in one direction to send packets to the other networks on the path.

(10) The traffic components flowing from the one network to another are specified by the user.

(11) The simulation length is specified by the user. It is specified in terms of average number of packets per node to simulate.

(12) On the completion of the simulation the following information is provided.

(a) Mean Packet Delay for each priority class at each node.

(b) Maximum buffer size (over the simulation period) requirement at each node.

(c) The Packet Delay averaged over all the nodes (excluding the bridges/routers) for each priority class and 90% confidence interval for this delay.

(d) Mean Packet Delay and 90% confidence interval at every bridge/router.

(e) Total Packet Delay experienced by a packet in going from network i to network j. This delay is provided for every network pair.

(13) A simple, user-friendly interface has been provided. Users can enter data from the keyboard in an interactive manner. Data entered through keyboard can be stored in a file for subsequent use. The input data file thus created has useful comments in it which allow the user to make changes in the input file itself. Every data is validated before starting the simulation. Outputs can optionally be stored in a file.



## Suggestions For Improvement And Future Work

This work done in thesis can be improved in three ways. The method of statistics collection can be improved using one of the methods of output data collection described in the Appendix of Chapter 2. The use of a suitable output data collection method will give tighter confidence intervals with smaller simulation run lengths. Two methods namely the method of batch means and the method of regenerative cycles look promising. The method of batch means requires efforts to find a suitable batch size. The method of regenerative cycles requires identification of suitable regeneration points of the process and proper estimators for the mean packet delay.

The abstract model used in simulation assumes Poisson arrivals but the actual traffic characteristics of the network traffic are very complex and may be poorly represented by the Poisson process. The actual traffic characteristics depend both on the protocols running over the network layer and the nature of the application running over these layers. A better model such as a batch Poisson process (where each Poisson arrival is a cluster of one or more packets and the number of packets in each cluster may be geometrically or uniformly distributed) may be used. This model can easily be implemented in the simulation model used by me with minor modifications. The initial choice of Poisson arrival model is justified for the purpose of validating the simulation results as the same assumption is used by various analytical models. The simple model of the bridges/routers used by me can be improved to model higher layer gateways.

## REFERENCES

- [1] D. Bertsekas, R. Gallager, "Data Networks", Prentice-Hall, 1989
- [2] ANSI/IEEE Standards 802.5, "Token Ring Access Methods and Physical Layer Specifications", IEEE Publications.
- [3] ANSI/IEEE Standards 802.5, "Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications", IEEE Publications.
- [4] S.K. Bose, K.R. Srivathasan, "A System Using Dynamic bus Reconfiguration for Fast and Efficient Contention-Free Access in LANs", Proc. of the 10th ICC, New Delhi, 1990.
- [5] S.K. Bose, "A Hybrid Ring/Bus Approach for Fast Contention Free Access", To appear in Proc. ITC-13, Copenhagen.
- [6] A.S. Tanenbaum, "Computer Networks", Prentice-Hall, 1988.
- [7] V.S. Frost, W.W. Larue, K.S. Shanmugan, "Efficient Techniques for the Simulation of Computer Communication Networks", IEEE J. Select. Areas Commn., Jan 1988.
- [8] M.J. Ferguson, Y.J. Aminetzab, "Exact Results for Nonsymmetric Token Ring System", IEEE Trans. Commn., May 1985.
- [9] D. Sarkar, W.I. Zangwill, "Expected Waiting Time for Nonsymmetric Cyclic Queueing Systems - Exact Results and Application", Management Science, Dec. 1989.
- [10] J. Gianini, D. Manfield, "Performance Analysis of the 802.5 Token Ring Standard Access Mechanism", Proc. ITC-12, Torino, July 1988.

- [11] S.S. Lam, "Carrier Sense Multiple Access Protocol for Local Networks", Computer Networks, Feb. 1980.
- [12] F.A. Tobagi, V.B. Hunt, "Performance Analysis of CSMA/CD", Computer Networks, Oct/Nov. 1980.
- [13] M. Murata, H. Takagi, "Performance of Token Ring Networks with Finite Capacity Bridge", TRL Tech. Rep, TR87- 0027, IBM Tokyo Res. Lab, May 1987.
- [14] G.M. Exley, L.F. Merakos, "Throughput-Delay Performance of Interconnected CSMA Local Area Networks", IEEE J. Select. Areas Commn. Dec. 1987.
- [15] I. Stavrakakis, D. Kazakos, "On the Approximation of the Output Process of Multiuser Random-Access Communication Networks", IEEE Trans. Commn. Feb. 1990.
- [16] G.S. Fishman, "Concepts and Methods in Discrete Event Digital Simulation", John Wiley 1973.
- [17] W.M. McCormack, R.G. Sargent, "Comparison of Future Event Set Algorithms for Simulations of Closed Queueing Networks", "Current Issues in Computer Simulation", Academic Press, 1979.
- [18] N. Deo, "System Simulation with Digital Computer", Prentice-Hall, 1987.
- [19] IMSL Library Manual
- [20] K. Pawlikowski, "Steady-State Simulation of Queueing Process: A Survey of Problems and Solutions", ACM Computer Surveys, June 1990.
- [21] S.K. Bose, S.K. Dalal, K.R. Srivathsan, "Improving the Priority Service Provided in a Token Ring with the New Ring/Bus Protocol", submitted for IECON' 91, Oct. 1991, Kobe, Japan.