The Impact of Modeling Abstraction on Network Simulation

Gustavo Saurez
Victor S. Frost

TISL Technical Report TISL-8670-4

Prepared for:
COMDISCO Systems, Inc.
1310 Wakarusa Drive Stuie A
Lawrence, Kansas 66049

and

Kansas Technology Enterprise Corporation (KTEC)
112 W 6th, Suite 400
Topeka, Kansas 66603

November 1990

Telecommunications and Information Sciences Laboratory
The University of Kansas Center for Research, Inc.
2291 Irving Hill Drive Lawrence, Kansas 66045

ABSTRACT

Communication networks operate according to complex distributed algorithms. Obtaining predictions of network performance is thus a difficult task. Often, analytic approaches are not applicable. When resorting to simulation there is the issue of the level model abstraction. That is, at what level of detail should the network and its protocols be modeled. Clearly, an overly detailed model will be useless as it does not provide accurate performance estimates in a reasonable time. On the other hand, an overly simplified model may miss important characteristics and thus not provide accurate performance estimates. The purpose of this work is to show the impact of different levels of modeling abstraction on the prediction of Ethernet performance. The accuracy of the predictions versus execution time will be given for various levels of modeling detail. The intent will be to stimulate discussions on the importance of modeling abstractions to the practical application of tools for the computer aided design and analysis of networks.

CHAPTER 1: Introduction

The sharing of computer resources has acquired real importance in today's business environment. Adequate computer networks configurations are critical for efficient and reliable data exchanges. The design and performance analysis of computer networks require accurate modeling and simulation of a specific network structure. Because of the complex nature of some communication networks, analytical analysis are not always appropriate.

Depending on the level of detail utilized for the model implementation, network simulation models may involve different levels of structural complexity. This study is concerned with the level of modeling abstraction used to analyze networks. Models with low abstraction levels may require prolonged simulation times which many times is not desired. On the contrary, extremely simplified models may fail in providing accurate performance estimates.

Presently, there are three well defined standards for local computer networks technologies: Token Bus, Token Ring, and Random Access. The largest number of existing computer networks utilize the CSMA/CD technology. The present project is aimed to the study of the impact of modeling abstraction on CSMA/CD systems. The next sections give a general description of the CSMA/CD technology and the methodology of the project. Further, a brief description of the models and tools utilized for this study is given.

1.1 The CSMA/CD Technology

The Carrier Sense Multiple Access with Collision Detection (CSMA/CD) technique is at the present one of the most widely used for local computer networks. Its distributed nature, makes it well suited for local area networks where flexibility and simplicity of operation are important. The CSMA/CD topology is a bus structure. This gives a relatively wide transmission rate e.g.

10Mbps. In addition, the performance of CSMA/CD is inversely proportional to the end-to-end propagation delay in the system.

Since the stations can sense signals on the common channel, the number of conflicts is reduced because a packet transmission is made when there is no activity in the channel. Before a station attempts to transmit a packet, it checks the signal level of the channel to identify the presence of a current transmission. Each time a station finds a busy channel, it can attempt to retransmit (with probability p, which is equal to one for the Ethernet scheme) when the line becomes idle again or it can wait a random time interval before sensing the channel for retransmissions. Following a collision the station selects a random time for retransmission according to a backoff algorithm

Different backoff techniques can be utilized to manage collisions in the system. The standard backoff technique uses the binary exponential backoff algorithm[1]. In this algorithm, the waiting time before retransmitting a packet is uniformly distributed between 0 and 2^n, where n is the number of collisions suffered by the packet. When n reaches ten collisions the range of the distribution is kept from 0 to 2^{10}. If more than sixteen consecutive collisions occur, the packet is discarded. Another backoff technique sets the retransmission of the collided packet after an exponentially distributed random period of time. There is no time adjustment for packets with successive collisions.

1.2 Overview of the Project

1.2.1 General Description

This project considers several proposed performance models used to analyze local area networks which use CSMA/CD technology. These models range from simple models which consist of a single queue to more complicated models which consist of several complex subroutines. These models were implemented using four simulation tools. An explanation of each model and its implementation is given in section 2. The simulation tools used to implement these model are mentioned in section 1.4. Section 2 also describes these tools with respect to the models implemented on each of them.

The simulations executed for each model were intended to model the same system. The parameters used on each model simulation were set as close as possible to each other. Further, the simulation length was the same for all the models. All these aspects are explained in detail in section 3. Also, the results obtained from comparing the different CSMA/CD models are totally described in section 3.

1.2.2 Goal

The main objective of this project is to evaluate the accuracy of several CSMA/CD performance prediction models with respect to a selected base system. The base model has been validated with measurements. The level of detail in each CSMA/CD model is different for each case considered here. Thus, this project studies the impact of different levels of modeling abstraction on the simulation of CSMA/CD systems. From this study will be possible to determine if a CSMA/CD system can be accurately simulated with a simple model.

The quality of each model is related to mean packet transmission delay and simulation execution time. The packet transmission delay results obtained from the models are compared to the base system to identify the accurate CSMA/CD models. The simulation execution time is obtained as an optional comparison parameter in order to have a tradeoff between accuracy and speed.

This project also intends to observe the behavior of each CSMA/CD model for various loads and for different arrival traffic and packet length distributions.

1.3 CSMA/CD Models Implemented

All the models utilized in this study are intended to provide a method to analyze the behavior of a CSMA/CD system under certain conditions, i.e. topology and traffic. In this project six CSMA/CD simulation models were related to five levels of abstraction. These levels an their correspondent simulation model are presented in Table 1.1

Level of Abstraction	Model Utilized
1 (lowest)	Full detailed simulation of all propagation and protocol functions (GLAS)
2	State Transition model with Binary Exponential Backoff, SIMLAN™
3	State Transition model with Exponential Backoff
4	Non-preemptive Priority Queue model
5	Single Queue model with varying service time

TABLE 1.1
Levels of abstraction and respective models

The lowest level is the base model. This model (GLAS) can be configured by the user. The CSMA/CD model only requires the user to set the parameters of the LAN[5],[6]. A more detailed explanation of the GLAS model is presented in section 2.3. In this level of abstraction the simulation follows closely all the protocols and propagation functions that rule a CSMA/CD system.

Another user configurable tool utilized to model a CSMA/CD system is SIMLAN™. As the GLAS model, SIMLAN™ has preconfigured simulation systems that require the user to set parameters.

The models included in the second and third levels of abstraction (Table 1.1) are based on the state of the bus in a CSMA/CD system. The bus state of idle, busy, or collision implies different actions. The difference between them is in the backoff algorithm. One uses an exponential backoff and the other a binary exponential backoff algorithm. These models were called: State Transition model with exponential backoff and State Transition model with binary exponential backoff respectively. Sections 2.2 and 2.3 explain these two models and their implementation. An explanation can also be found in [4].

The model with the fourth level of abstraction is based on a simple queueing system. It consists of a non-preemptive priority queue with two kinds of arrivals: collision packets (highest priority) and successfully transmitted packets.

The model with the highest level of abstraction is also based on a simple queueing system. It is a single queue with only one kind of arrival packets. The service time of the queue varies with the load of the system.

A more detailed description of the last two models is presented in [2],[3] respectively, and also in section 2.

An analytical model was also considered in this project. This model is a mathematical expression that describes the Load-Delay characteristic of a CSMA/CD model as a function of propagation delay, packet transmission time, and load[1]. Section 2.5 explains this model in more detail.

The GLAS model is used as the base system in this project. The justification for this selection is that this model has been validated with measurements taken from an operating system[7]. Figure 1.1[7] shows the experimental setup utilized to obtain measurements of two way end-to-end delay. Furthermore, Figures 1.2, 1.3, and 1.4 show the results obtained in [7] for the MAC access delay for different packet sizes.

From these figures it can be seen that the packet delay is almost the same for both the simulation and experimental cases.

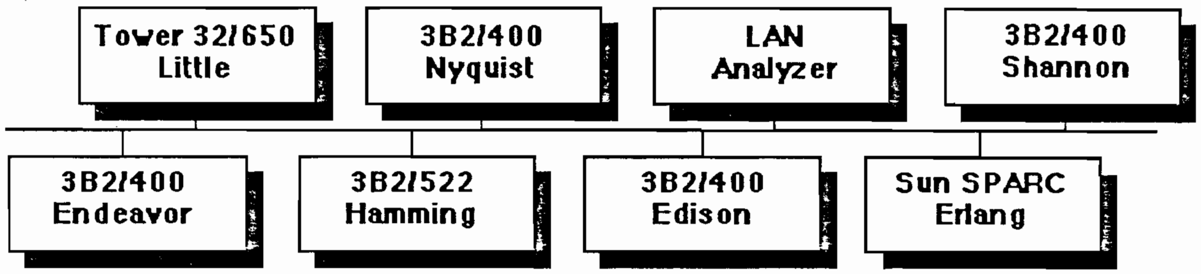


FIGURE 1.1
Experimental Setup

1.4 Simulation Tools Utilized

The CSMA/CD models described above were implemented utilizing four simulation tools or programs: BONeS™ [8], Q+™[4], SIMLAN™[6], and GLAS[5]. The reasons of using different simulation tools to implement the different modeling abstractions are:

1) To evaluate the cost-effectiveness (simulation time and easy of implementation) of the simulation tools along with the most accurate modeling abstraction.
2) To compare the simulation execution time of each modeling abstraction for the different simulation tools.
3) To determine the implementation difficulty of each modeling abstraction for the different simulation tools.

BONeS is a Trademark of COMDISCO Systems Inc.
Q+ is a trademark of AT&T Bell Telephone Laboratories
SIMLAN is a Trademark of CACI Products Company

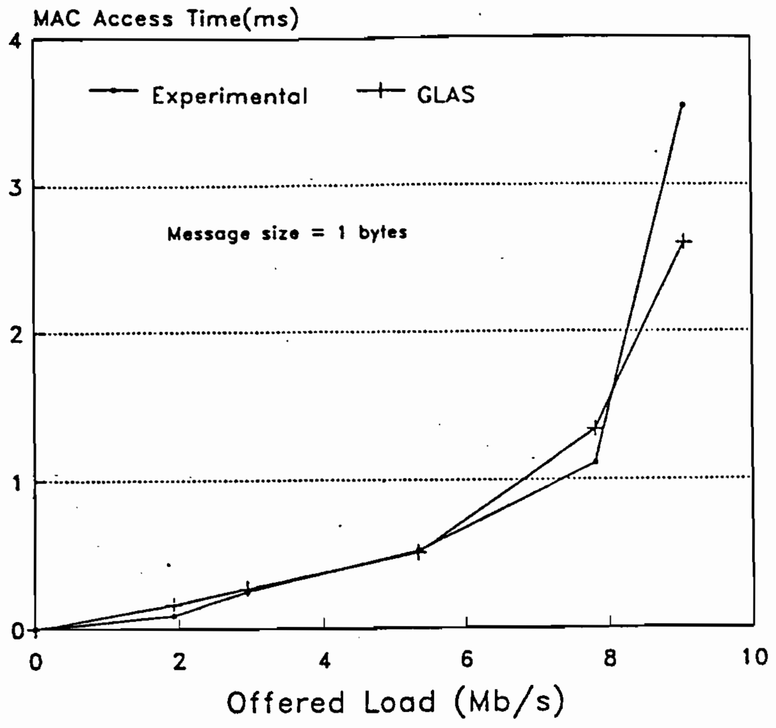


FIGURE 1.3
MAC Access time vs. offered Load for a message length of 1 bit

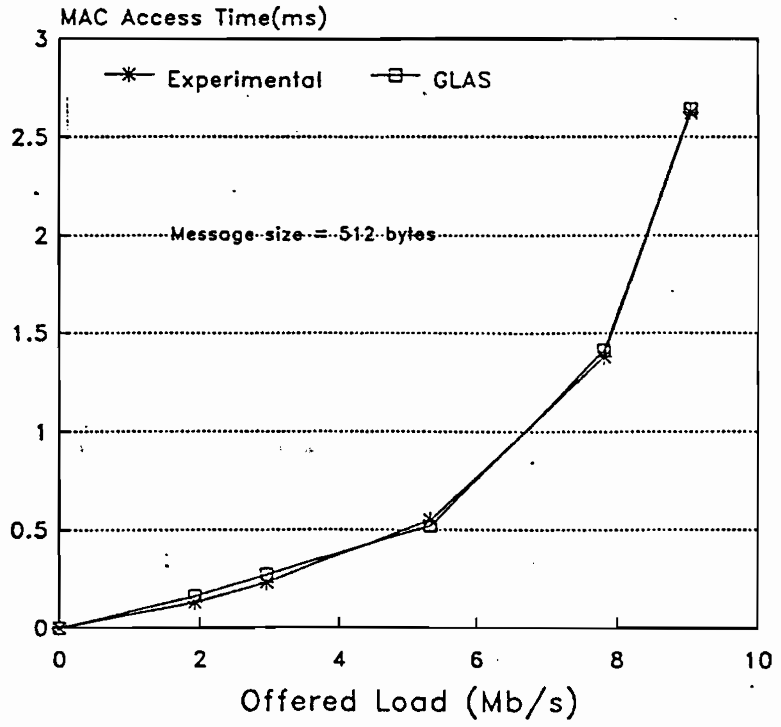


FIGURE 1.4
MAC Access time vs. offered Load for a packet length of 512 bit

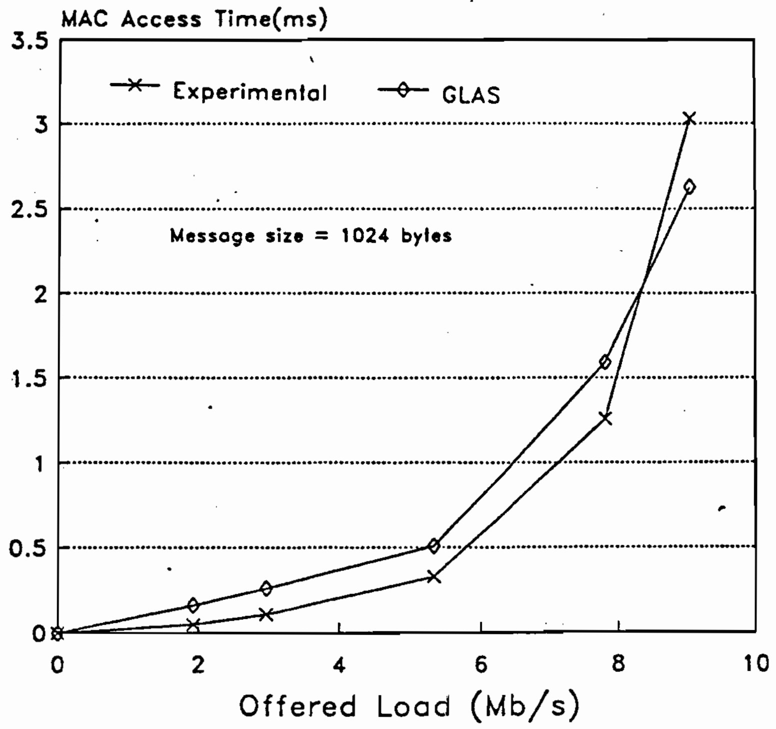


FIGURE 1.5
MAC Access time vs. offered Load for a message length of 1024 bits

BONeS™ and Q+™ are general purpose simulation tools, that is, the user can build models of general networks. BONeS™ is based on a data flow block diagram approach. The user gives a function to each block and interconnects them. Q+™ is based on the extended queuing network paradigm. The user has to specify the queue parameters and their topology. Since BONeS™ and Q+™ are very flexible, the single queue model and the priority queue model were both built using BONeS™ and Q+™. The same with the State Transition model with exponential backoff. The State Transition model with binary exponential backoff was build only on BONeS™ because of the difficulty in modeling the binary exponential backoff algorithm using Q+™ primitives. The results obtained from both tools should be very similar for comparable models.

The other two programs, GLAS and SIMLAN™, provide the user with a system focused on LAN performance analysis. The user has to specify parameters for these systems. SIMLAN™ allows interconnection among networks.

BONeS™ and SIMLAN™ were executed on a Sun Sparcstation-1+. While Q+™ and GLAS were run on a AT&T 3B2/522. This gives a machine dependent difference in the execution time for the simulations. A benchmark was executed on both machines to obtain a factor to convert times from one machine to another.

The implementation of the CSMA/CD models detailed in Section 1.3 is explained in Section 2. Appendix A gives a comparison of the different features of BONeS™, Q+™, and SIMLAN™.

1.5 Summary of Results

The study was conducted for two types of arrival traffic: exponentially distributed packet arrivals and for a general distribution for interarrival times. The latter distribution allows to observe the behavior of the model for an arrival distribution which does not have an analytical representation. The Load-Delay characteristic from the GLAS model for both types of arrival traffic and exponential packets was compared to the Load-Delay curve of the other models. Based on this comparison, the Single Queue model and the State Transition model with binary exponential backoff were the most accurate models for exponential interarrival times. On the other hand, the SIMLAN™ model was the most accurate model for a general arrival traffic distribution.

The characteristic curve presented for all the models were very similar (for both arrival distributions) for low loads (less than 70%), the differences became bigger for higher loads.

In addition, a comparison was also made for simulations with exponential arrivals and fixed packet lengths. The State Transition model with binary exponential backoff, the Single Queue model and the SIMLAN model were the most accurate for these distributions.

Results were also obtained for fixed packet lengths and exponential arrivals. The Single Queue Model and the State Transition model were the most accurate.

The BONeS™ models had the lowest execution time when comparable models where related. The Single Queue model was the fastest model to execute the simulation as expected.

These results suggest that the Single Queue model, the State Transition model with binary exponential backoff, and the SIMLAN™ models are an alternative to simulate accurately a CSMA/CD system as the detailed GLAS model. Section 3 explains in more detail the results obtained in this study.

CHAPTER 2: Models Description and Implementation

This section describes the different modeling abstractions utilized for this study. The modeling framework and the approach applied in the implementation of each model in the different simulation tools is described in detail. The description is made beginning with the highest level of abstraction (Single Queue model). The last model described is the GLAS model (lowest abstraction level). The Single Queue, Priority Queue, and State Transition models were implemented on BONeS™ and Q+. A separate description of the implementation on each tool is given.

2.1 Single Queue Model

2.1.1 Description

This model[2] is basically a single queue and a server which service time changes with the load offered to the system. Figure 2.1 shows an illustration of this model.

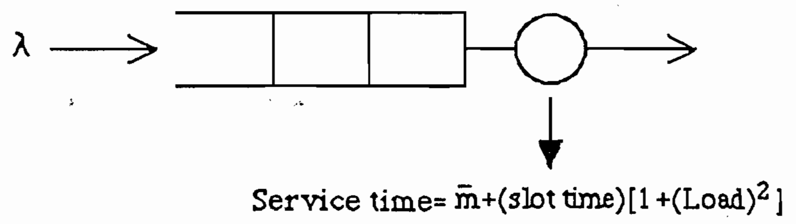


FIGURE 2.1
Single Queue Model

The service rate of this model is adjusted to include collision and backoff delays. This is done by increasing the mean packet transmission time $(\bar{m})$ by T_{add}, where:

$$T_{add} = (\text{slot time})[1+(\text{offered load})^2] \qquad (2.1)$$

In the latter expression, slot time is a unit of time used in the backoff algorithm (usually 51.2µs) and offered load is the fraction of bandwidth used by all transmissions requested.

In [2] this model is described as a LIFO M/G/1 system with slightly increased service time. Expression 2.1 was obtained empirically[2]. The added service time is independent of the packet length, from this the following moments for the transmission time can be obtained[2]:

$$\bar{m}^2 = (\bar{m}+T_{add})^2 + T_{add}^2 + (\bar{m} * C_b)^2$$

$$\bar{m}^3 = 6 * (\bar{m} + T_{add}) * T_{add}^2 + (1 + C_b^2) * \bar{m}^2 * (3 * T_{add} + (1+2C_b^2) * \bar{m})$$

From these moments, the average waiting time can be calculated [2],[10]. In [2], the analytical results obtained from the single queue model matched to results produced by a simulation.

A modification of the LIFO M/G/1 system is presented in [9]. They present a FIFO queue with exponential service. The mean service time was also increased by the same expression (2.1). The product-form equivalence was assumed as an approximation in [9]. The analytical results of this model were also compared to the results obtained from a simulation programs and they were very similar [9].

The two single queue models described in this section were validated and reported to be useful to analyze CSMA/CD systems. However, in this project only one of these models is studied since they both have a similar structure. The single queue model based on the FIFO queue with exponential service is the one considered in this project.

2.1.2 Single Queue Model BONeS™ Implementation

2.1.2.1 Data structure

Figure 2.2 shows the BONeS' data structure hierarchy. This project is only concerned with class CSMA/CD and its subclasses: State Transition Model (STM), Priority Queue Model (PQM), and Single Queue Model (SQM).

The data structure for class CSMA/CD is shown in Figure 2.3. The first field is inherited from the superclass and is used for identification purposes. The other three fields store the basic information needed to transmit a packet and obtain its transmission delay. Packet Length stores the packet length that can be used for general purposes.

The data structure used for the Single Queue module is shown in figure 2.4. This data structure inherits the fields from its superclass. The added field, Service Time, is used to store the service time of each packet. This service time changes with the load offered to the system.

2.1.2.2 Model implementation

The highest level module of the single queue model consists of three submodules: a traffic generator, the single queue module, and a receiver. Figure 2.5 shows these modules. The traffic generator creates and sends packets to the queue. It simulates an infinite number of nodes. One version of this generator sends packets with an exponentially distributed interarrival times, the other version does it with a general distribution.

The single queue model consists of a FIFO queue with one server. The service time can be exponential, uniform, or fixed. The expanded single queue module is shown in Figure 2.6. The receiver is just a module that is used to obtain the exact transmission delay of each packet from the difference of the time when the packet was received to the time when the packet was transmitted.

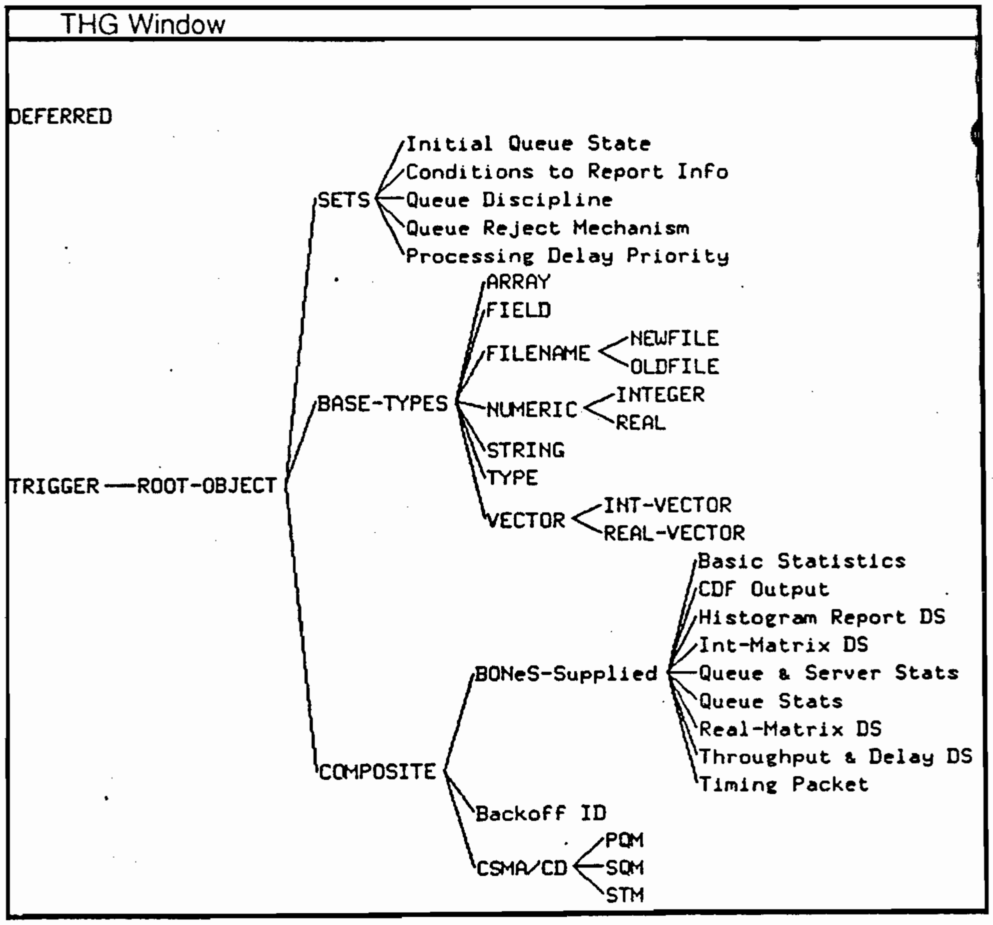
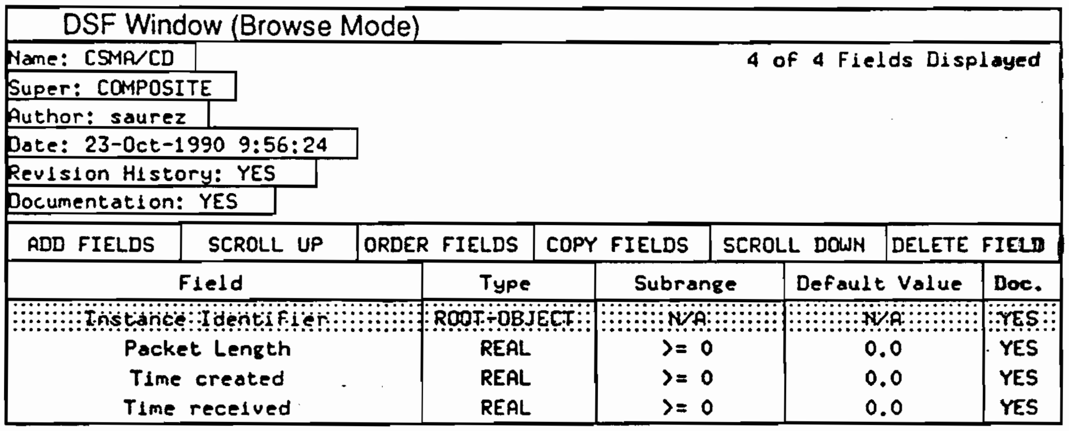


FIGURE 2.2
BONeS™ Data Structure Hierarchy



DSF Window (Browse Mode)

Name: CSMA/CD

4 of 4 Fields Displayed

Super: COMPOSITE

Author: saurez

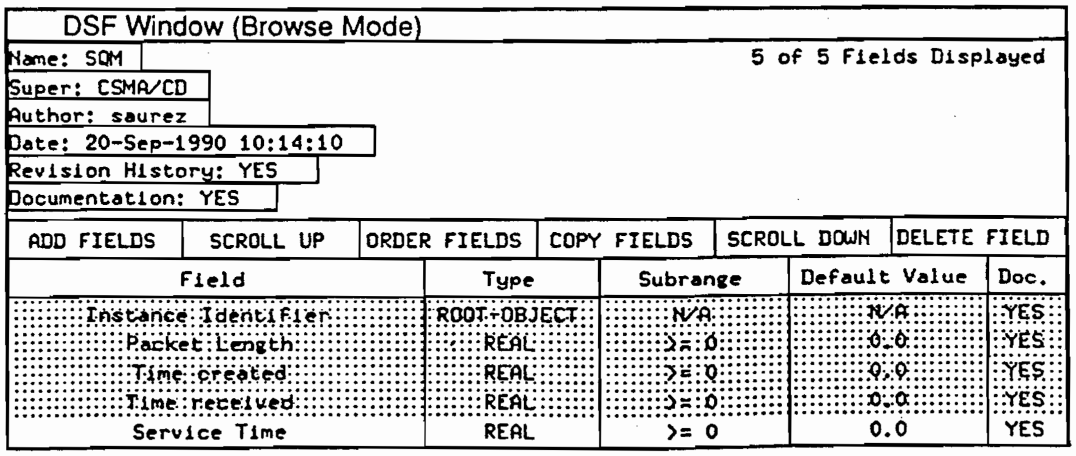
Date: 23-Oct-1990 9:56:24

Revision History: YES

Documentation: YES

Field	Type	Subrange	Default Value	Doc.
Instance Identifier	ROOT+OBJECT	N/A	N/A	YES
Packet Length	REAL	>= 0	0.0	YES
Time created	REAL	>= 0	0.0	YES
Time received	REAL	>= 0	0.0	YES

FIGURE 2.3
Data Structure for Class CSMACD



DSF Window (Browse Mode)

Name: SQM

5 of 5 Fields Displayed

Super: CSMA/CD

Author: saurez

Date: 20-Sep-1990 10:14:10

Revision History: YES

Documentation: YES

Field	Type	Subrange	Default Value	Doc.
Instance Identifier	ROOT+OBJECT	N/A	N/A	YES
Packet Length	REAL	>= 0	0.0	YES
Time created	REAL	>= 0	0.0	YES
Time received	REAL	>= 0	0.0	YES
Service Time	REAL	>= 0	0.0	YES

FIGURE 2.4
BONeS™ Single Queue Model Data Structure

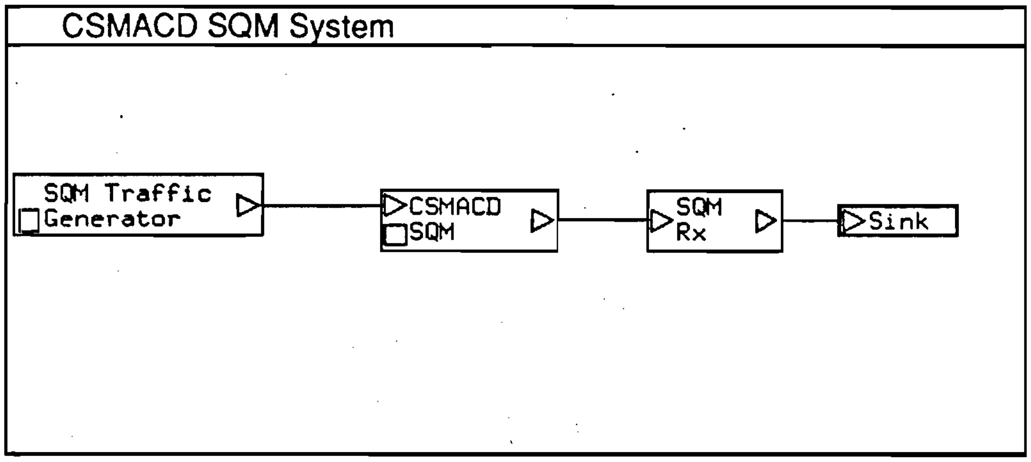


FIGURE 2.5
BONeS™ Single Queue model
Simulation System

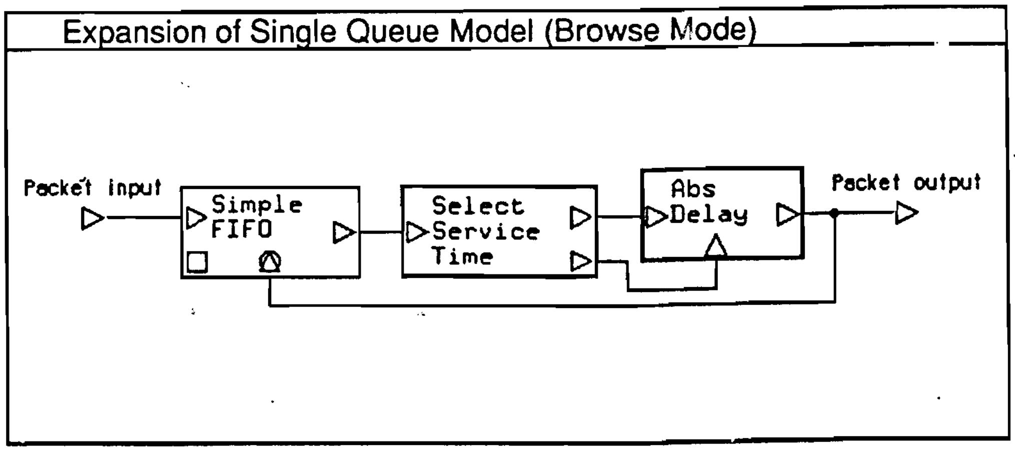


FIGURE 2.6
BONeS' Single Queue model

This information is taken from the packet's data structure. The structure of the other submodules of the Single Queue model is presented in Appendix B.

2.1.3 Single Queue Model Q+™ Implementation

The Single Queue model is very simple to implement in Q+™. The implementation of this model is shown in Figure 2.7. It only consists of one node or queue. Only one class of transaction is specified in this queue, which was called **pkt**.

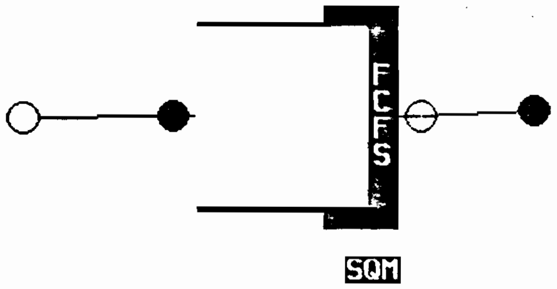


FIGURE 2.7
Q+™ Single Queue model

The packet arrival and service time distributions can be set to any distribution. The arrival and service rates are specified directly in the queue before running the simulation. Both rates vary with the load offered to the system.

From Figure 2.7, the normalized transmission delay is calculated by dividing the average waiting time from the statistics window by the mean packet transmission time (service time $\overline{m}$).

2.2 Priority Queue Model

2.2.1 Description

This model[3] consists of a nonpreemptive priority queue with one server. There are two kinds of arrival packets: collision packets and successfully transmitted packets. The former kind of packets has a higher priority than the latter. This model is illustrated in figure 2.8.

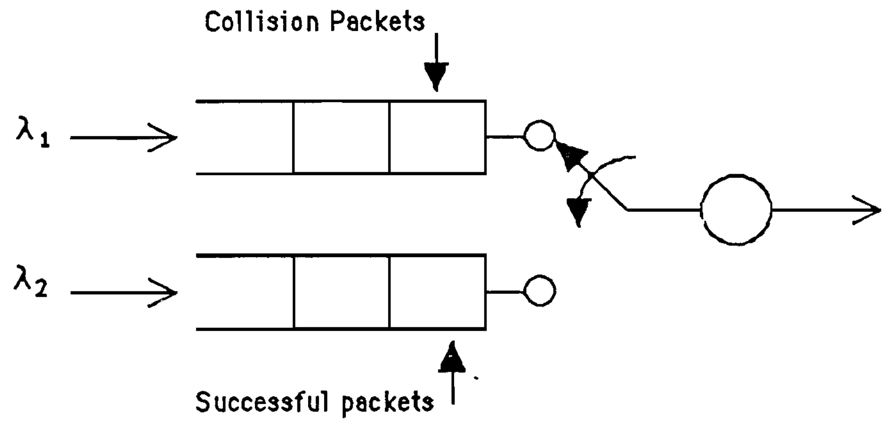


FIGURE 2.8
Priority Queue Model

The priority queue model describes the following characteristics of CSMA/CD[3]:

1) Because of the non-preemptive property, a station transmits completely once it acquires the first slot of transmission.
2) Because of the priority queueing property, no successful packet transmission occurs when there is a collision packet transmission.
3) Because the arrival rate of collision packets is maintained at (v-1) times that of data packets on average, the probability of having successful transmission in a contention slot is 1/v (v=e= 2.718).

An imbedded Markov chain is used to analyze this model. Given the number of messages in the queue at a particular departure epoch, the number of messages in the queue at future epochs depends only on new arrivals and not on past occupancies. Thus the succession of message states forms an imbedded Markov chain. A Markov chain is imbedded at the instants when there is a service completion and a consequent departure from the queue (transmission completion epoch). Each instant is at the end of a slot. Four terms are defined next:

k = the class of the packet (1 or 2).

n_{ik} = the number of class k packets in the system at the ith departure epoch.

a_{jk} = the number of class j packets arriving during the transmission of a class k packet.

$\lambda = \lambda_1 + \lambda_2 =$ total arrival rate.

$\hat{\rho} = \lambda_1 \bar{m}_1 + \lambda_2 \bar{m}_2$, where $\bar{m}_1$ and $\bar{m}_2$ are the mean transmission time for collision and successful packets respectively.

If the ith departing message leaves a non-empty queue with n_{i1} messages (class 1 packets), the state of the system at the next departure (the (i+1)st departure) is given by:

$$n_{i+1,1} = n_{i1} - 1 + a_{11}$$
$$n_{i+1,2} = n_{i2} + a_{21}$$

If the ith departing message leaves a non-empty queue with n_{i2} messages (only class 2 packets), the state of the system at the (i+1)th departure is given by:

$$n_{i+1,1} = a_{12}$$
$$n_{i+1,2} = n_{i2} - 1 + a_{22}$$

If the departure of the ith message leaves the queue empty, the (i+1) departing message arrived to an empty queue. Consequently this packets leaves the queue only with the packets that arrived during its service time. If the (i+1)st packet is a class 1 packet (k=1), which has a probability of λ_1/λ, then:

$$n_{i+1,1} = a_{11}$$
$$n_{i+1,2} = a_{21}$$

If the (i+1) packet is a class 2 packet (k=2), which has a probability of λ_2/λ the state is given by:

$$n_{i+1,1} = a_{12}$$
$$n_{i+1,2} = a_{22}$$

Three probabilities are defined next:

a. Probability of an empty queue = $P[n_{i1}=0 \text{ and } n_{i2}=0] = 1-\hat{\rho} = \mathbf{P_0}$

b. Probability of having class 1 packets = $P[n_{i1}>0] = \lambda_1\hat{\rho}/\lambda = \mathbf{P_1}$

c. Probability of having only class 2 packets = $P[n_{i1}=0 \text{ and } n_{i2}>0] = \lambda_2\hat{\rho}/\lambda = \mathbf{P_2}$

The two dimensional probability generation function is used to analyze the equations that have been developed so far. Thus for $n_{i+1,1}$ and $n_{i+1,2}$, the two dimensional probability generating function is given by:

$$\begin{aligned} E[Z_1^{n_{i+1,1}} Z_2^{n_{i+1,2}}] = P_0\{ & \left(\frac{\lambda_1}{\lambda}\right) E[Z_1^{a_{11}} Z_2^{a_{21}} \mid n_{i1} = n_{i2} = 0] + \\ & \left(\frac{\lambda_2}{\lambda}\right) E[Z_1^{a_{12}} Z_2^{a_{22}} \mid n_{i1} = n_{i2} = 0]\} + \\ & P_1 E[Z_1^{n_{i1}-1+a_{11}} Z_2^{n_{i2}+a_{21}} \mid n_{i1} > 0] + \\ & P_2 E[Z_1^{a_{12}} Z_2^{n_{i2}-1+a_{22}} \mid n_{i1} = 0, n_{i2} > 0] \end{aligned}$$

After differentiating the last expression with respect to Z_1 and Z_2, the next expression can be obtained:

$$\bar{n}_{22} = 1 + \frac{\lambda_2[\lambda_1\bar{m}_1^2 + \lambda_2\bar{m}_2^2]}{2\hat{\rho}(1-\lambda_1\bar{m}_1)(1-\lambda_1\bar{m}_1-\lambda_2\bar{m}_2)} \qquad (2.2)$$

The last expression represents the mean number of class 2 packets in the system given that one of them is beginning transmission. Then, by using Little's formula, the mean waiting time of of class 2 packets, $\bar{W}_2$, is given by:

$$\bar{W}_2 = \frac{\hat{\rho}}{\lambda_2}(\bar{n}_{22} - 1) \qquad (2.3)$$

For the CSMA/CD model the following terms are defined:

τ = end-to-end propagation delay of the channel

$\bar{m}_1 = \bar{m}_1 = k\tau$	(k= constant)
$\bar{m}_2 = \bar{m} + \tau$	($\bar{m}$= actual packet length)
$\lambda 1 = \lambda 2(v-1)$	

As the channel is slotted, a data packet entering into an empty system with very light traffic suffers a waiting time of half a slot on average. This factor is ignored for simplicity [3]. To make the correction, the next term is added to the mean waiting time of data packets[3]:

$$(1 - \lambda_1\bar{m}_1 - \lambda_2\bar{m}_2)\frac{\tau}{2}$$

If the packet length is made constant then $\bar{m} = m$ and $\bar{m}^2 = m^2$. The mean delay of data packets is the sum of the mean packet waiting time and the mean packet transmission time. The normalized delay is given by:

$$D_n = 1 + \frac{\rho_d[k^2\alpha^2(v-1) + (1+v)^2]}{2[1 - k\rho_d\alpha(v-1)]\{1 - \rho_d[1 + \alpha + k\alpha(v-1)]\}} + \{1 - \rho_d[1 + \alpha + k\alpha(v-1)]\}\frac{\alpha}{2} \qquad (2.4)$$

where:

$$\rho_d = \lambda_2 m \qquad \alpha = \frac{\tau}{m}$$

Since $\lambda_1\bar{m}_1 + \lambda_2\bar{m}_2 < 1$ the maximum data throughput is:

$$\rho_d < \frac{1}{1 + \alpha(k(v-1) + 1]}$$

For exponentially distributed packet lengths, the second moment of the mean length is obtained by:

Exponential distribution:

Mean= $\bar{m}$

Variance= $(\bar{m})^2 = \sigma_m^2 = \bar{m^2} - (\bar{m})^2$

then,

$$\bar{m^2} = 2*(\bar{m})^2$$

Following the same procedure used in earlier sections, and replacing $\bar{m}^2$ by $2*(\bar{m})^2$ in (2.1), the next formula for normalized mean data delay is obtained:

$$\bar{D}_n = 1 + \frac{\rho_d[k^2\alpha^2(v-1) + 2(1+v)^2]}{2[1 - k\rho_d\alpha(v-1)]\{1 - \rho_d[1 + \alpha + k\alpha(v-1)]\}} + \{1 - \rho_d[1 + \alpha + k\alpha(v-1)]\}\frac{\alpha}{2} \qquad (2.5)$$

where:

$$\rho_d = \lambda_2 \bar{m}_2 \qquad \alpha = \frac{\tau}{\bar{m}}$$

Equation 2.4 was validated in [3] by comparing its results for different loads and some values of α with the results obtained from simulation. Thus, the previous model was shown to be suitable for analyzing the CSMA/CD networks.

2.2.2 Priority Queue Model BONeS™ Implementation

2.2.2.1 Data structure

Figure 2.9 shows the data structure used for the Priority Queue model packets. The class field is used to identify the class or priority of the packet. Class equal to zero (lowest priority) represents successful packets, and class equal to one represents collision packets. The class value is set in the traffic generator.

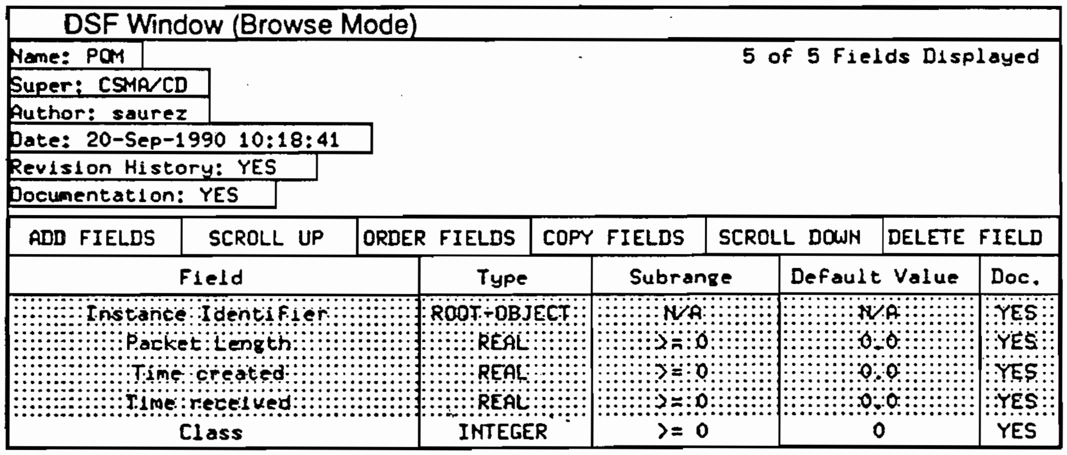
2.2.2.2 Model implementation

The top level module of the Priority Queue model consists of two traffic generators, a priority queue module and a receiver. Figure 2.10 shows the top level module. One traffic generator creates and sends the collision packets (class 1). The other one generates the successful packets (class 2).

The traffic generator was created in two versions: exponential interarrival times and general interarrival times. The receiver is used to obtain the transmission delay for each packet.

The priority queue module is shown in figure 2.11. This module consist mainly of a nonpreemptive priority queue and a server. Any class 1 packet is sent to the server (Abs. Delay) only if there is no class 2 packets in the queue.

Appendix B shows the rest of the submodules used to implement the Priority Queue model.



Name: PQM

Super: CSMA/CD

Author: saurez

Date: 20-Sep-1990 10:18:41

Revision History: YES

Documentation: YES

5 of 5 Fields Displayed

Field	Type	Subrange	Default Value	Doc.
Instance Identifier	ROOT-OBJECT	N/A	N/A	YES
Packet Length	REAL	>= 0	0.0	YES
Time created	REAL	>= 0	0.0	YES
Time received	REAL	>= 0	0.0	YES
Class	INTEGER	>= 0	0	YES

FIGURE 2.9
BONeS™ Priority Queue Model Data Structure

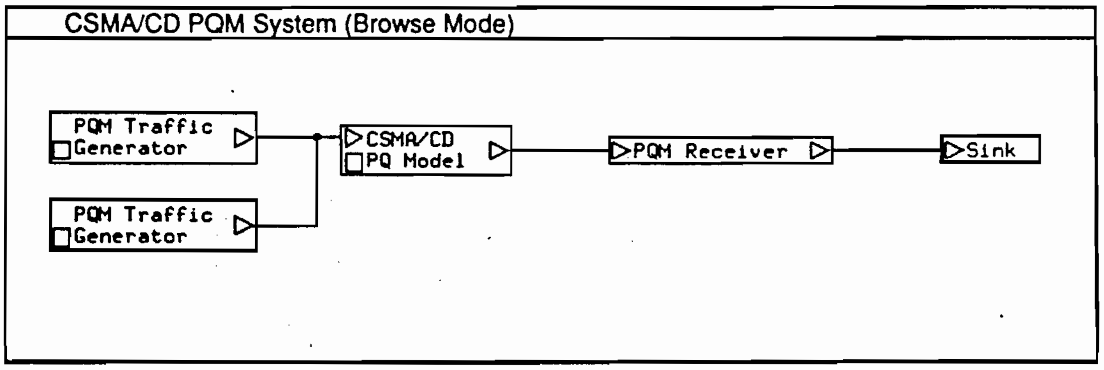


FIGURE 2.10
BONeS™ Priority Queue Model Simulation System

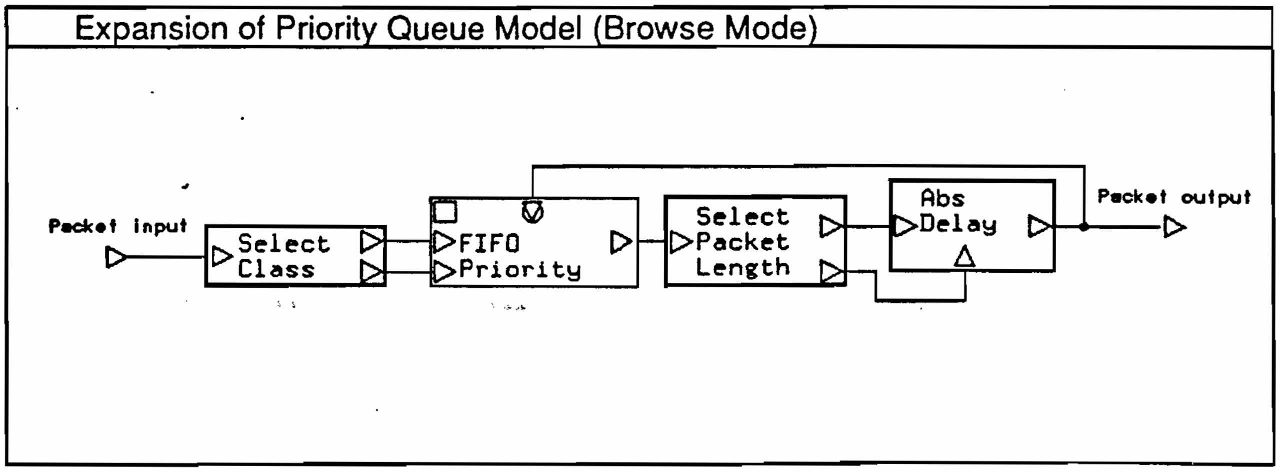


FIGURE 2.11
BONeS' Priority Queue Model

2.2.3 Priority Queue Model Q+™ Implementation

Figure 2.12 shows the queue used to implement the Priority Queue model on Q+™. This queue has two classes of transactions: **c0** (successful packets) and **c1** (collision packets). The latter has priority 0 (highest). The arrival and service distributions were set to exponential and uniform for this project. The arrival rate changes with the offered load for both classes. The two classes have different arrival and service rates. The normalized delay is obtain from the average waiting time for **c0**.

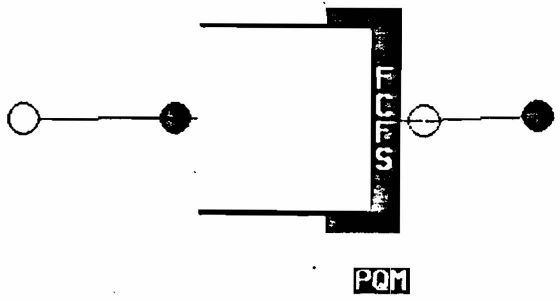


FIGURE 2.12
Q+™ Priority Queue Model

2.3 State Transition Models

2.3.1 Description

The models presented here follow [11]. These models are based on the state of the channel. They follow the protocol associated to the 1 persistent CSMA/CD technology. One of the models uses an exponential backoff and the other uses a binary exponential backoff.

The algorithm these models use assumes four states of the channel: **idle, t1** (collision vulnerable), **t2** (collision detection), and **tm** (busy or transmitting). The states diagram is presented in Figure 2.13.

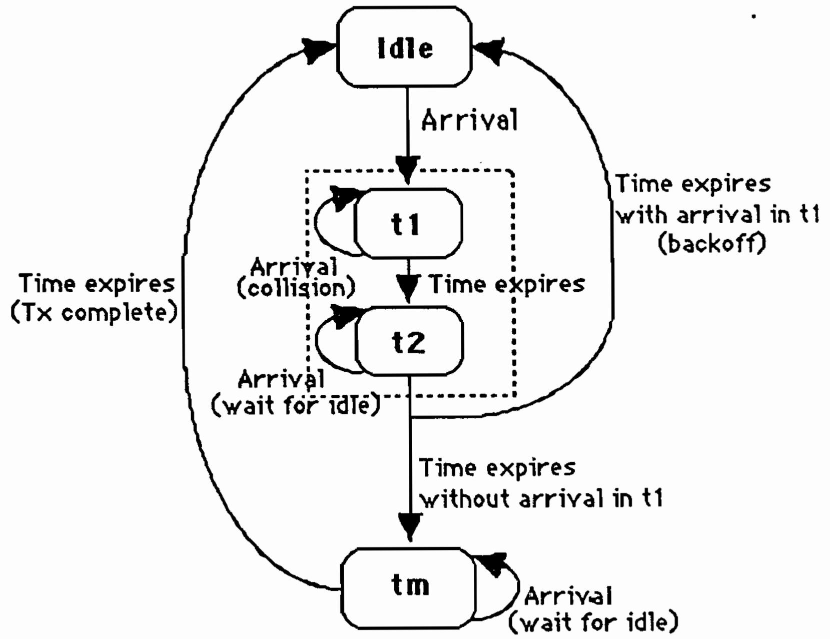

Idle
Arrival
t1
Arrival (collision)
Time expires
t2
Arrival (wait for idle)
Time expires with arrival in t1 (backoff)
Time expires (Tx complete)
Time expires without arrival in t1
tm
Arrival (wait for idle)

FIGURE 2.13
Diagram for the States Transition Models Algorithm

From the figure, if the channel is **idle**, any station can transmit and if a packet arrives it sets the state to **t1** that simulates the end-to-end propagation time. Collisions can occur only when the medium is in state **t1** (vulnerable time). If any other packet arrives during **t1**, the packets are backed off (put in a backoff queue) and they will try to transmit after a random time interval. This time depends on the backoff technique utilized.

When **t1** expires the channel state is changed to **t2** which simulates the maximum time to detect a collision. Any other packet that arrives during state **t2** finds the channel busy and it is sent to a wait queue. The packets in the wait queue will transmit again whenever the channel becomes **idle**. When the state **t2** ends, the backoff queue is checked to look for collisions. If any collisions occurred, the state is set to **idle**. In addition, all the packets in the backoff queue are given a random period of time to wait before they try to transmit again. If there were no collisions the state is changed to **tm** which simulates the packet transmission time. Any packet arriving during this state finds the channel busy and is also sent to the wait queue. When **tm** expires, the channel becomes **idle** again.

2.3.2 State Transition Models BONeS™ Implementation

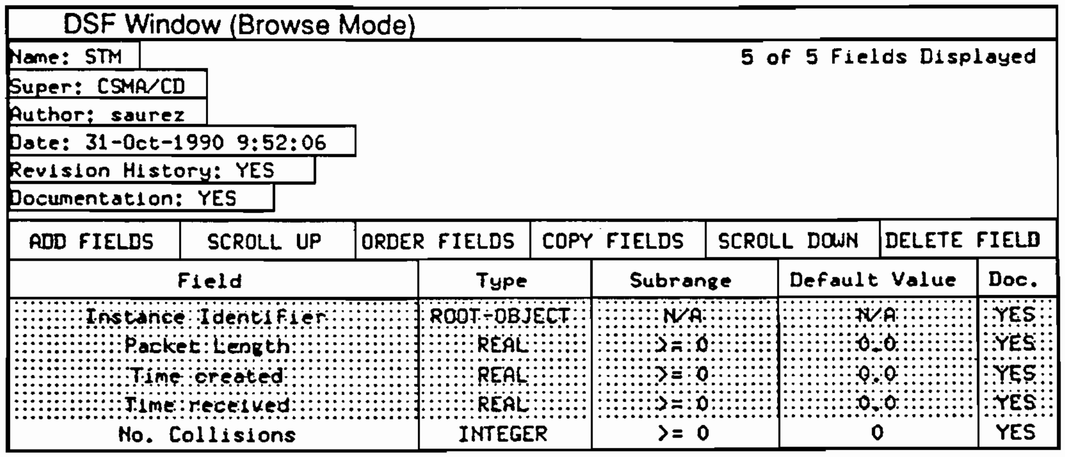
2.3.2.1 Data structures

Both State Transition models use the same data structures for the packets. This data structure is illustrated in Figure 2.14. The extra field is used to keep track of the number of collisions a packet has suffered. This number is used to calculate the backoff time using the binary exponential backoff algorithm. This means that the model with the exponential backoff algorithm can use a more general data structure, e.g. CSMACD (Figure 2.2). Both models use the same data structure only for simplicity and clarity.

2.3.2.2 Models implementation

The top level module is shown in Figure 2.15. It also consists of a traffic generator, State Transition module, and a receiver. The traffic generator is the same for both models. It creates Ethernet packets and sends them to the State Transition module. The receiver obtains the delay for each packet.

An expansion of the State Transition module is shown if Figure 2.16. In this figure, the State Machine module controls the state of the channel, it informs every incoming packet of the present state. Depending on the state, the arriving packet is sent from the Ethernet Buffer module to the channel (output), the backoff module, or the Wait for Idle module.


DSF Window (Browse Mode)

Name: STM

5 of 5 Fields Displayed

Super: CSMA/CD

Author: saurez

Date: 31-Oct-1990 9:52:06

Revision History: YES

Documentation: YES

Field	Type	Subrange	Default Value	Doc.
Instance Identifier	ROOT-OBJECT	N/A	N/A	YES
Packet Length	REAL	>= 0	0.0	YES
Time created	REAL	>= 0	0.0	YES
Time received	REAL	>= 0	0.0	YES
No. Collisions	INTEGER	>= 0	0	YES

FIGURE 2.14
Data Structure for the
State Transition Models

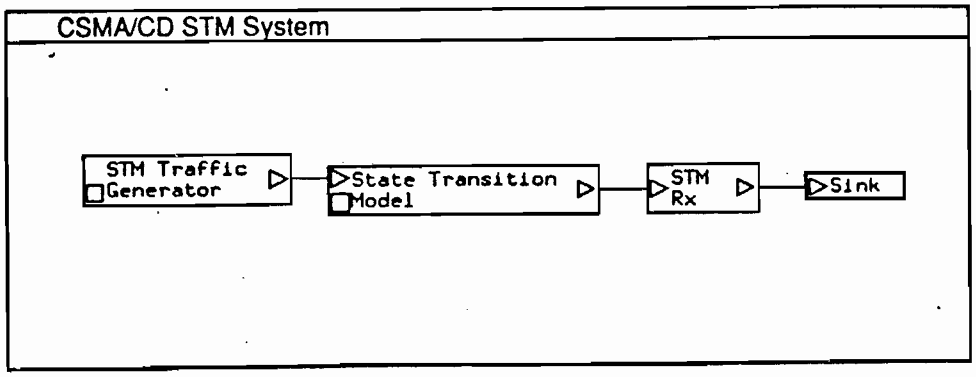


FIGURE 2.15
BONeS™ State Transition Model
Simulation System

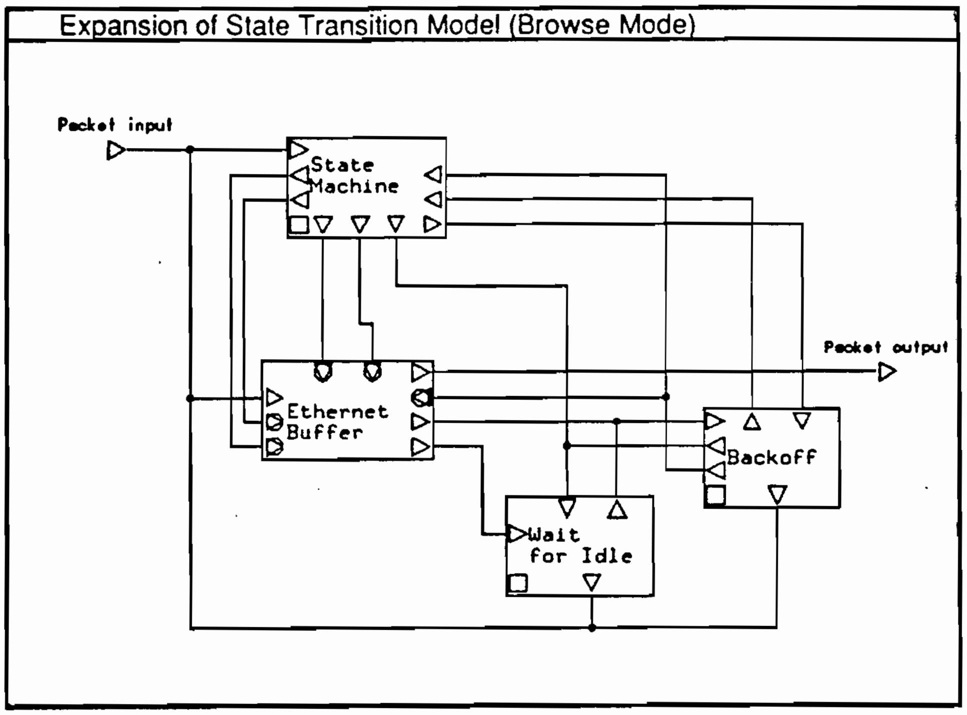


FIGURE 2.16
BONeS™ State Transition Model

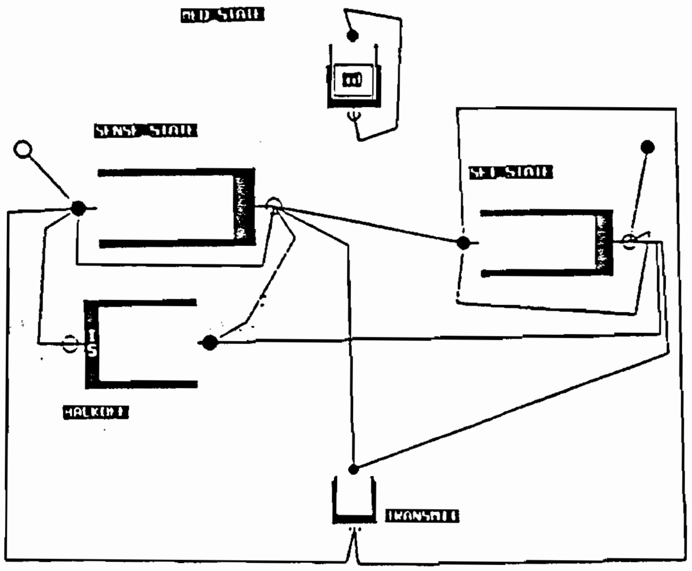


FIGURE 2.17
Q+™ State Transition Model

The packets stored in the Backoff Queue are released after a random time interval which starts when the collision is detected (when t2 ends). This interval depends on the backoff algorithm used. The internal structure of this module varies with the exponential backoff and the binary exponential backoff. The packets kept in the Wait for Idle module are released every time the State Machine module indicates the idle states. Appendix B contains all the submodules that conform the State Transition models.

2.3.3 State Transition Models Q+™ Implementation

Because of its complexity the State Transition model with binary exponential backoff was not realized on Q+. Figure 2.17 shows the implementation of the State Transition model with exponential backoff[11]. In the figure, transmitted packets arrive to the SENSE_STATE node (class **st1**) and they check the present state (**id**, **t1**, **t2**, or **tm**) from the MED_STATE node. If the present state is **t1** the packets are routed to the BACKOFF module as a **cl** class.. If the channel is busy (states **t2** or **tm**) the packet stays in the SENSE_STATE node but its class changes to **wt1**. When a packet finds the **id** state, it is routed to the TRANSMIT node where times **t1**, **t2**, and **tm** are simulated. When time **t2** ends, the BACKOFF queue is checked for collisions. If there was at least one collision the **cl** classes in BACKOFF become **bo**, and they are going to be transmitted after an exponential random time. If there were no collisions the state is set to **tm**. The SET_STATE node checks the present state and sets the proper next state. This model was build for exponential and general arrivals of **st1** class in the SENSE_STATE.

2.4 SIMLAN™ and GLAS models

SIMLAN™ and GLAS provide CSMA/CD user configurable models. The parameters of both programs were set as close as possible to each other. This considering that SIMLAN™ utilizes a worst case approach for the propagation delay while GLAS allows to specify the distance between two adjacent stations. Both systems were configured to have ten nodes or stations. The model was build for exponential and general (uniform) arrivals .

SIMLAN™ provides different configurations for its CSMA/CD model. It also allows the user specify any parameters. This feature was used to implement a model similar to the model built on GLAS, which is based on the MAC layer.

2.5 Analytical Model

The analytical model used in this project was taken from [1]. The equation for the normalized transmission delay for CSMA/CD is given by:

$$\bar{D}_n = \rho \frac{\left[\frac{\bar{m}^2}{(m)^2} + (4e+2)\alpha + 5\alpha^2 + 4e(2e-1)\alpha^2 \right]}{2\{1-\rho[1+(2e+1)\alpha]\}} + 1 + 2e\alpha - \frac{(1-e^{-2\alpha\rho})\left(\frac{2}{\rho} + 2\alpha e^{-1} - 6\alpha\right)}{2[F_p(\lambda)e^{-\rho\alpha-1} - 1 + e^{-2\rho\alpha}]} + \frac{\alpha}{2} \qquad (2.6)$$

where:

$F_p(\lambda)$ = Laplace transform of the packet length distribution

For fixed packet length:

$F_p(l) = e^{-r}$ $\qquad \bar{m}^2/(m)^2 = 1$

For exponential packet length:

$F_p(l) = 1/(1+r)$ $\qquad \bar{m}^2/(m)^2 = 2$

Equation 2.6 was obtained from a discrete time analysis based on slots 2τ units of time wide, where τ is the maximum end-to-end delay along the bus. This model only can be used to analyze models with exponential interarrival times.

CHAPTER 3: Simulations and Results

In order to test the performance of the different modeling abstractions in predicting the behavior of a CSMA/CD system, several kinds of simulation were run. Simulations were executed with variations in the arrival process and packet length distributions. This will show the behavior of each model under different traffic configurations for the CSMA/CD system. Ten subruns were executed for each simulation to obtain valid statistical results.

An estimate of the simulation execution time was obtained for each simulation. This time is compared among related models to observe the variation of speed with the abstraction level. The execution time is estimated in seconds for a Sparcstation 1+ computer. Since the Q+™ simulations were run on a 3B2/550, it was necessary to obtain a conversion factor by running a performance benchmark (Whetstone) on both machines.

In the next sections the main considerations taken to execute the simulation models are explained. First, the results of the simulations are presented separately for every model compared to the base system. Then, a model performance comparison is made among the different models with respect to the base system. This is done for systems with exponential arrivals and packet length, general (uniform) arrivals and exponential packet length, and for exponential arrivals and fixed packet length.

3.1 Simulation Parameters

The general parameters used to simulate each model describe the same CSMA/CD system. The main parameters that defined the simulated system are shown in Table 3.1.

PARAMETER	VALUE
Channel Speed	10Mbps
Mean packet length	8361 bits
Mean packet Tx time	836.1 ms
Propagation delay	.8907453 ms
$a = t/\overline{m}$	0.00106
Packet Length distrib.	Exponential, Constant*
Arrival traffic distrib.	Exponential, Uniform
System Distribution	M/M, G/M, M/G*
No. Nodes**	10
Simulation Run Length	2500*Mean Interarrival time

TABLE 3.1
CSMA/CD System main parameters

The simulation length was the same for every model. It was set to 2500 times the mean interarrival time. With this value, approximately 2500 packets are sent to the system in each simulation. The mean interarrival time determines the load offered to the systems. The following expression was used to determine the mean interarrival time:

$$T_i = \frac{\overline{m}}{\text{Load}} \qquad (2.5)$$

where Load = 0.1, 0.2, ..., 0.9
$\overline{m}$: Mean Packet Trans. time

The parameters set for each model implementation are presented in Appendix C.

* Only for the BONeS models
** Only for SIMLAN and GLAS. The BONeS and Q+ models assume an infinite number.

3.2 Description of Results

3.2.1 Single Queue model results

The Single Queue model was implemented and simulated on Q+™ and BONeS™. Figure 3.1 shows plots of the Load-Delay characteristic for this model. This figure shows the curves for the Q+™ and BONeS™ models simulated with exponential (Figure 3.1a) and uniform (Figure 3.1b) distributions for the arrival process and exponential packet lengths. The curve obtained from the GLAS system is also presented for comparison on both plots.

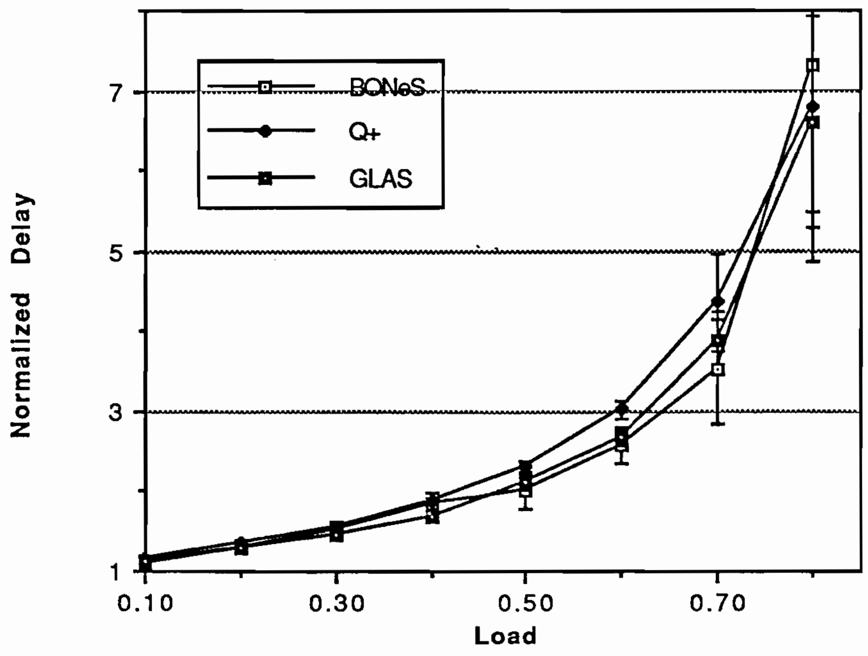


FIGURE 3.1a
Single Queue model delay for exponential arrivals and packet length

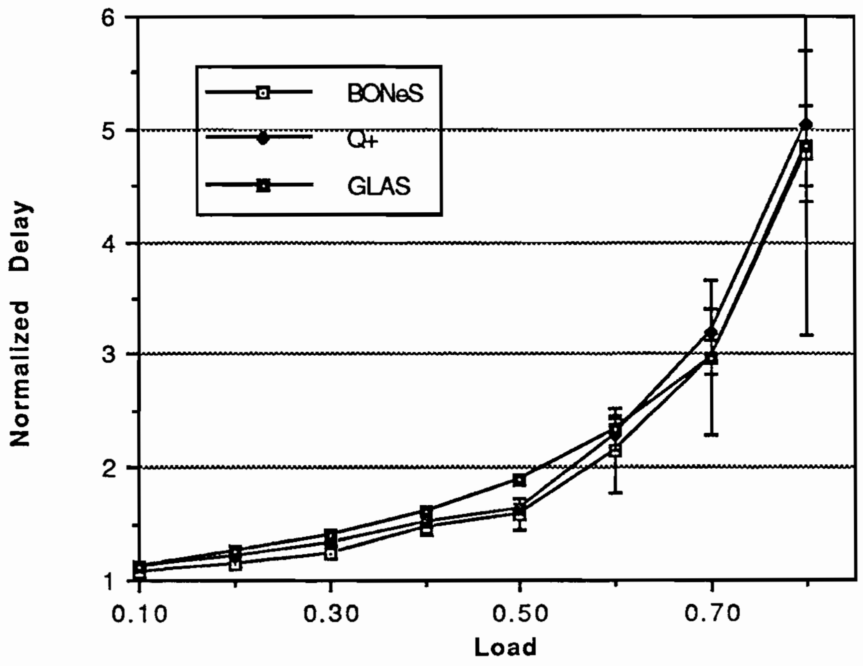


FIGURE 3.1b
Single Queue model delay for uniform arrivals and exponential packet length

From these plots it can be seen that the BONeS™ and Q+™ models present a very similar normalized mean delay for the different loads as expected. The GLAS curve seems to match very well the Load-Delay characteristic presented by the Single Queue model. This indicates that a CSMA/CD model with a high level of abstraction is able to predict the performance of a CSMA/CD system. In the next sections these results are compared to the ones obtained from the other models. Thus, the impact of level abstraction in network simulation can be studied.

The estimated execution time for the Single Queue model required for each simulation tool is represented in Figures 3.2a and 3.2b. Both figures show a big difference between the GLAS execution times and the BONeS™ and Q+™ execution times. The Single Queue model presents an almost flat curve of execution time vs. load. This is because of the simplicity of the Single Queue model relative to the GLAS model. The BONeS™ model evidences a difference of speed over the Q+™ model by a factor of two approximately for the two types of arrival distributions.

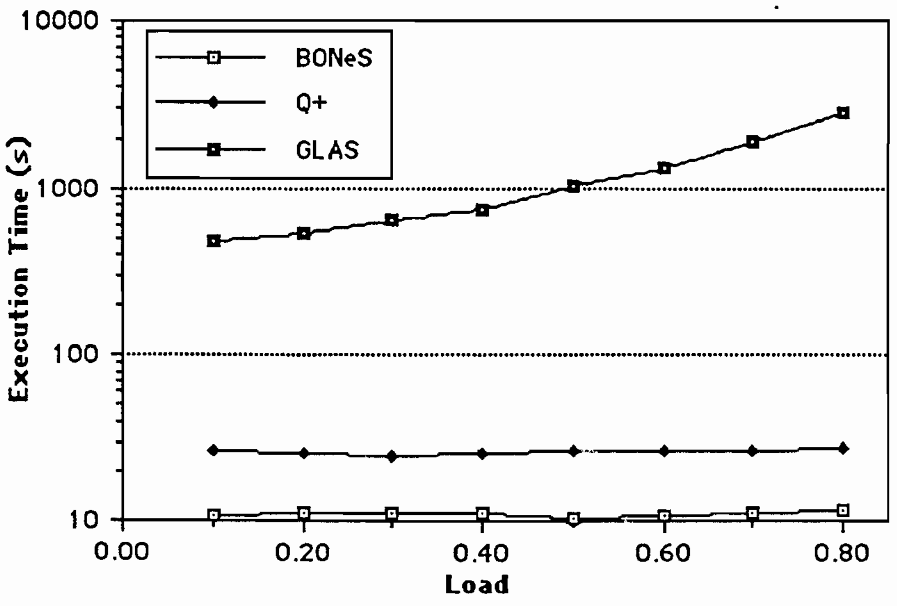


FIGURE 3.2(a)
Single Queue model execution time for exponential arrivals and packet length

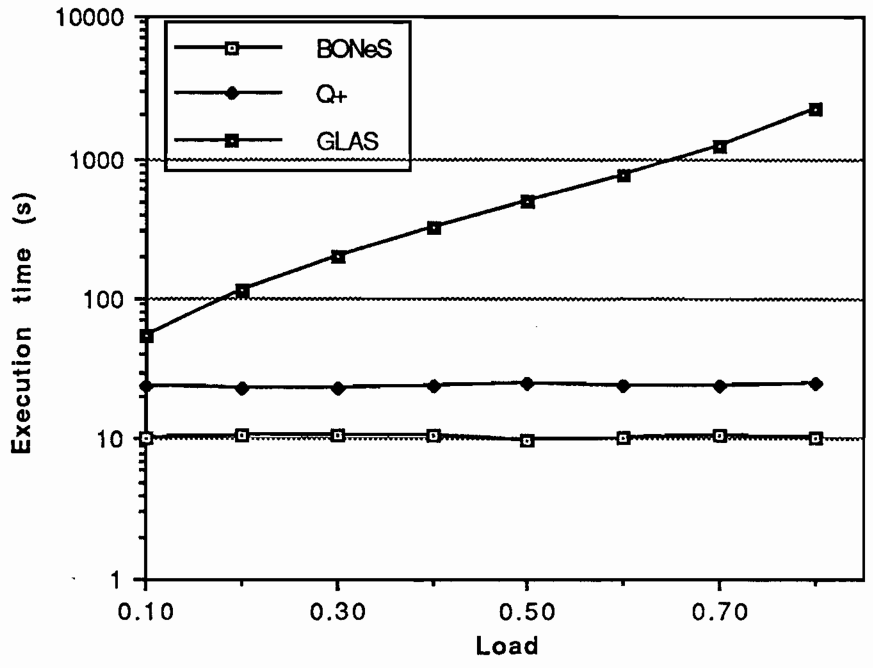


FIGURE 3.2(b)
Single Queue model execution time for uniform arrivals and exponential packet length

3.2.2 Priority Queue model results

The mean delay results obtained from the Priority Queue model simulations are presented in Figure 3.3 for BONeS™ and Q+. This Load-Delay characteristic is compared to the one obtained from the GLAS model. In Figures 3.3a and 3.3b, the BONeS™ and Q+™ models are again very close to each other. When the GLAS characteristic curve is compared to the Priority Queue results (Figure 3.3a), a suitable similarity is found for low loads (less than 70%). The difference in the mean delay becomes more appreciable for higher loads. This difference become even bigger in the case of uniform arrivals. This indicates that the Priority Queue model does not behave considerably well when the system incurres in a high number of collisions, which is the case for high loads. Note the control of backed off packets is not considered properly by the Priority Queue model. In addition, the Priority Queue model performance seems to deteriorate for nonexponential arrivals.

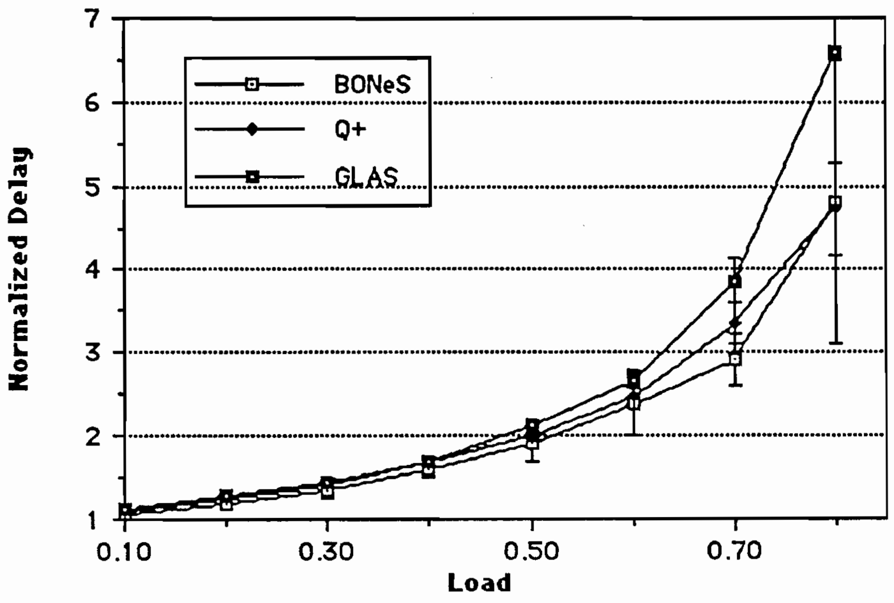


FIGURE 3.3(a)
Priority Queue model delay
for exponential arrivals and packet length

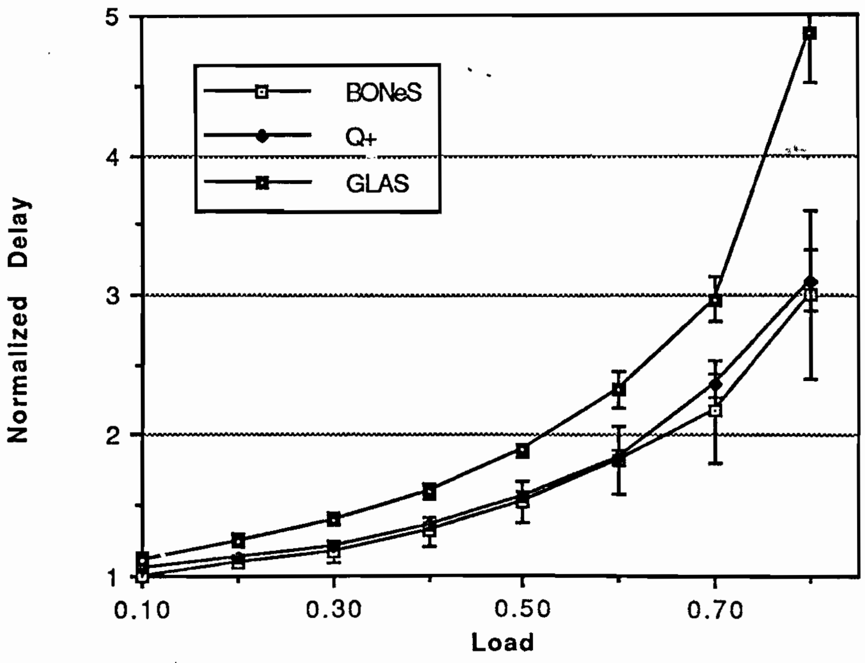


FIGURE 3.3(b)
Priority Queue model delay for uniform
arrivals and exponential packet length

The execution time for the three simulations is compared in Figures 3.4a and 3.4b. It can be seen that the relationship among the models is similar to the one obtained for the Single Queue model (Figure 3.2). The Priority Queue model has a very simple structure compared to the GLAS model. Thus, the difference in execution times becomes high, specially for high loads. The Priority Queue curves are practically constant because of the simplicity in the collision control schemes of the model. So far, the Single Queue models seems to be a more accurate (and simpler) model to analyze CSMA/CD system relative to the the Priority Queue Model.

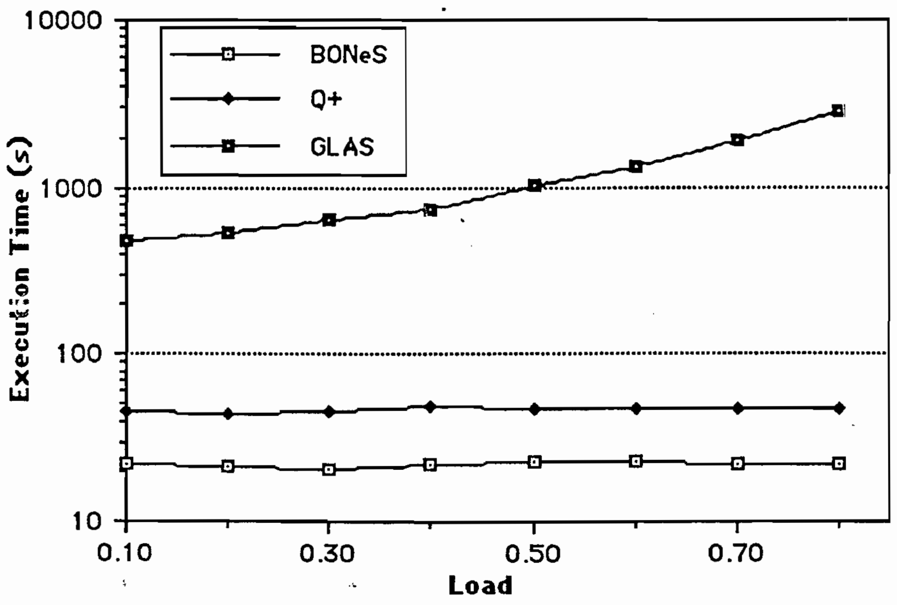


FIGURE 3.4(a)
Priority Queue model execution time
for exponential arrivals and packet length

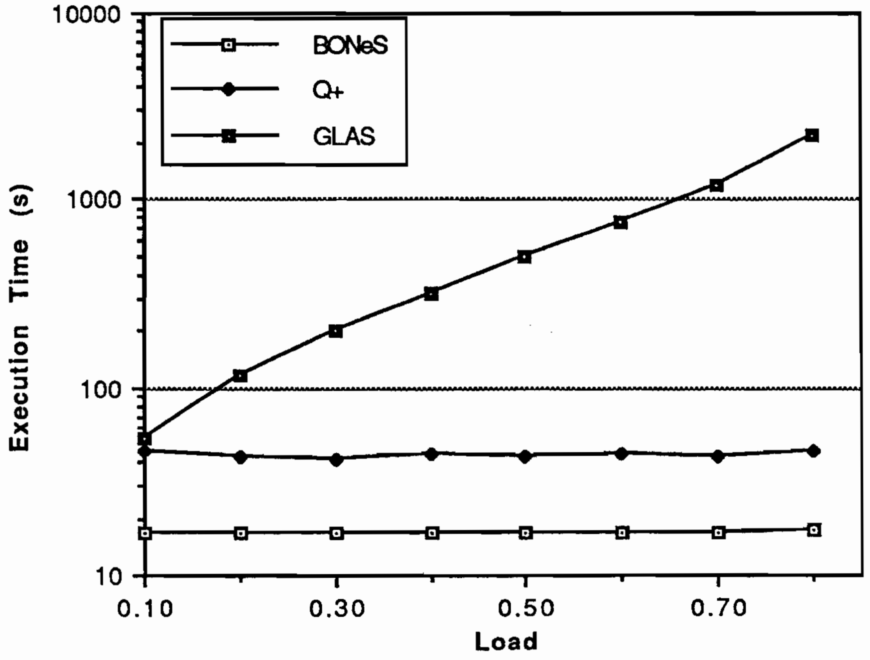


FIGURE 3.4(b)
Priority Queue model execution time for uniform arrivals and exponential packet length

3.2.3 State Transition models results

Simulations were run for two State Transition models which vary in the backoff algorithm they employ. The Load-Delay characteristic of the State Transition model with exponential backoff (STM1) and the State Transition model with binary exponential backoff (STM2) is given in Figure 3.5. The two curves are also compared to the GLAS results. Once again, BONeS™ and Q+™ results match very well for the STM1 model for exponential and uniform arrivals across all the loads. The STM1 model behaves similar to the GLAS models for loads not higher than 70% (difference is bigger for uniform arrivals), while the STM2 model results follow closely the GLAS results for high and low loads. That is, the State Transition model with binary exponential backoff algorithm (utilized by STM2) is a better approximation for a CSMA/CD model, with a very high level of abstraction, than the State Transition model with exponential backoff.

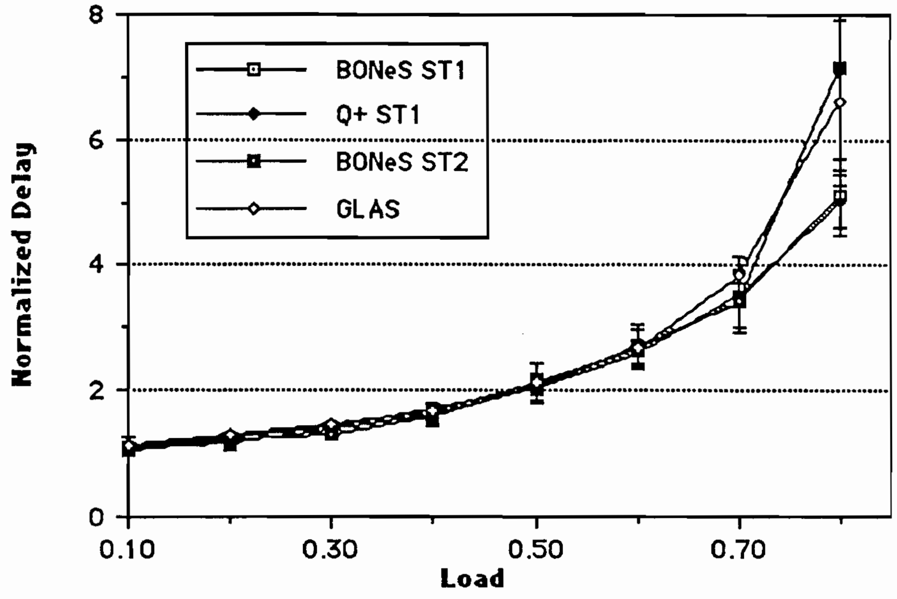


FIGURE 3.5(a)
State Transition model delay
for exponential arrivals and packet length

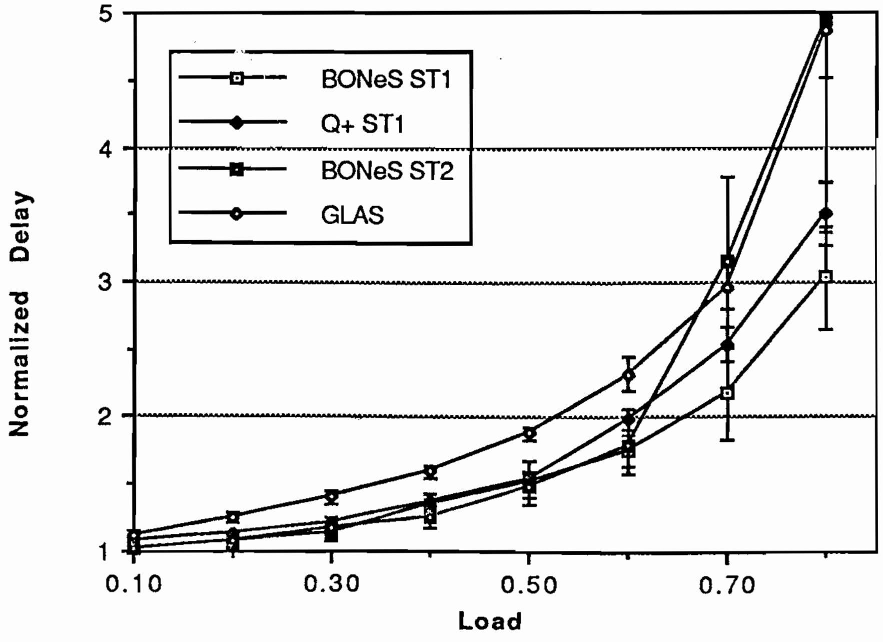


FIGURE 3.5(b)
State Transition model delay for
uniform arrivals and exponential packet length

The simulation execution time of the four models is compared in Figure 3.6. This time, none of the time curves show a flat response. The simulation completion time becomes higher as the load increases for both arrival distributions. The interpretation for this is that all these models follow closely the CSMA/CD protocol, which requires higher processing time when the number of collisions is increased as the load becomes higher. The BONeS™ models required much less time to complete the simulation for the STM1 and STM2 models. These time results and the ones obtained for the Single Queue model (Figure 3.2) and Priority Queue model (figure 3.4), make BONeS™ the most cost-effective (computer time) tool for simulating the mentioned models.

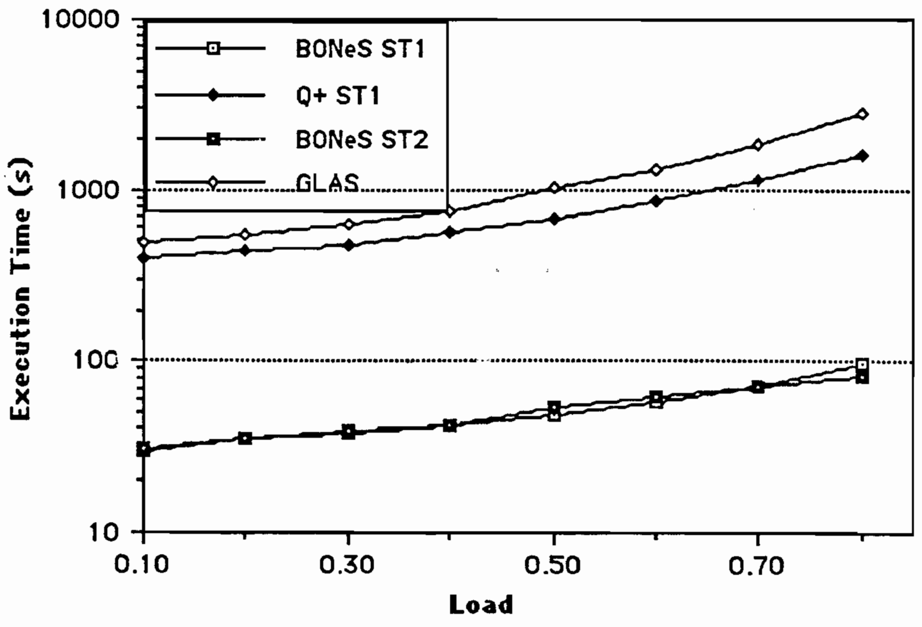


FIGURE 3.6(a)
State Transition model execution time for exponential arrivals and packet length

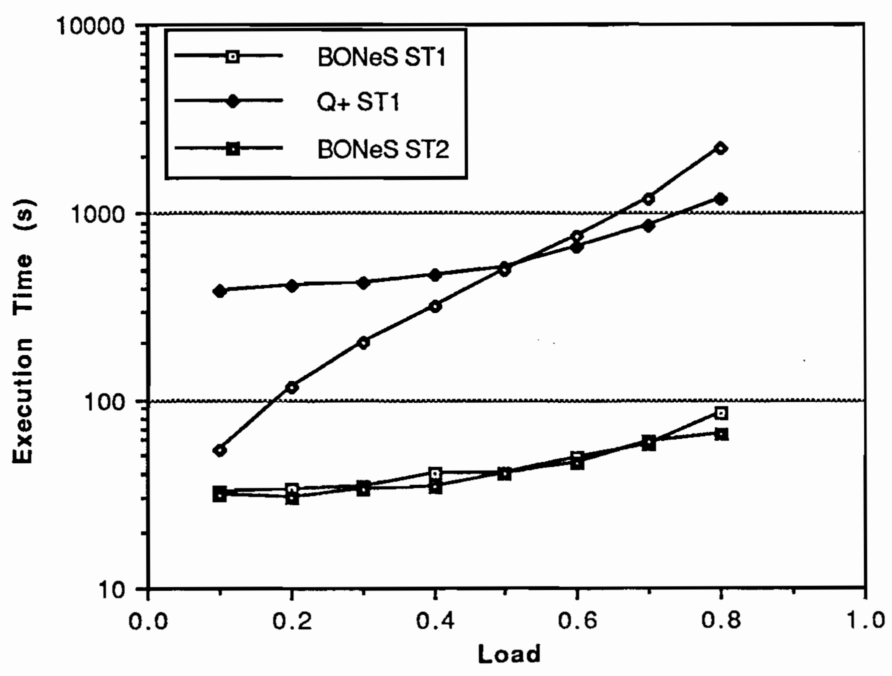


FIGURE 3.6(b)
State Transition model execution time for uniform arrivals and exponential packet length

3.2.4 SIMLAN™ and Analytical models results

Two CSMA/CD models with a low level of abstraction, and an analytical description are compared in this section. Figure 3.7 shows the Load-Delay characteristic obtained from the GLAS, SIMLAN™, and analytical models. From Figure 3.7a, it can be seen that the analytical curve is very close to the other two curves for loads lower than 70%. The analytical model was obtained without considering any backoff delay. Further, as the load increases, the number of collisions in the system also increases. This explains the delay difference of the analytical model with respect to the base system for high loads. The SIMLAN™ model performs very well for uniform arrivals and exponential packet lengths as can be seen in Figure 3.7b. The SIMLAN™ Load-Delay characteristic is very close to the GLAS characteristic specially for low loads (less than %70).

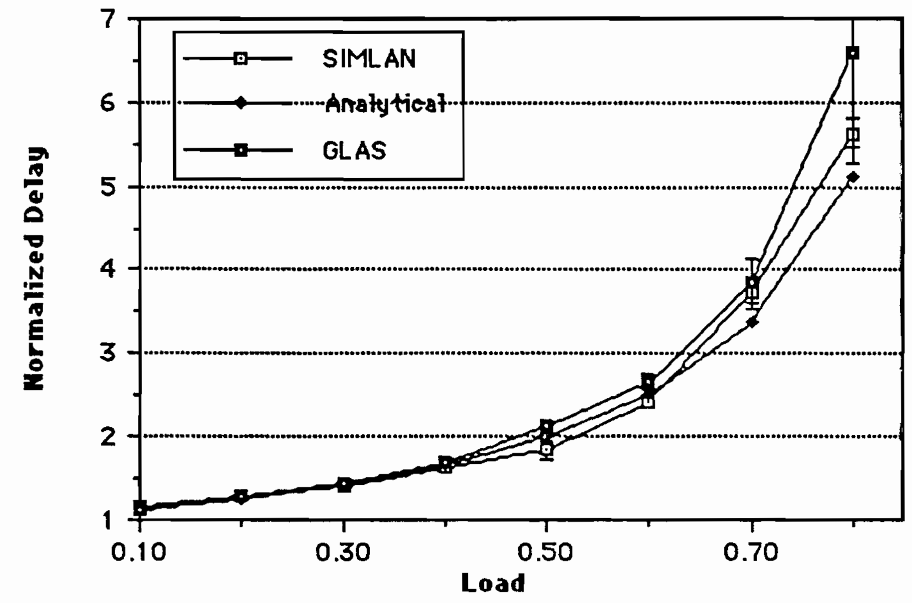


FIGURE 3.7(a)
SIMLAN™ and Analytical models delay for exponential arrivals and packet length

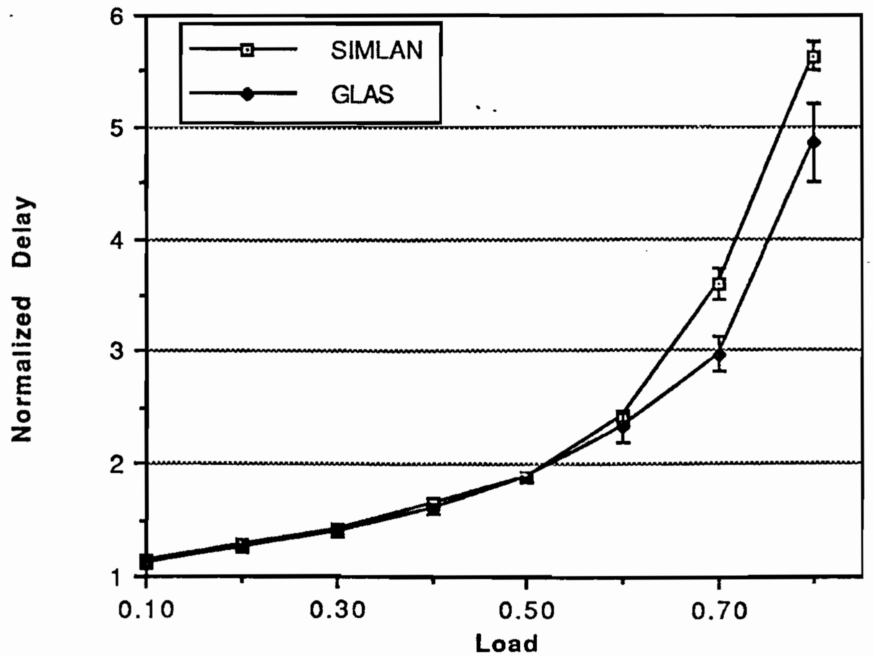


FIGURE 3.7(b)
SIMLAN™ model delay for uniform arrivals and exponential packet length

The estimated execution time of the SIMLAN™ model is compared to the one of the GLAS model in Figures 3.8a and 3.8b for exponential and uniform arrivals. An interesting result is shown in these figures. The SIMLAN™ model is considered as a low level modeling abstraction. However, the execution time only increases slightly when the load becomes higher. The increasing rate is not as big as the one presented for the others models with a low level of abstraction.

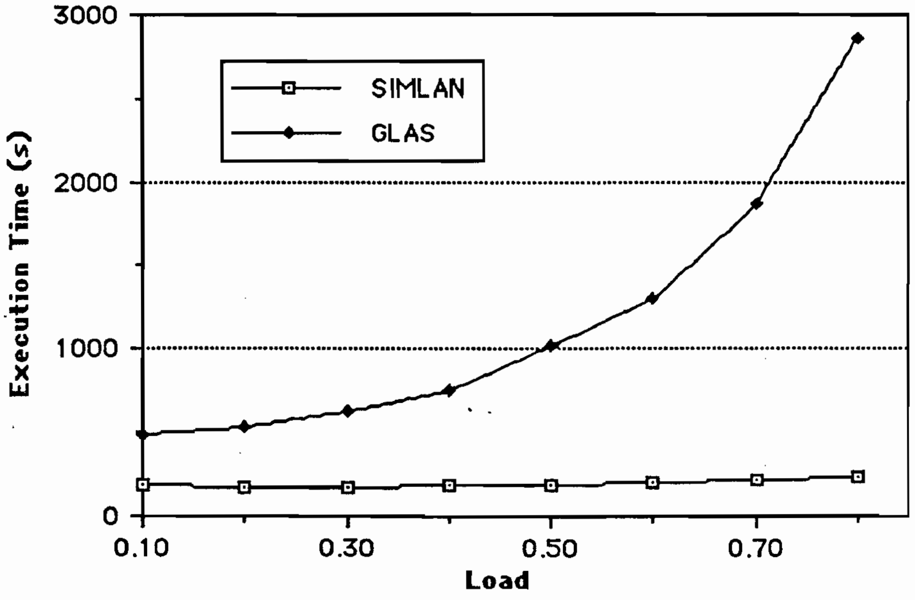


FIGURE 3.8(a)
SIMLAN™ and GLAS models execution time for exponential arrivals and packet length

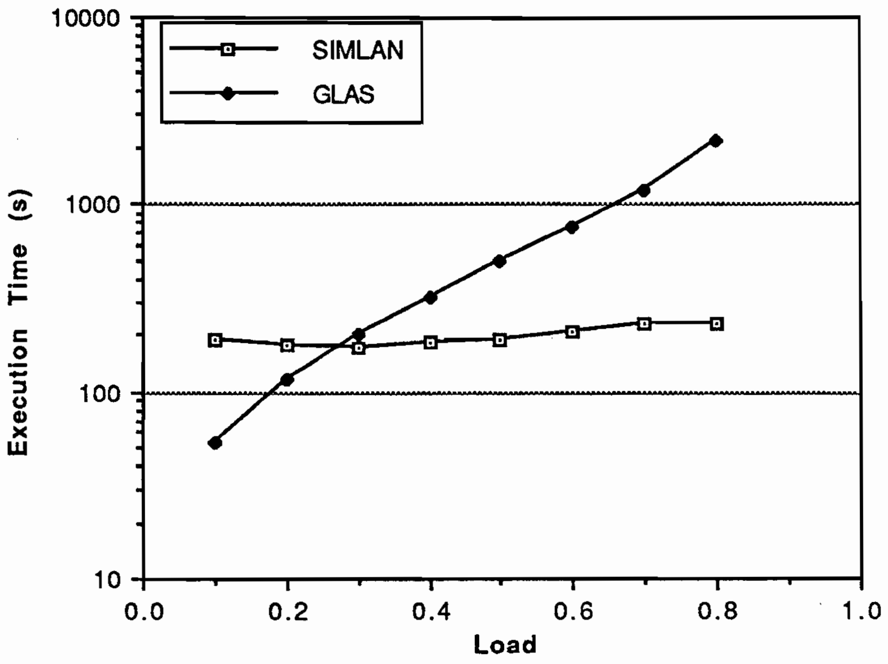


FIGURE 3.8(b)
SIMLAN™ and GLAS models execution time for uniform arrivals and exponential packet length

3.2.5 Additional Results

Results where also generated for a CSMA/CD system with exponential arrivals and fixed packet lengths. Figure 3.8 presents a comparison of the Load-Delay characteristic for the Single Queue, Priority Queue, State Transition, and GLAS models. It can be seen that the delay difference among the models is small for low loads (<70%). The variance becomes bigger for higher loads. It can be seen that the models that follow closest the GLAS results are the Single Queue Model and the State Transition model with exponential backoff. The next section presents these results as quality ratio comparisons

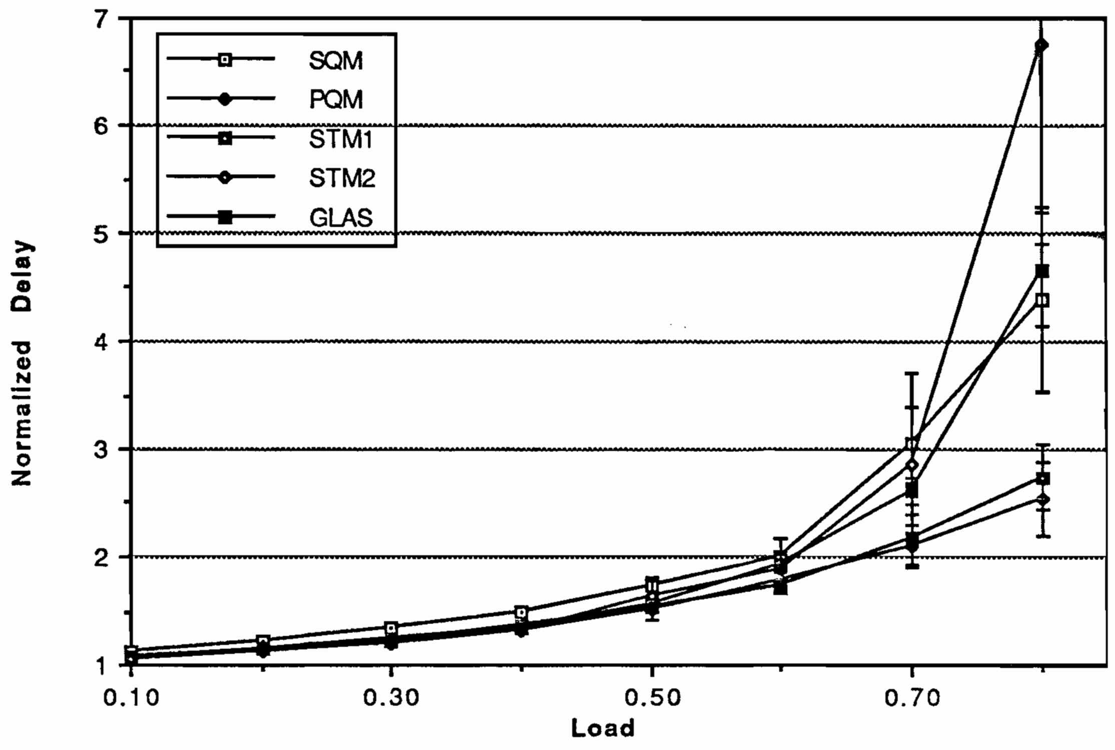


FIGURE 3.9
Models Comparison for Exponential Arrivals and Fixed Packet Length

3.2.6 Final results

Simply by looking at Delay vs. Load graphs presented in here, three models (e.g. STM2, Single Queue, and SIMLAN™) seem to be the ones that follow closely the behavior of the base system for high and low loads. To obtain the most accurate system with respect to the base system, a quality ratio was obtained by dividing the delay of each model by the GLAS delay. This was done for each load. Figures 3.10, 3.11, and 3.12 present a comparison of the quality ratio for every modeling abstraction and for different loads. These graphs allow the selection of an accurate model for a specific load and different arrival and packet length distributions.

Figure 3.10 present the quality ratio at each load for simulations with exponential arrivals and packet lengths. The average of the quality ratio is obtained for each model across all loads. This permits to identify the models that, in the average, follow most closely the characteristic curve of the base system utilized in this project. This average Quality Ratio is presented in Table 3.2. From the table it can be seen again that the Single Queue, STM2 and the SIMLAN™ models are the three most accurate models for a CSMA/CD system with exponential arrivals an packet length. The model that most time differed from the base system happened to be the Priority Queue model. This was expected after looking at the results in Figure 3.3.

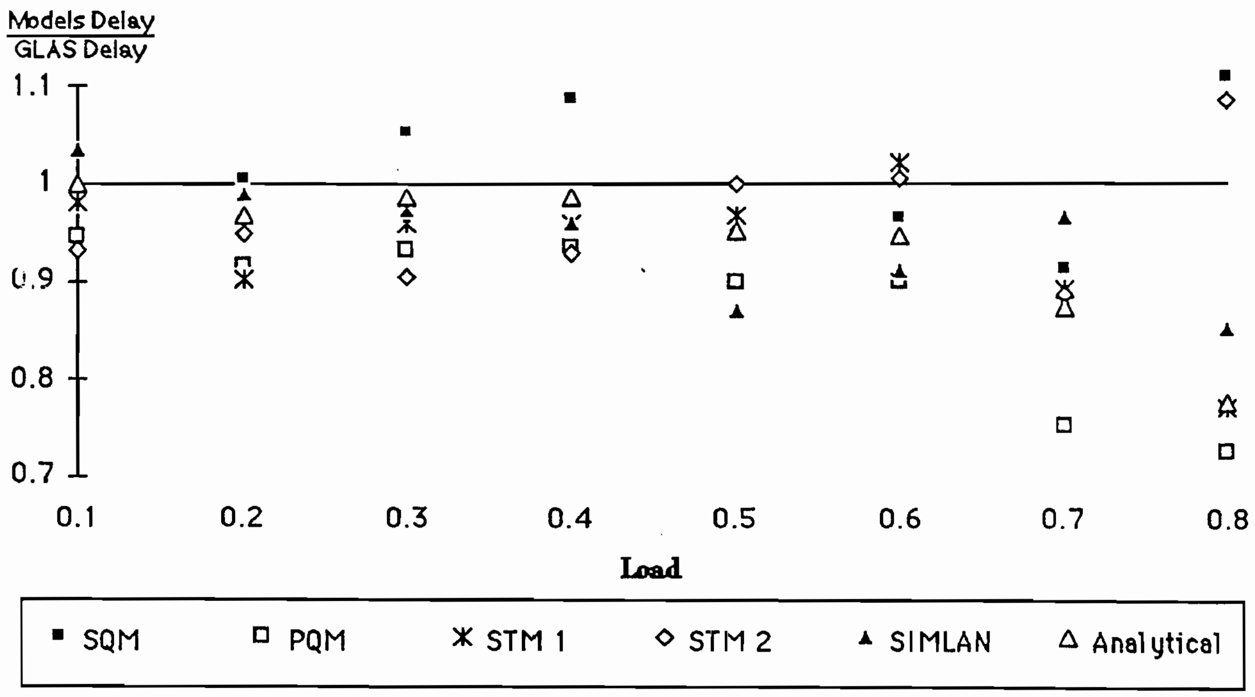


FIGURE 3.10
Modeling Abstractions Quality Ratio for Exponential Arrivals and Packet Length

MODEL	AVERAGE RATIO
Single Queue	1.00932737
ST BE Backoff	0.96108453
SIMLAN	0.94381265
Analytical	0.93541984
ST Exp. Backoff	0.93093699
Priority Queue	0.87577663

TABLE 3.2
Modeling Abstraction Average Quality Ratio for Exponential Arrivals and Packet Length

The same procedure for obtaining the accuracy of the models with respect to the base system was utilized to simulate and compare the different modeling abstractions of a CSMA/CD system with general (uniform) arrivals and exponential packet lengths. The Quality ratio at each load and for each model for this kind of system is presented in Figure 3.11. From the figure, the SIMLAN™ model appears to be the one with the most accurate characteristic curve with respect to the base system. The Single Queue model and the STM2 model are also close to the GLAS results.

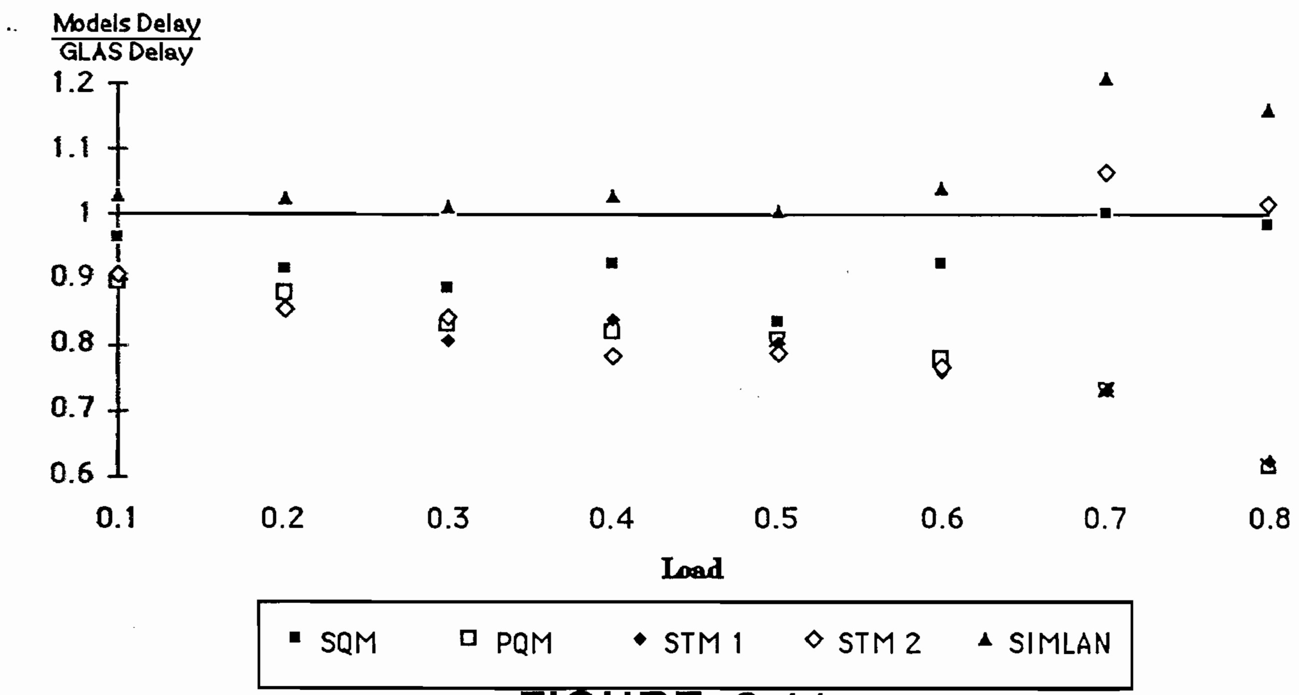


FIGURE 3.11
Modeling Abstractions Quality Ratio for Uniform Arrivals and Exponential Packet Length

Table 3.3 shows the average Quality Ratio of the models presented in Figure 3.11. Effectively, the SIMLAN™ model curve are closer in the average to the base system curve. Further, the Single Queue model and the State Transition model with binary exponential backoff are also close to the base system as evidenced in Figure 3.11 and Table 3.2.

MODEL	AVERAGE RATIO
SIMLAN	1.06295452
Single Queue	0.92929205
ST BE Backoff	0.87822261
Priority Queue	0.79562882
ST Exp. Backoff	0.7924525

TABLE 3.3
Modeling Abstraction Average Quality Ratio for Uniform Arrivals and Exponential Packet Length

The Priority Queue model and the Single Queue model are simpler to build in the Q+™ environment than in the BONeS™ environment. Thus, Q+™ should be considered when a single CSMA/CD system is to be simulated with a high level abstraction model.

This study shows that the arrival traffic distribution (exponential and uniform in this case) does not alter singificantly the performance of the different models. The delay difference of all the models with respect to the base system is a little higher for uniform arrivals. Furthermore, the most accurate models for exponential packet lengths are also the most accurate ones for fixed packet lengths.

The results of the study indicate that a simple model of a CSMA/CD network can provide accurate results. Further, it lead us to the conclusion that models of large interconnected networks can be constructed using simple models for its constituent LANs.

REFERENCES

[1] M. Schwartz, "Telecommunications Networks: Protocols, Modeling, and Analysis", Addison-Wesley Publishing Company, 1987.

[2] M. Marathe and S. Kumar, "Analytical models for an Ethernet-like local area network link", Proceedings ACM Sigmetrics Conference, September 1981.

[3] H.K. Hon and H.W. Lee, "Performance Analysis of the CSMA/CD and Movable-Slot TDM Protocols", IEEE INFOCOM 1987, pp.575-579

[4] B. Melamed, "The AT&T Performance Analysis Workstation. User's Guide and Reference Manual", AT&T Bell Laboratories, 1988.

[5] V.S Frost, W.W. LaRue, A.G. Mckee, A.J. Ernstein, P. Kishore, and M.J. Gormish, "A Tool for Local Area Network Modeling and Analysis", Simulation, pp.152-169, November 1989.

[6] CACI Products Company, "SIMLAN II™ User's Manual", Release 1.0, January 1990.

[7] Demetriou M., Frost V., "Modeling Networked Information Systems: Estimation of the User Perceived Performance", TISL Technnical Report 8580-2, University of Kansas, August 1990.

[8] V.S. Frost, K.S. Shanmugan, D. Resnik, W.W. LaRue, E. Komp, and S. Schaffer, "A Block Oriented Paradigm for Modeling Communications Networks", IEEE Military Communications Conference 1990, Vol.2, pp.689-695.

[9] Spirn J. R., Chien J., and Hawe W., "Brusty Traffic Local Network Modeling", IEEE Journal on Selected Areas in Communications, VOL. SAC-2, No.1, January 1984.

[10] Kleinrock L., "Queueing Systems Vol. 1", Wiley-Iterscience, New York,1975.

[11] Heffes H., Melamed B., "Visual Simulation of Teletraffic Models", AT&T Laboratories, New Jersey.

[12] Shoch J. F., Hupp J. A., "Measured Performance of an Ethernet Local Network", Communication of the ACM, Vol.23, No.12, December 1980.

[13] A.M. Law and W.D. Kelter, "Simulation Modeling and Analysis", McGraw Hill Inc., 1982.

[14] Hayes J. F., "Modeling and Analysis of Computer Communications Networks", Plenum Press, New York, 1984.

[15] J.L. Hammond and P. O'Reilly, "Performance Analysis of local computer Networks", Addison Wesley, Massachusetts, 1986

[16] Lam S. S., "A Carrier Sense Multiple Access Protocol for Local Networks", Computer Networks, Vol.4, No.1, January 1980.

[17] Kernighan B. W., Ritchie D. M., "The C Programming Language", Prentice Hall, second edition, New Jersey, 1988.

APPENDIX A: Comparison of Three Network Simulation Systems

This appendix presents a general discussion about three network simulation programs, such as: BONeS™, SIMLAN™ and Q+. The discussion is made from the user's point of view. It also describes the most important features and differences of the three applications. These features can be an important for the selection of one of these tools to study a specific system.

Initially, a general overview of the three systems is given. The way each program works is explained briefly. Finally, this overview is used to list some differences among the three simulation systems.

A.2 General Overview

This section gives a description of the three network simulation systems mentioned above. It is important to indicate that the three programs have a graphical user interface that avoid the need to write a program to implement a specific simulation.

A.2.1 BONeS™

BONeS™ stands for Block Oriented Network Simulator. Its user interface allows to build blocks and assign any function to them. These blocks can be interconnected so an information exchange may exist among blocks. A group of blocks can be encapsulated in one block, and this block can be encapsulated with other blocks until the top level module is obtained.

The information that is passed among blocks consists of packets (data structures or records with any number of fields of any type) or simple numbers. The packets can be modified, queued, and discarded depending on system structure. The BONeS™ standard library comes with basic blocks (number the other hand, BONeS™ and Q+™ allow the user to build almost any system. To build a system on BONeS™ is easier since it's like graphical programming. So the user only needs to know the exact algorithm of the system. In Q+™ the user must represent the system with queues, which can be very complicated.

BONeS™ provides for the construction of hierarchical modules. This makes easier to implement big and complicated systems. Q+™ and SIMLAN™ don't provide any kind of encapsulation.

The Q+™ user can build systems by writing a C program. Q+™ provides C functions that allow to create and modify Q+™ systems. BONeS™ and SIMLAN™ only allow the user to build the systems from the user interface. BONeS™ primitives can also be directly constructed in C.

BONeS' user interface can show help text to the user at any moment when building a system. Q+™ and SIMLAN™ don't have this feature.

SIMLAN™ gives the user no control of the data structures used to represent packets. Q+™ allows packets to be described by a class tag and a family membership. BONeS™ gives the user full control over a hierarchical description of data structures including encapsulation.

A.3.2 Simulation Execution

The Q+™ and SIMLAN™ users can actually see their simulations running. However, the Q+™ animation is easier to follow and to understand than the SIMLAN™ animation. Q+™ can show the packets moving around the system. While SIMLAN™ only shows the status of the system elements. SIMLAN™ does it by using colors, which can be confusing. BONeS™ doesn't have animated simulations, but the user can see part of the results with the Post Processor when the simulation is running.

Q+™ and BONeS™ run simulations on the background. In Q+™ this is done by writing a C program and running it from the UNIX host. While in

BONeS™ any number of simulations can be run on the background from user interface. SIMLAN™ can't run simulations on the background. The LANGIN program permits to save the files in batch mode, which allows to create a batch file an start any number of simulations from the UNIX host running only one at a time.

BONeS™ gives the user the facility to iterate parameters. The user only has to start the simulation and BONeS™ will run a specified number of iterations. Q+™ doesn't do that directly. The user has to write a C program with a loop that modifies a parameter and runs a simulation for every modification. There is no way to iterate parameters with SIMLAN™. The user has to create the n simulations and run them with a batch file.

For statistical purposes, BONeS™ can run any number of independent subruns specified by the user. Q+™ and SIMLAN™ don't have this feature. In Q+™ the user can write a C program and iterate the seed of the simulation. In SIMLAN™ the user has to manually change the seed and run the same number of simulations as subruns required.

A.3.3 Simulation Results

BONeS™ Post Processor allows user to plot any results in combination with statistical operations on data and other plots. But the user needs the user interface to look at the results. Q+™ can show windows for each node that display statistical results and plots dynamically as the simulation is running. The Q+™ user can read the results without using the user interface. This is done by running a command from the UNIX environment that creates a text file with the results. SIMLAN™ always creates a report file that stores statistical results. It also provides a program to plot the network utilization.

BONeS™ can give the user a confidence interval for different results in a simulation. Q+™ and SIMLAN™ don't provide the user with this information since they are not designed to iterate parameters nor provide independent subruns.

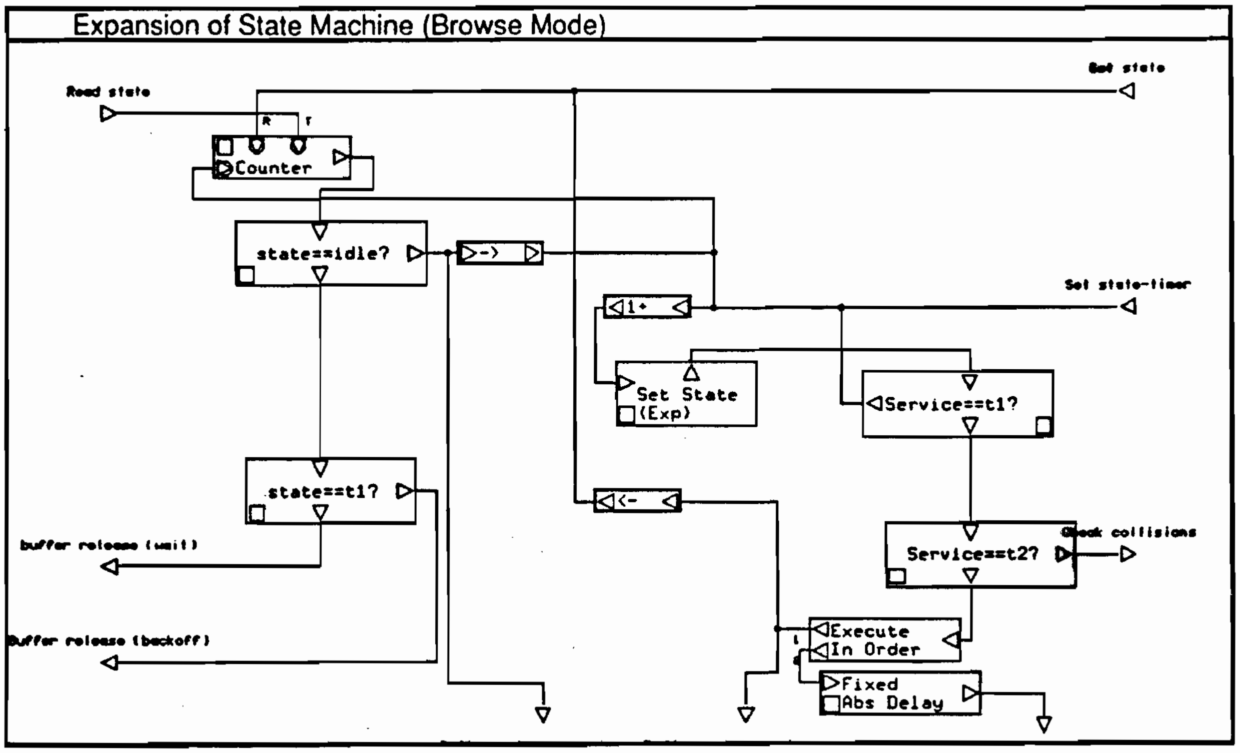


FIGURE B.6
State Machine Module

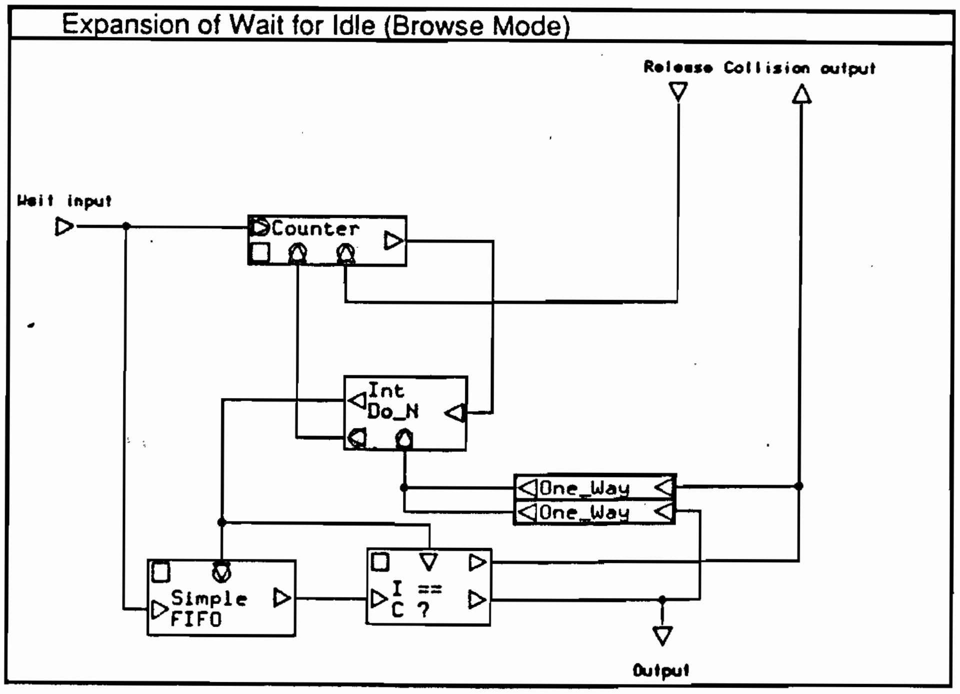


FIGURE B.7
Wait for Idle Module

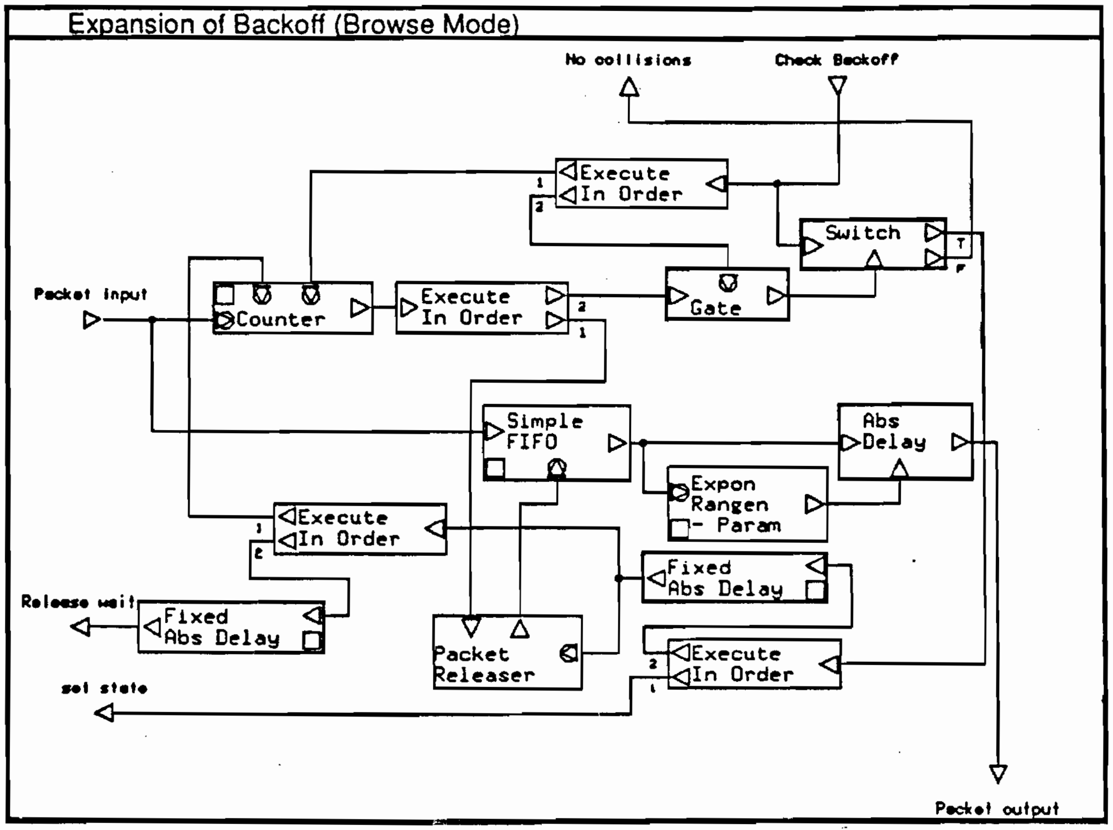


FIGURE B.8
Exponential Backoff Module

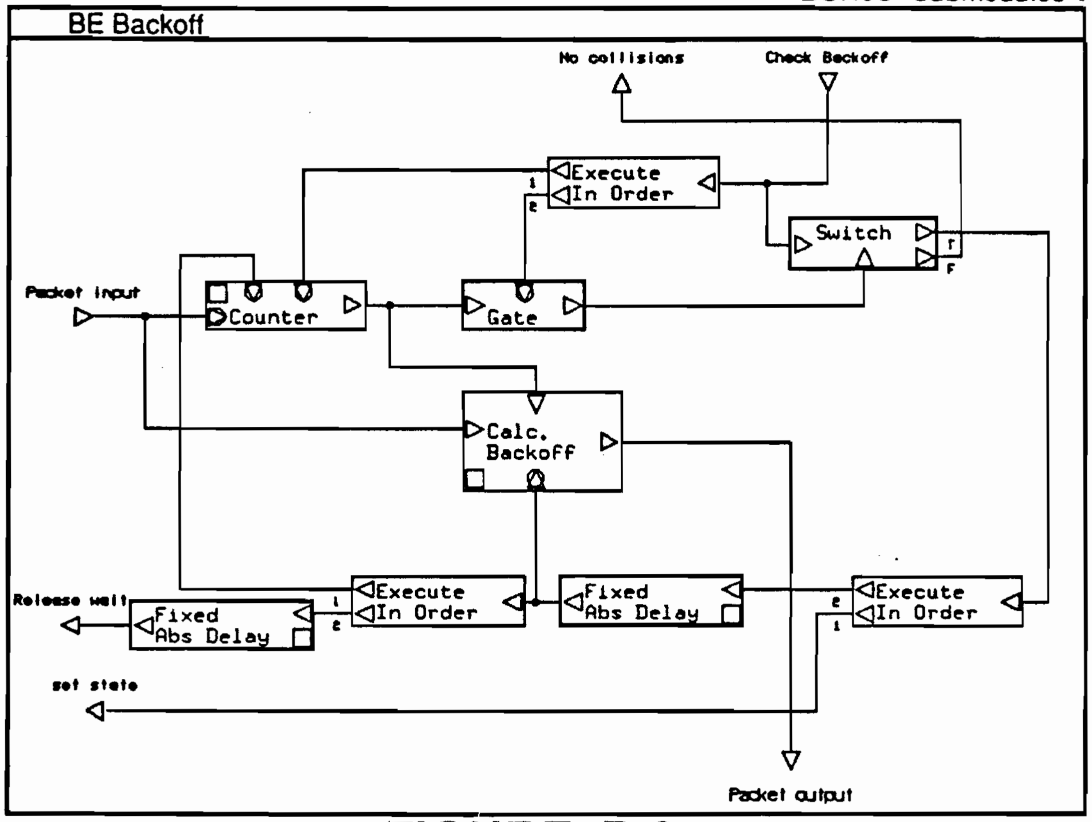


FIGURE B.9
Binary Exponential Backoff Module

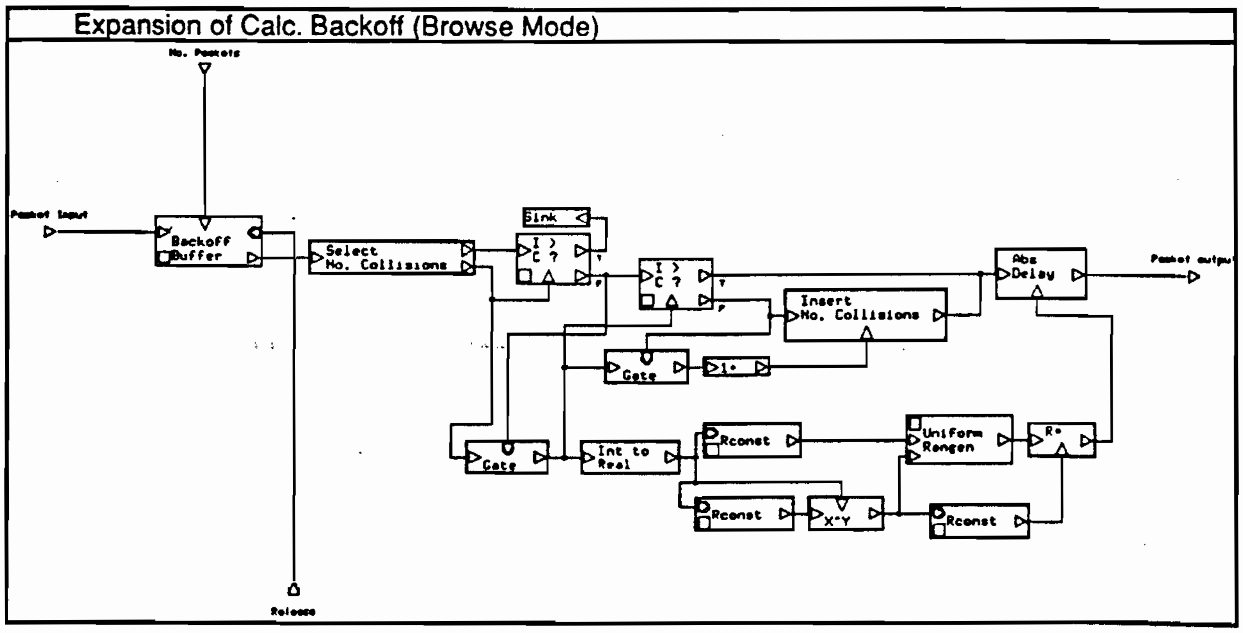


FIGURE B.10
Calc. Backoff Module (from Figure B9)

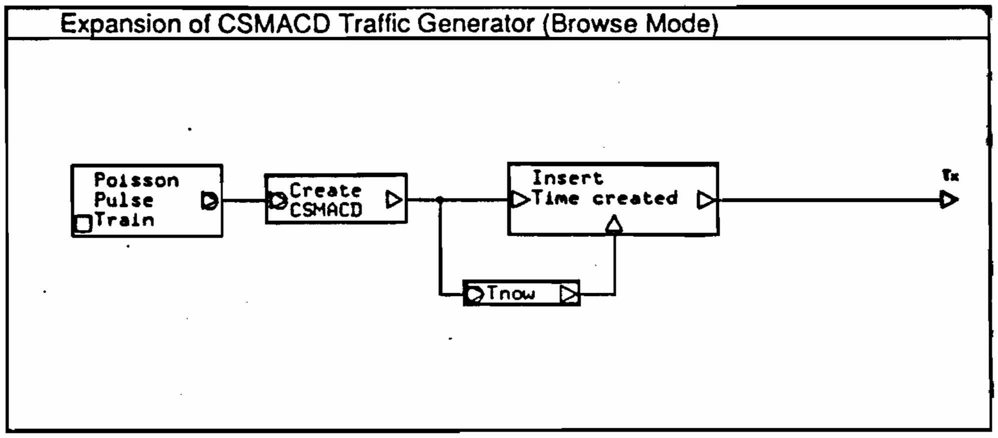


FIGURE B.11
CSMA/CD Traffic Generator Module

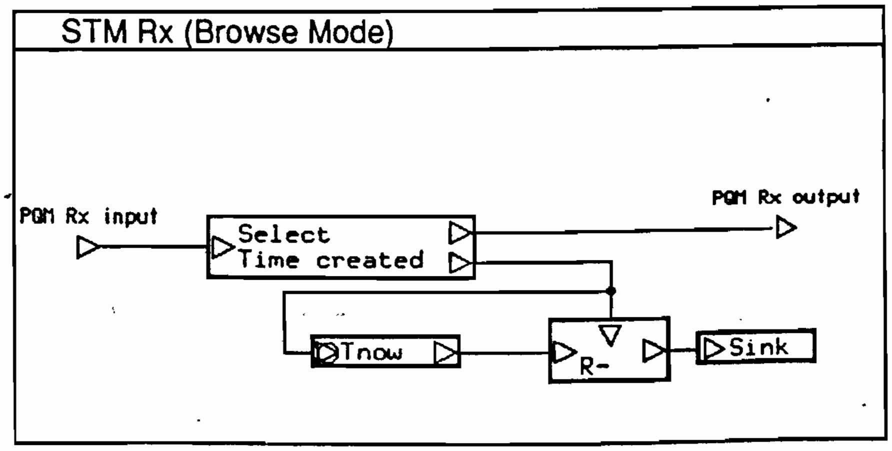


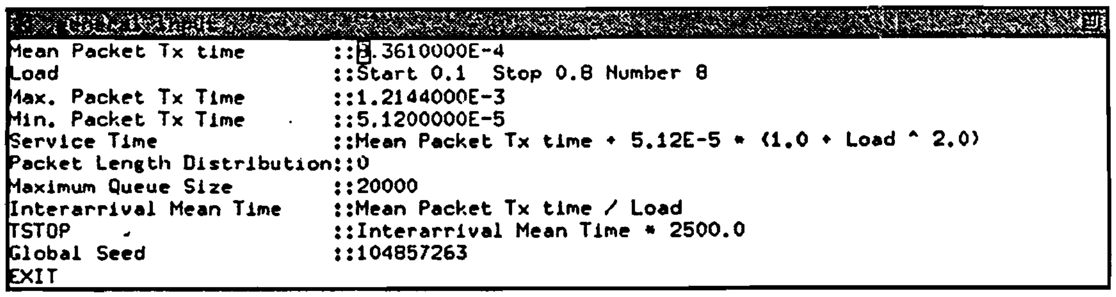
FIGURE B.12
STM Receiver Module

APPENDIX C: Simulation Parameters of each Model

This appendix presents the parameters utilized to simulate the Single Queue, Priority Queue, and State Transition models for Q+™ and BONeS™. The SIMLAN™ and GLAS parameters are also presented.

C.1 Single Queue model parameters

The specific parameters used for the Single Queue model are presented in figures C.1 and C.2 for BONeS™ and Q+. This figures only show the parameters for exponential interarrival times.



```
Mean Packet Tx time         ::8.3610000E-4
Load                        ::Start 0.1  Stop 0.8 Number 8
Max. Packet Tx Time         ::1.2144000E-3
Min. Packet Tx Time         ::5.1200000E-5
Service Time                ::Mean Packet Tx time + 5.12E-5 * (1.0 + Load ^ 2.0)
Packet Length Distribution::0
Maximum Queue Size          ::20000
Interarrival Mean Time      ::Mean Packet Tx time / Load
TSTOP                       ::Interarrival Mean Time * 2500.0
Global Seed                 ::104857263
EXIT
```

FIGURE C.1
BONeS™ Single Queue Model parameters

From figure C.1 the packet length distribution equal to one means an exponential distribution. Fixed and uniform packet lengths can be specified with 0 and 2 respectively.

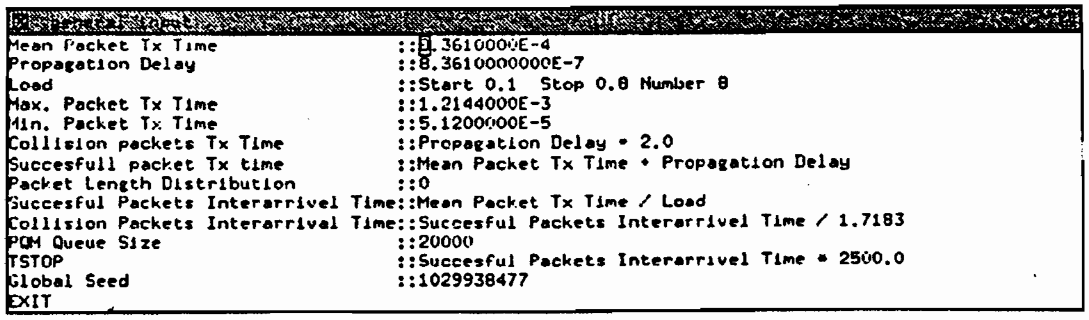
```
Q+ (6-1-89) model listing of SQM11.q+
=========================================

SIMULATION CLOCK READINGS:
current = 0.000000     interval = 0.000000

SIMULATION MARK TIMES:
reset = 0.000000     end = 2500.000000

Parameters for node  SQM
------------------------

Queueing discipline:  FCFS

Number of servers:  1

Queue capacity:  infinite

Transaction classes at node  SQM:
   class = pkt , priority = 0

Arrival interval of class  pkt:
   exponential (mean=  1.000000)

Service time of class  pkt:
   exponential (mean=  0.102020)
```

FIGURE C.2
Q+™ Single Queue model parameters

The parameters shown in figure C.2 correspond to a system with an offered load of 0.1.

C.2 Priority Queue model parameters

The parameters used with this model are shown in figures C.3 and C.4 for BONeS™ and Q+. This figures only show the parameters for exponentially distributed interarrival time intervals.



```
Mean Packet Tx Time                  ::8.3610000E-4
Propagation Delay                    ::8.3610000000E-7
Load                                 ::Start 0.1  Stop 0.8 Number 8
Max. Packet Tx Time                  ::1.2144000E-3
Min. Packet Tx Time                  ::5.1200000E-5
Collision packets Tx Time            ::Propagation Delay * 2.0
Succesfull packet Tx time            ::Mean Packet Tx Time + Propagation Delay
Packet Length Distribution           ::0
Succesful Packets Interarrivel Time::Mean Packet Tx Time / Load
Collision Packets Interarrival Time::Succesful Packets Interarrivel Time / 1.7183
PQM Queue Size                       ::20000
TSTOP                                ::Succesful Packets Interarrivel Time * 2500.0
Global Seed                          ::1029938477
EXIT
```

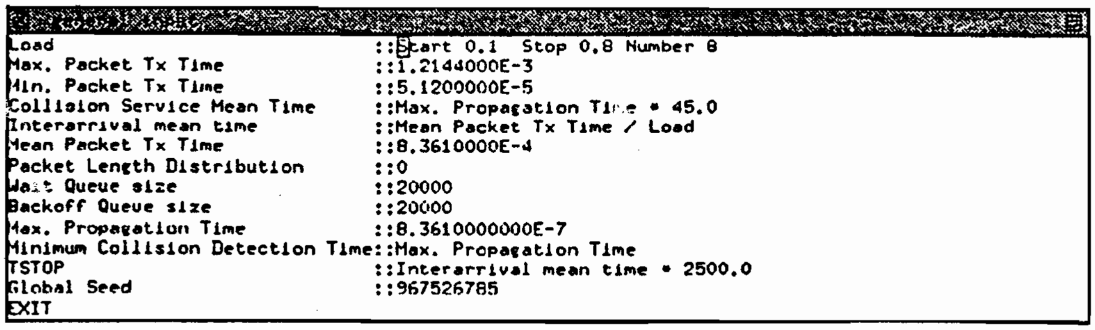
FIGURE C.3
BONeS™ Priority Queue model parameters

```
        Q+ (6-1-89) model listing of PQM11.q+
      ========================================

             SIMULATION CLOCK READINGS:
        current = 0.000000    interval = 0.000000

               SIMULATION MARK TIMES:
         reset = 0.000000     end = 312.500000

             Parameters for node  PQM
             ------------------------

Queueing discipline:  FCFS

Number of servers: 1

Queue capacity: infinite

Transaction classes at node  PQM:
   class = c1 , priority = 1
   class = c0 , priority = 0

Arrival interval of class  c1:
 exponential (mean=  0.500000)

Service time of class  c1:
   prob = 1. , time = 0.000002

Arrival interval of class  c0:
 exponential (mean=  1.000000)

Service time of class  c0:
 exponential (mean=  0.000837)
```

FIGURE C.4
Q+™ Priority Queue model parameters

C.3 Parameters for the State Transition model with exponential backoff

The parameters for the BONeS™ and Q+™ State Transition model with exponential backoff are presented in figures C.5 and C.6 respectively.



```
Load                           ::Start 0.1  Stop 0.8 Number 8
Max. Packet Tx Time            ::1.2144000E-3
Min. Packet Tx Time            ::5.1200000E-5
Collision Service Mean Time    ::Max. Propagation Time * 45.0
Interarrival mean time         ::Mean Packet Tx Time / Load
Mean Packet Tx Time            ::8.3610000E-4
Packet Length Distribution     ::0
Wait Queue size                ::20000
Backoff Queue size             ::20000
Max. Propagation Time          ::8.3610000000E-7
Minimum Collision Detection Time::Max. Propagation Time
TSTOP                          ::Interarrival mean time * 2500.0
Global Seed                    ::967526785
EXIT
```

FIGURE C.5
Parameters for the BONeS™ State Transition model with exponential backoff

```
                Q+ (6-1-89) model listing of Et11.q+
                ====================================
                    SIMULATION CLOCK READINGS:
              current = 0.000000     interval = 0.000000
                     SIMULATION MARK TIMES:
              reset = 0.000000      end = 2500.000000

                 Parameters for node   MED_STATE
                 -----------------------------
Queueing discipline:  FCFS
Number of servers:  1
Queue capacity:  infinite
Transaction classes at node   MED_STATE:
   class = t1 , priority = 0; class = t2 , priority = 0
   class = tm , priority = 0; class = id , priority = 0
Service time of class  t1: prob = 1. , time = infinite
Service time of class  t2: prob = 1. , time = infinite
Service time of class  tm: prob = 1. , time = infinite
Service time of class  id: prob = 1. , time = infinite
Queue configuration at node   MED_STATE: id

                 Parameters for node   SET_STATE
                 -----------------------------
Queueing discipline:  IM_CL_YANK
Number of servers:  infinite
Queue capacity:  infinite
Transaction classes at node   SET_STATE:
   class = xt1 , priority = 0; class = xt2, priority=0
   class = cd , priority = 0; class = xtm , priority=0
   class = rel , priority = 0; class = t1 , priority = 0
   class = t2 , priority = 0; class = bo , priority = 0
   class = tm , priority = 0; class = id , priority = 0
Yank targets of class  xt1:
   prob = 1. , from_node= MED_STATE, from_class = all
Yanked batch of class  xt1: prob=1.,size=1,hop_class=t1
Routing rule of class  xt1:
   prob = 1. , hop_node= TRANSMIT , hop_class = xt1
Yank targets of class  xt2:
   prob = 1. , from_node= MED_STATE, from_class = all
Yanked batch of class  xt2: prob=1.,size=1,hop_class=t2
Routing rule of class  xt2:
   prob = 1. , hop_node= TRANSMIT , hop_class = xt2
Yank targets of class  cd:
   prob = 1. , from_node= BACKOFF, from_class = cl
Yanked batch of class  cd:
  prob=1.,size=infinite,hop_class = bo
Routing rule of class  cd:
   prob = 1. , hop_node= SET_STATE , hop_class = bor
Alternate routing rule of class  cd:
```

prob = 1. , hop_node= SET_STATE , hop_class = bor
Yank targets of class xtm:
prob = 1. , from_node= MED_STATE, from_class = all
Yanked batch of class xtm:
prob = 1. , size= 1 , hop_class = tm
Routing rule of class xtm:
prob = 1. , hop_node= TRANSMIT , hop_class = xtm
Yank targets of class rel:
prob = 1. , from_node= MED_STATE, from_class = all
Yanked batch of class rel:prob=1., size=1, hop_class= id
Routing rule of class rel:
prob = 1. , hop_node= sink , hop_class = rel
Routing rule of class t1:
prob = 1. , hop_node= MED_STATE , hop_class = t1
Routing rule of class t2:
prob = 1. , hop_node= MED_STATE , hop_class = t2
Routing rule of class bo:
prob = 1. , hop_node= BACKOFF , hop_class = bo
Routing rule of class tm:
prob = 1. , hop_node= MED_STATE , hop_class = tm
Routing rule of class id:
prob = 1. , hop_node= MED_STATE , hop_class = id
Yank targets of class bor:
prob = 1. , from_node= MED_STATE, from_class = all
Yanked batch of class bor:prob=1., size=1, hop_class=id
Routing rule of class bor:
prob = 1. , hop_node= BACKOFF , hop_class = bo

Parameters for node BACKOFF

Queueing discipline: IS
Number of servers: infinite
Queue capacity: infinite
Transaction classes at node BACKOFF:
class = bo , priority = 0; class = cl , priority = 0
Service time of class bo:exponential (mean=0.000037)
Service time of class cl: prob = 1. , time = infinite
Routing rule of class cl:
prob = 1., hop_node= SENSE_STATE , hop_class = st1

Parameters for node TRANSMIT

Queueing discipline: FCFS
Number of servers: 1
Queue capacity: infinite
Transaction classes at node TRANSMIT:
class = xt1 , priority = 0; class = xt2 , priority = 0
class = xtm , priority = 0
Service time of class xt1: prob=1., time=0.000001

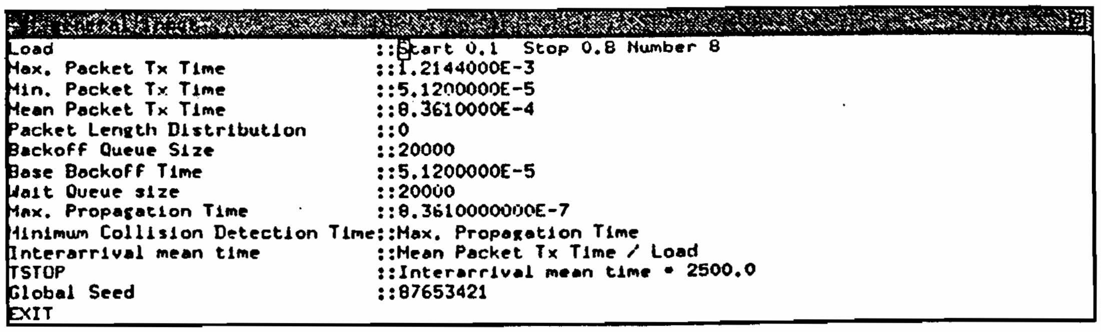
```
Routing rule of class  xt1:
   prob = 1. , hop_node= SET_STATE , hop_class = xt2
Service time of class  xt2: prob=1.,  time=0.000001
Routing rule of class  xt2:
   prob = 1. , hop_node= SENSE_STATE , hop_class = scl
Service time of class xtm: exponential(mean=0.000836)
Routing rule of class  xtm:
   prob = 1. , hop_node= SET_STATE , hop_class = rel

                 Parameters for node   SENSE_STATE
                      ------------------------------
Queueing discipline:  IM_CL_YANK
Number of servers: infinite
Queue capacity:  infinite
Transaction classes at node   SENSE_STATE:
   class = st1 , priority = 0; class = wt1 , priority = 0
   class = nid , priority = 0; class = scl , priority = 0
   class = t1 , priority = 0; class = id , priority = 0
   class = cl , priority = 0; class = wt2 , priority = 0
   class = xt1 , priority = 0
Arrival interval of class st1:exponential(mean=1.000000)
Yank targets of class  st1:
   prob = 1. , from_node= MED_STATE, from_class = id
Yanked batch of class  st1:prob=1. , size= 1 , hop_class = id
Routing rule of class  st1:
   prob = 1. , hop_node= SET_STATE , hop_class = xt1
Alternate routing rule of class  st1:
   prob = 1., hop_node= SENSE_STATE, hop_class = nid
Yank targets of class  wt1:
   prob = 1. , from_node= MED_STATE, from_class = id
Yanked batch of class  wt1:prob=1., size=1 , hop_class = id
Routing rule of class  wt1:
   prob = 1., hop_node= SENSE_STATE, hop_class = wt2
Yank targets of class  nid:
   prob = 1. , from_node= MED_STATE, from_class = t1
Yanked batch of class  nid:prob=1., size= 1 , hop_class = t1
Routing rule of class  nid:
   prob = 1. , hop_node= BACKOFF , hop_class = cl
Alternate routing rule of class  nid:
   prob = 1., hop_node= SENSE_STATE, hop_class = wt1
Yank targets of class  scl:
   prob = 1. , from_node= BACKOFF, from_class = cl
Yanked batch of class  scl:prob= 1. , size= 1 , hop_class = cl
Routing rule of class  scl:
   prob = 1. , hop_node= SET_STATE , hop_class = cd
Alternate routing rule of class  scl:
   prob = 1. , hop_node= SET_STATE , hop_class = xtm
Routing rule of class  t1:
   prob = 1. , hop_node= MED_STATE , hop_class = t1
```

```
Routing rule of class  id:
   prob = 1. , hop_node= MED_STATE , hop_class = id
Routing rule of class  cl:
   prob = 1. , hop_node= BACKOFF , hop_class = cl
Yank targets of class  wt2:
   prob = 1. , from_node= SET_STATE, from_class = xt1
Yanked batch of class  wt2:prob=1., size=1, hop_class= xt1
Routing rule of class  wt2:
   prob = 1. , hop_node= BACKOFF , hop_class = cl
Routing rule of class  wt2:
   prob = 1. , hop_node= SET_STATE , hop_class = xt1
Routing rule of class  xt1:
   prob = 1. , hop_node= SET_STATE , hop_class = xt1
```

FIGURE C.6
Parameters for the Q+™ State Transition model with exponential backoff

C.5 Parameters for the State Transition model with binary exponential backoff.

These parameters are displayed in figure C.7 for BONeS™.



```
Load                          ::Start 0.1  Stop 0.8 Number 8
Max. Packet Tx Time           ::1.2144000E-3
Min. Packet Tx Time           ::5.1200000E-5
Mean Packet Tx Time           ::8.3610000E-4
Packet Length Distribution    ::0
Backoff Queue Size            ::20000
Base Backoff Time             ::5.1200000E-5
Wait Queue size               ::20000
Max. Propagation Time         ::8.3610000000E-7
Minimum Collision Detection Time::Max. Propagation Time
Interarrival mean time        ::Mean Packet Tx Time / Load
TSTOP                         ::Interarrival mean time * 2500.0
Global Seed                   ::87653421
EXIT
```

FIGURE C.7
Parameters for the BONeS™ State Transition model with binary exponential backoff

C.6 SIMLAN™ Parameters

The following figure show the parameters used in the CSMA/CD model provided by SIMLAN™.

```
        *****  STATISTICAL  DISTRIBUTION  FUNCTIONS
STATISTICAL DISTRIBUTIONS =
 NAME = INTERARRIVAL TIME
  TYPE = EXPONENTIAL
   MEAN =     0.00139
 NAME = PACKET LENGTH
  TYPE = EXPONENTIAL
   MEAN =   8361.000
 NAME = I/ETHERNET RETRY SDF
  TYPE = IEEE.BACKOFF
   SLOT.TIME =    51.200
   RETRY.LIMIT =    16.000
   LIMIT.DELAY = +1.00E+006

          *****  LAN  ETHERNET  DEFINITION  BEGINS
HARDWARE TYPE = DATA TRANSFER
* LAN SUBTYPE IS IEEE 802.3 ETHERNET 10BASE5
 NAME = ETHERNET
  PROTOCOL = COLLISION
  CYCLE TIME = .100 MICROSEC
  RETRY INTERVAL = I/ETHERNET RETRY SDF
  COLLISION WINDOW = 8.361 MIC
  CONTENTION INTERVAL = 9.600 MIC
  JAM TIME = 3.2 MIC
  DRAW TYPE = BUS
  BITS PER CYCLE = 1 BITS
  CYCLES PER WORD = 8 CYCLES
  WORDS PER BLOCK = 1496 WORDS
  WORD OVERHEAD TIME = 0. MICROSEC
  BLOCK OVERHEAD TIME = 24. MICROSEC
  BUS CONNECTIONS =
   S1/STATION      S2/STATION       S3/STATION
   S4/STATION      S5/STATION       S6/STATION
   S7/STATION      S8/STATION       S9/STATION
   S10/STATION
```

FIGURE C.8
SIMLAN™ model parameters

C.7 GLAS Parameters

The parameters for the CSMA/CD system simulated on GLAS are presented in figure C.9.

```
                 Parameters for : MACCSM model.
        ---------------------------------------------------------------
   The lines up to "$" are reserved for comments. The comments may
be copied to files created from this file. Please do not remove these
lines or change the position of the "$" markers below. The data
consists of a 50-character description field, followed by a 20-character
value field. The fields are separated by two spaces.
        ---------------------------------------------------------------
$                   General Simulation Controls
Analyst's name.      ( up to 20 characters used )  U of K TISL
Project title.    ( up to 20 characters used )  CSMACD Model
Print SLAM input listing?              ( y/n )  N
Print SLAM echo summary report?          ( y/n )  N
Attempt execution?                   ( y/n )  Y
Print SLAM summary report?               ( y/n )  N
Confidence interval for output tabulation.  ( % )  95
Number of simulation subruns.        ( Integer )  10
Time for each subrun.                  ( uS )  3.484E+06
Set-up time as percent of run-time.      ( % )  10.00
Abort if an application layer queue fills? ( y/n )  N
$                    Media Access Parameters
Velocity of media as fraction of light speed.     880.0E-03
Line rate.                         ( Mbps )  10.00
Maximum number of collisions.          (integer)  16
Backoff limit.                    (integer)  10
Interframe spacing.                   ( uS )  6.000
Jam time.                          ( uS )  9.600
Slot time.                        ( uS )  51.20
Maximum packet size.                  ( bits )  50000
Minimum packet size.                 ( bits )  1
Number of overhead bits per packet.      ( bits )  1
$   Node # 1
Message length distribution.  ( CONSTANT, EXPON )  CONSTANT
-- Constant message length.            ( bits )  8360
Application layer buffer size.       ( messages )  20
Link distance.                    ( meters )  5.000
Bus distance to following node if appl. ( meters )  25.00
Interarrival time distribution.                EXPON (XMEAN)
-- Mean interarrival time.               ( uS )  13.94E+03
$   Node # 2
```

```
Message length distribution.   ( CONSTANT, EXPON )  CONSTANT
-- Constant message length.             ( bits )  8360
Application layer buffer size.      ( messages )  20
Link distance.                    ( meters )  5.000
Bus distance to following node if appl. ( meters )  25.00
Interarrival time distribution.               EXPON (XMEAN)
-- Mean interarrival time.              ( uS )  13.94E+03
$   Node # 3
Message length distribution.   ( CONSTANT, EXPON )  CONSTANT
-- Constant message length.             ( bits )  8360
Application layer buffer size.      ( messages )  20
Link distance.                    ( meters )  5.000
Bus distance to following node if appl. ( meters )  25.00
Interarrival time distribution.               EXPON (XMEAN)
-- Mean interarrival time.              ( uS )  13.94E+03
$   Node # 4
Message length distribution.   ( CONSTANT, EXPON )  CONSTANT
-- Constant message length.             ( bits )  8360
Application layer buffer size.      ( messages )  20
Link distance.                    ( meters )  5.000
Bus distance to following node if appl. ( meters )  25.00
Interarrival time distribution.               EXPON (XMEAN)
-- Mean interarrival time.              ( uS )  13.94E+03
$   Node # 5
Message length distribution.   ( CONSTANT, EXPON )  CONSTANT
-- Constant message length.             ( bits )  8360
Application layer buffer size.      ( messages )  20
Link distance.                    ( meters )  5.000
Bus distance to following node if appl. ( meters )  25.00
Interarrival time distribution.               EXPON (XMEAN)
-- Mean interarrival time.              ( uS )  13.94E+03
$   Node # 6
Message length distribution.   ( CONSTANT, EXPON )  CONSTANT
-- Constant message length.             ( bits )  8360
Application layer buffer size.      ( messages )  20
Link distance.                    ( meters )  5.000
Bus distance to following node if appl. ( meters )  25.00
Interarrival time distribution.               EXPON (XMEAN)
-- Mean interarrival time.              ( uS )  13.94E+03
$   Node # 7
Message length distribution.   ( CONSTANT, EXPON )  CONSTANT
-- Constant message length.             ( bits )  8360
Application layer buffer size.      ( messages )  20
Link distance.                    ( meters )  5.000
Bus distance to following node if appl. ( meters )  25.00
Interarrival time distribution.               EXPON (XMEAN)
-- Mean interarrival time.              ( uS )  13.94E+03
$   Node # 8
Message length distribution.   ( CONSTANT, EXPON )  CONSTANT
```

```
-- Constant message length.            ( bits )  8360
Application layer buffer size.       ( messages )  20
Link distance.                     ( meters )  5.000
Bus distance to following node if appl. ( meters )  25.00
Interarrival time distribution.                EXPON (XMEAN)
-- Mean interarrival time.              ( uS )  13.94E+03
$   Node # 9
Message length distribution.   ( CONSTANT, EXPON )  CONSTANT
-- Constant message length.            ( bits )  8360
Application layer buffer size.       ( messages )  20
Link distance.                     ( meters )  5.000
Bus distance to following node if appl. ( meters )  25.00
Interarrival time distribution.                EXPON (XMEAN)
-- Mean interarrival time.              ( uS )  13.94E+03
$   Node # 10
Message length distribution.   ( CONSTANT, EXPON )  CONSTANT
-- Constant message length.            ( bits )  8360
Application layer buffer size.       ( messages )  20
Link distance.                     ( meters )  5.000
Bus distance to following node if appl. ( meters )  25.00
Interarrival time distribution.                EXPON (XMEAN)
-- Mean interarrival time.              ( uS )  13.94E+03
```

FIGURE C.9
GLAS model parameters