1-1-1989

Simulation Networking Protocol Alternatives: Final Report

Michael Georgiopoulos

INSTITUTE FOR SIMULATION AND TRAINING

FINAL REPORT

SIMULATION NETWORKING

PROTOCOL ALTERNATIVES

DR. MICHAEL GEORGIOPOULOS

August 1, 1988–July 31, 1989

IST

B 103

FINAL REPORT

CONTRACT
N61339-88-G-002 ORDER 0008

SIMULATION NETWORKING PROTOCOL ALTERNATIVES

PRINCIPAL INVESTIGATOR
DR. MICHAEL GEORGIOPOULOS

1 AUGUST 1988 THRU 31 JULY 1989

TABLE OF CONTENTS

1. INTRODUCTION

A local area network (LAN) is a geographically confined communication system that uses a shared transmission medium. Various choices usually exist for the main ingredients of a LAN (i.e., transmission medium, topology, access protocols, etc.) with each exhibiting advantages and providing benefits that depend on the objectives of the LAN. The ability to model, analyze and evaluate the impact of these choices on network performance is essential to ensuring maximum utilization of the LAN.

One of the pioneering LANs for connecting computers was a bus-based ETHERNET developed by Xerox Corporation in the early 1970s. The contention access method used by each node in ETHERNET is based on a pre-emptive protocol of first listening for network activity and then broadcasting the message onto the network. If a collision with another message occurs, each sender (node) backs-off from transmitting its message for a random period of time and then attempts the transmission again. This access technique is known as Carrier Sense Multiple Access with Collision detection (CSMA/CD) [1,2]. Standards for CSMA/CD protocols such as ETHERNET are known as IEEE 802.3 standards [1].

The networking of real-time, interactive simulation training systems departs from the traditional use of a computer network whose function is to provide sharing of computing resources among multiple users (nodes) on the network. When used to interconnect real-time training simulators, the network is used almost exclusively for communication of process state information between the simulators engaged in the training exercise.

There are many inherent limitations to using a network in this application. For example, as the number of simulators on the network and the workload per simulator increases, there may be a deterioration in throughput and an increase in message delays. If message delays become too large, for example, the effectiveness of a real-time training simulation may be overly compromised due to the time-critical response requirements in the simulation of true-to-life, action requiring training scenarios. Depending

upon the network communication protocol being used, there may also be an increase in the percentage of lost messages.

Recently, there has been a tremendous interest in LANs implemented using the non-contention class of network protocols known as token-passing protocols [3,4]. Two schemes falling under this class are Token-Ring and virtual Token-Bus protocols. In a Token-Ring LAN, a distinctive bit sequence, called a token, is passed from one node to another in order to signify the availability of the network medium for the transmission of data for the node. Possession of the token by the node gives it, and only it, permission to transmit across the network, as opposed to having all nodes contend for this privilege. In a Virtual Token-bus LAN, a virtual, or imaginary token, is passed from node to node thus providing access to the network. This virtual token is actually a predetermined instant in time when each node knows if it is its turn to access the network.

The primary goal of this research effort has been to develop predictive/analytical models for network performance of two LAN configurations operating under real-time, interactive simulation/training constraints. These LANs are bus networks which utilize baseband transmission to send messages over a coaxial cable that is common to all nodes. The medium access control scheme for the first is ETHERNET, which is a member of the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol family and for the second is the Generalized Broadcast Recognizing Access Method (GBRAM) [3], which is a member of the Virtual Token-Bus protocol family. Discussions of network performance for these two protocols, the implications of the results of a comparison study and the insight gained from this project for improving real-time simulation networking are presented in this report.

2. NETWORK SYSTEM CONFIGURATION MODEL

The bus network configuration for ETHERNET and GBRAM is shown in Figure 1. In this implementation, up to eight nodes can be connected through a multi-port transceiver to a single point on the coaxial cable, via a media-access unit. A single coaxial cable links all nodes together.

3. NETWORK MEDIUM-ACCESS PROTOCOLS

3.1 ETHERNET

CSMA/CD protocols, including ETHERNET, are characterized by their distributed network control, whereby each node on the network determines its own channel access time based only on information available from the common network channel (bus). When a node is ready to transmit a message onto the network, it first monitors the network bus to determine whether any other transmissions from other nodes are in progress. If the node senses the network channel to be busy, it simply waits for the channel to become idle before attempting to transmit its message. Once the channel is sensed to be idle, the node waits a pre-specified amount of time to assure the channel is clear and then begins transmitting its message. During its own transmission, the node also monitors the channel in order to detect whether its message is interfering (colliding) with messages from other nodes. If a collision is detected, each node involved in the collision transmits a bit sequence onto the network known as a jam signal, after which each node in the collision waits (backs-off) for a randomly generated amount of time before re-attempting its transmission. A complete description of the ETHERNET protocol can be found in [1].

3.2 Generalized Broadcast Recognizing Access Method (GBRAM)

The GBRAM protocol is a member of the Token passing protocol family. It differs significantly, however, from a Token-Ring protocol. In the GBRAM, rather than each node having to capture the free token from the network to gain transmission access, an imaginary (virtual) token is passed from node to node achieving the same result (i.e., contention-free access). The virtual token scheme provides each node access to the network at a unique time instant, which is determined by a decentralized scheduling function.

To understand the operation of the GBRAM better we need to introduce some notation with the aid of Figure 2. The nodes that are connected onto the network via the same multi-port transceiver belong to the same group. As we see, from Figure 2, the identity of a node is determined by an ordered pair of indices. The first index corresponds to the group identity of the node

and the second index corresponds to the identity of the node within a group. For example, node (i,j) is the node with group index i and identity j within a group. Group indices and node indices are assigned to nodes in the network with increasing order from left to right (see Figure 2). Taking into consideration that at most eight nodes can be attached at the same point on the cable via the multi-port transceiver, N nodes can be accommodated into at least $M=\lceil N/8 \rceil$ groups. For example, when N=100, one way of distributing the nodes in the network is in twelve groups of eight nodes and one group of four nodes. Let us finally denote, in a network of N nodes and M groups, by $N_i (1 \leq i \leq M)$, the number of nodes in group i and, by $D(i_1,j_1 ; i_2,j_2)$, the propagation delay between nodes (i_1,j_1) and (i_2,j_2).

Every node in the network perceives the channel state, under GBRAM as consisting of cycles of scheduling and transmission periods (see Figure 3). Roughly speaking, the end of a transmission period designates the beginning of a scheduling period and the end of a scheduling period signals the beginning of the next scheduling period. During a scheduling period every node gains the right to access the channel starting with the node whose index is next to node who transmitted last. The first scheduling period starts at the beginning of time and the node who is allowed to access the channel first is node (1,1).

GBRAM avoids collisions by scheduling different nodes at unique time instances. The time interval between two successive scheduling instances depends on the geographical location of the nodes that are allowed to access the channel at these instances. In fact, it is equal to the propagation delay between the nodes which are scheduled to transmit at these instances. To be more specific, let us denote by t a common time instant known to all the nodes in the network. Let us assume that node (i_1, j_1) is the node who transmitted last on the channel prior to time t. GBRAM then schedules node (i_1,j_1+1) at time t, node (i_1,j_1+2) at time $t + D(i_1, j_1+1 ; i_1, j_1+2)$ and so on (see Figure 4). Furthermore, under GBRAM, node $(i_1+1,1)$ is scheduled to transmit $D(i_1,N_{i_1} ; i_1+1,1)$ units of time after the last node in group i_1 (i.e. node (i_1, N_{i_1})) is allowed to access the channel. Other nodes are scheduled accordingly.

A complete scheduling cycle starting at time t is shown in Figure 5. It is worth mentioning that, under GBRAM, if a node is scheduled at a particular time instant and it sees the channel busy, it withholds and waits until the transmission is over to reschedule.

Provided that the beginning of scheduling periods are known to all the nodes and the node scheduling instances are chosen as above, GBRAM avoids collisions altogether. In the next section, we show that the nodes, can easily determine the beginning of scheduling periods. In Appendix A, we present a more formal description of the GBRAM protocol.

4. DISCUSSION

We expect that the GBRAM will outperform ETHERNET for medium to high input traffic loads but it will be inferior to ETHERNET for light to medium input traffic loads. We will try to justify our claim by presenting two simple arguments.

4.1 Argument 1

We assume that in a network of 100 nodes only one node is active (i.e., it generates packets into the network), while the other 99 nodes are silent. Furthermore, we assume that the active node, after the successful transmission of its packet, generates another packet, with probability one, uniformly distributed in an interval equal to a complete scheduling cycle (see Figure 5). Hence, the active node has, at all times, at most one packet for transmission in its buffer. We also assume that the propagation delay between two nodes in the same group is 30 bits and the propagation delay along the cable is 20 bits. This is a case of light input traffic. If we take the packet length to be equal to 1000 bits, then it is easy to see that GBRAM induces an average and a maximum packet delay of approximately (30 x 100 + 40) / 2 +1000 = 2,520 bits and (30 x 100 + 40) + 1000 = 4,040 bits, respectively. Every contention protocol, and consequently ETHERNET, induces under the aforementioned light input traffic conditions an average and a maximum packet delay of approximately 1000 bits. As the number of nodes in the network increase, while the above input traffic scenario remains valid (i.e. one node out of the total number of nodes is active), the

performance difference between GBRAM and ETHERNET widens, due to the fact that the input traffic becomes lighter.

4.2 Argument 2

We assume now that in a network of 100 nodes all of them are active. Each active node, after the successful transmission of its packet, generates another packet, with probability one, uniformly distributed in the interval between the node's successive scheduling instances. As before, the propagation delay between two nodes in the same group is equal to 30 bits and the propagation delay along the cable is 20 bits. This is a case of high input traffic load. If we take the packet length to be equal to 1000 bits, we can show that, under GBRAM, the network experiences an average and a maximum packet delay of approximately **(100 x 1000)/2 + 1000 = 51,000** bits and **(100 x 1000) + 1000 = 101,000** bits, respectively. The aforementioned input traffic load is approximately equal to 100%. ETHERNET [5] and other contention protocols attain a throughput smaller than 100% even under ideal conditions (i.e., small end-to-end propagation delay vs packet length ratio). As a result, ETHERNET and other contention protocols will induce unreasonably large or even unbounded delays for the above high input traffic scenario.

Arguments 1 and 2, although relatively simple, verify our expectations that there will be a region of input traffic loads (light to medium) where ETHERNET (or other contention protocols) outperform GBRAM and another region of input traffic loads (medium to high) where GBRAM is superior to ETHERNET (or other contention protocols). The cut-off point (i.e., the input traffic at which GBRAM becomes better) depends on the total number of nodes in the network and increases as the number of nodes in the network increases. The exact cutoff point will be determined by simulations.

In the previous section, it was evident that the successful operation of GBRAM depended (at least for the version of the GBRAM presented in this paper) on the fact that all nodes know a common time epoch. The common time epoch corresponds to the beginning of a scheduling period. In our version of the GBRAM, a scheduling period corresponds to either the end of

a transmission period (as perceived by the transmitting user) plus the propagation delay along the cable or a complete scheduling cycle after the beginning of the previous scheduling period (see Figure 5). It is obvious that the common time epoches can be determined by any node, who observes the channel state at all times and knows its propagation delay from any other node in the network. Note that most contention protocols require only that nodes observe the state of the channel at all times. Nevertheless, other versions of GBRAM are described in [3] that do not require the nodes to have a complete knowledge of the network topology. These versions of GBRAM , however, will not perform as well as the GBRAM version presented in the previous section and in Appendix A.

It has been observed that in large scale networks not all users are active at all times. Consider, for example, a tank simulator which is active at the beginning of a battle, but it is destroyed by enemy fire during the simulation. These inactive nodes must be taken out of the token passing sequence (list) so as to reduce the number of wasted idle slots. Hence, we must devise a procedure to sign-off inactive nodes from the network. This procedure is rather straightforward. An active node signs off by broadcasting at its scheduled to transmit time instant a sign-off packet. All other active nodes read the sign-off packet and update their scheduling sequence accordingly.

5. SIMULATION MODEL - EXPERIMENTAL RESULTS & OBSERVATIONS

The network configuration is shown in Figure 1. A single coaxial cable links all the nodes together. The capacity of the cable is 10Mb/s. The performance of the ETHERNET and GBRAM protocol depend on a number of important parameters listed below.

i) The time that it takes for a node to recognize that the channel is idle (I)

ii) The time that it takes for a node to recognize that the channel is busy (B)

iii) The time that it takes for a node to recognize that a collision is in progress on the channel (C)

iv) The end-to-end propagation delay (x) (i.e., the time that it takes for a message to travel from one node at the one end of the cable to another node at the other end of the cable)

v) The duration of the jam signal sent by a colliding node (J)

vi) The time unit used by the back-off algorithm (R)

vii) The time interval between two successive packet transmissions (IG)

viii) The length of the packet (PL)

ix) The number of nodes in the network (N)

x) The way the MAU units are distributed along the cable

xi) The total input traffic generated by the active nodes

For the results that we are going to present we took the parameter values specified in [1]. In particular, for the network configuration of Figure 1 we have I=30.14 bits, B=14.14 bits, C = 28.14 bits, x = 52.93 bits, J=48 bits, R=512 bits and IG=96 bits. Furthermore, we chose PL=1024 bits, N=100 or 400, the MAU units uniformly distributed along the cable and the traffic generated by all the nodes to be a Poisson process.

There is one additional node in the network that generates traffic in a different manner than the other nodes. This node is referred to as the Management Command and Control (MCC) unit and simulates a variety of combat service and support vehicles. The number of vehicles that MCC simulates is taken to be equal to the number of nodes (N) in the network. Every second the MCC unit sends one update message (i.e., packet) for one fifth of the vehicles that it simulates. Due to its unique nature, the MCC

unit is treated differently than the other nodes. Every time the MCC unit gains access into the channel it is allowed to transmit ten packets at once.

Other choices for the parameters specified above are possible but the ones chosen provide us with the capability to conduct a detailed comparison between the GBRAM and the ETHERNET protocols. Figure 6 shows the delay versus traffic load performance for the GBRAM and ETHERNET protocols with N=100. We observe from Figure 6 that for light traffic load ETHERNET induces a delay approximately equal to the packet transmission time, i.e., there is almost no contention delay for access to the network. As the traffic load increases to medium loads the delay rises to several times the packet transmission time owing to collisions and the associated back-offs. While a node is incurring a back-off delay, it is not contending for network access. Thus, larger delays effectively reduce the instantaneous offered load and help maintain stability. Nevertheless, as the input traffic increases above a certain point, we observe an abrupt increase in the delay due to the fact that at high loads most nodes have more than one packet at a time awaiting transmission. While the discarding of packets feature of the ETHERNET protocol will guarantee relatively reasonable delays for the packets that are first in the queue, the second or third packet in the queue will experience even larger delays. This results in the "blow-up" behavior of the ETHERNET protocol, once the traffic load exceeds a certain limit. On the other hand, the GBRAM protocol exhibits a much more rational behavior. For light traffic loads, GBRAM induces a delay larger than the packet transmission time due to the fact that a packet may arrive at a node site before its scheduling instance comes up. As expected, GBRAM is worse than ETHERNET for light traffic load. As the traffic increases, the GBRAM performance becomes comparable to ETHERNET. For high traffic load GBRAM incurs relatively small delays and it outperforms ETHERNET. This is because the deterministic nature of the GBRAM avoids collisions altogether. As a result, the channel is either idle or busy with successful transmissions. At high loads, all nodes are active most of the time. Hence, the channel is almost entirely occupied by successful packet transmissions, allowing us to accommodate a traffic load close to 100%. It is worth noting that at traffic load of nine thousand packets/sec, GBRAM induces a delay smaller than

ETHERNET at traffic load of six thousand packets/sec. The cut off point (i.e., the traffic load at which GBRAM becomes better than ETHERNET occurs at the traffic load of approximately 4.5 thousand packets/per sec. But even for traffic loads below the cutoff point, GBRAM exhibits a reasonable performance (i.e., delays smaller than 0.4 ms)

Figure 7 shows the delay versus traffic load performance for the GBRAM and ETHERNET protocols with N=400. Similar observations, regarding performance comparisons between ETHERNET and GBRAM, can be drawn for Figure 7. We can see, from Figure 7, that as the number of nodes in the network increases the performance of GBRAM and ETHERNET deteriorates. For ETHERNET this is attributed to the fact that the same input load is distributed among a larger number of nodes resulting in a higher probability of collisions. For GBRAM the deterioration in performance stems from the fact that it takes longer to schedule 400 nodes than to schedule 100 nodes. Finally, the cutoff point, in Figure 7, occurs later than in Figure 6. In particular, with 400 nodes in the network GBRAM outperforms ETHERNET for traffic load above five thousand packets/sec. This behavior was predicted in the discussion of the GBRAM and ETHERNET protocols in Section 4.

In Figures 8 and 9 we show the histograms of the delay distribution for the GBRAM and ETHERNET protocols, with 400 nodes at traffic loads of, three thousand and six thousand packets/sec, respectively. The labels on the X-axis indicate the upper limit of each bin. The leftmost bin includes packets with delay $\leq$0.5 ms, the next bin, packets with delay >0.5 and $\leq$1.0 and so on. We observe from Figure 8 and 9 that most of the ETHERNET packets experience a delay $\leq$0.5. Actually, 93% and 65% of the ETHERNET packets experience a delay smaller than 0.5ms at loads of three and six thousand packets/sec, respectively. The corresponding numbers for the GBRAM protocol are 21% and 12%. On the other hand, we see that at a traffic load of six thousand packets/sec only 2% of the GBRAM packets experience a delay larger than 5ms, while 13% of the ETHERNET packets experience delays larger than 5ms. Figures 8 and 9 show the GBRAM packet delays are more evenly distributed than ETHERNET packet delays. ETHERNET packet delays (at high loads) are either very small or very large. Similar

conclusions, can be drawn from Figure 10 and 11, where we show the histograms of the delay distribution for the GBRAM and ETHERNET protocols with 100 nodes at input loads of three thousand packets/sec and six thousand packets/sec, respectively.

In Figures 12 and 13 we show the percentage of lost traffic for the ETHERNET and the GBRAM protocols in a network of 100 and 400 nodes, respectively. GBRAM by construction does not discard any packets. ETHERNET discards packets after 16 unsuccessful attempts to access the channel (i.e., after 16 collisions). As we see, from Figures 12 and 13, ETHERNET starts discarding packets at traffic loads above six thousand packets/per sec. In particular, at traffic load of 7.5 thousand packets/sec and 400 nodes in the network, the percentage of lost traffic is unacceptably high (~7%). This is another indication that GBRAM outperforms ETHERNET for high traffic loads.

Let us now discuss a variation of the ETHERNET protocol we examined which demonstrated better performance than ETHERNET in our simulations. One of the limitations of the ETHERNET protocol is that every colliding node with a back-off delay (specified by the back-off algorithm) decreases its back-off delay (to determine the instant of its next transmission attempt) independently of whether the channel is idle or busy. A variation of ETHERNET, which we named ETHERNET-1, allows the colliding nodes to decrease their back-off delay only during the time intervals that the node senses the channel idle. In Table 1, we show the average packet delay in ms and percentage of lost traffic versus traffic load for the ETHERNET-1 protocol, in a network of 100 nodes. Similar results are shown in Table 1 for the ETHERNET protocol. As we see, from Table 1, ETHERNET-1 out performs ETHERNET for traffic loads up to 7.5 thousand packets/sec. At a traffic load of 7.5 thousand packets/sec ETHERNET-1 experiences a little bit higher than ETHERNET average packet delay (~3ms) but accommodates more traffic into the network (~1.26%).

TRAFFIC LOAD	ETHERNET-1		ETHERNET	
(1000 packets/sec)	Mean Delay (ms)	% of Lost Traffic	Mean Delay (ms)	% of Lost Traffic
1.52	0.137	0	0.137	0
3.02	0.172	0	0.179	0
4.52	0.271	0	0.444	0
6.02	0.982	0	5.9	0.038
7.52	60.9	0.012	57.1	1.271

TABLE 1. Performance Comparison Between ETHERNET-1 AND ETHERNET (100 nodes)

As we mentioned in the introduction the networking of real-time, interactive simulation training systems departs from the traditional use of a computer network. When used to interconnect real-time training simulators, the network is used almost exclusively for communication of process state information between the simulators engaged in the training exercise. In this context, a new packet at a node site, carrying the most current state of the node, can replace an old packet, not yet successfully transmitted, that represents a now outdated state condition. We utilized this approach in an implementation of the GBRAM protocol, named GBRAM-1. Our results are shown in Table 2, for a network of 100 nodes. In the same Table we included the GBRAM results for comparison purposes. We observe from Table 2 that GBRAM-1 induces average packet delays which are smaller than GBRAM, at the expense of increased percentage of lost traffic. The average packet delay improvement of GBRAM-1 over GBRAM is not significant considering the fact that at high loads (nine thousand packets/sec) GBRAM-1 induces an unacceptably high percentage of lost traffic (~9%).

TRAFFIC LOAD	GBRAM-1		GBRAM	
(1000 packets/sec)	Mean Delay (ms)	% of Lost Traffic	Mean Delay (ms)	% of Lost Traffic
1.52	0.318	0.106	0.319	0
3.02	0.363	0.43	0.368	0
4.52	0.436	1.06	0.451	0
6.02	0.554	2.26	0.609	0
7.52	0.754	4.46	0.975	0

TABLE 2. Performance Comparison Between GBRAM-1 and GBRAM (100 nodes)

Other variations of the ETHERNET protocol might be worth examining. For example, ETHERNET and ETHERNET-1 protocol variations where we discard packets as in GBRAM-1 are suitable candidates for investigation. In particular, ETHERNET with discarding of packets is anticipated to give us a much better delay performance at high loads than ETHERNET, but at the expense of some increase in the percentage of lost traffic.

6. CONCLUSIONS

We now make some conclusive remarks regarding the performance comparisons between GBRAM and ETHERNET conducted in the previous sections.

C.1) ETHERNET outperforms GBRAM for light to medium traffic loads (i.e., less than 4.5 thousand packets/sec)

C.2) GBRAM outperforms ETHERNET for medium to high traffic loads (i.e., above 4.5 thousand packets/sec)

C.3) The performance of both GBRAM and ETHERNET deteriorates as the number of nodes on the network increases.

C.4) GBRAM exhibits good performance over the whole range of input traffic loads (i.e., even for light to medium traffic load).

C.5) ETHERNET performance deteriorates drastically for input loads above a certain threshold (i.e., for traffic loads above six thousand packets/sec).

C.6) GBRAM packets experience delays over a wide range of reasonable delay values.

C.7) ETHERNET packets experience either very small delays or relatively large (compared to GBRAM) delays.

C.8) MCC packets, under GBRAM and ETHERNET, experience larger delays than the node packets, given that we adhere to the policy postulated in Section 5, specifically that the MCC transmits at most ten packets at once when it gains access to the channel (see Table 3 below).

TRAFFIC LOAD (1000 packets/sec)	GBRAM (MCC) Mean Delay (ms)	ETHERNET (MCC) Mean Delay (ms)
1.58	10.66	4.19
3.08	12.61	4.3
4.58	15.46	4.43
6.08	20.09	14.7

TABLE 3. Average Delay per MCC Packet Under GBRAM and ETHERNET (400 nodes)

C.9) If we let the MCC unit transmit more than ten packets at once, we will significantly decrease the MCC packet delays at the expense of a relatively minor increase of the node packet delays.

C.10) A variation of ETHERNET which discards packets and decreases the back-off delays only during idle channel instances will exhibit a better performance than ETHERNET.

C.11) A variation of ETHERNET where we choose the unit of retransmission delays (back-off delays) in an optimal way will be superior than ETHERNET.

It is worth noting that conclusions C.9 through C.11 indicate topics for further research. The most important conclusion of our work is that deterministic protocols such as GBRAM, and for similar reasons, token ring protocols, outperform ETHERNET. If our objective is to design a local area network that supports communication among simulators which generate at most six thousand packets/sec then the ETHERNET protocol is the right choice. This is hardly the case though, especially at times, when interconnecting different local area network sites is seriously considered.

Hence, protocols that achieve higher throughput than ETHERNET should be carefully investigated. The GBRAM and token ring protocols attain higher throughput than ETHERNET. As a result, careful examination of these protocols and issues related with their implementation to support real-time simulation networking should be addressed.

7. ACKNOWLEDGMENTS

I would like to thank Dr. Mostafa Bassiouni, Jack Thompson, Nicos Christou and Jorge Cadiz for their techincal support during the course of this project. I would also like to thank Mr. Ernie Smart for his guidance and administrative assistance. Without their help, the successful completion of this project would have not been possible

8. REFERENCES

[1] ANSI/IEEE - International Standard 8802/3 "Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specification " IEEE Computer Society Press, 1985.

[2] R.M. Metcalfe and D.R. Boggs, "Ethernet Distributed Packet Switching for Local Computer Networks," Communication Ass. Comput. Mach., Vol. 19, no 7, pp. 395-403, 1976.

[3] T.T. Liu, L.Li and W.R. Franta, "A Decentralized Conflict-Free Protocol, GBRAM for Large Scale Local Networks," Computer Network Symposium Proceedings, pp. 39-54, Dec. 1981.

[4] ANSI/IEEE - International Standard 8802/5 "Token Ring Access," IEEE Computer Society Press, 1985.

[5] T.A. Gonsalves, "Measured Performance of the Ethernet," Advances in Local Area Networks, edited by K. Kummerle, J.O. Limb and F.A. Tobagi, IEEE Press 1987, pp 383-410.

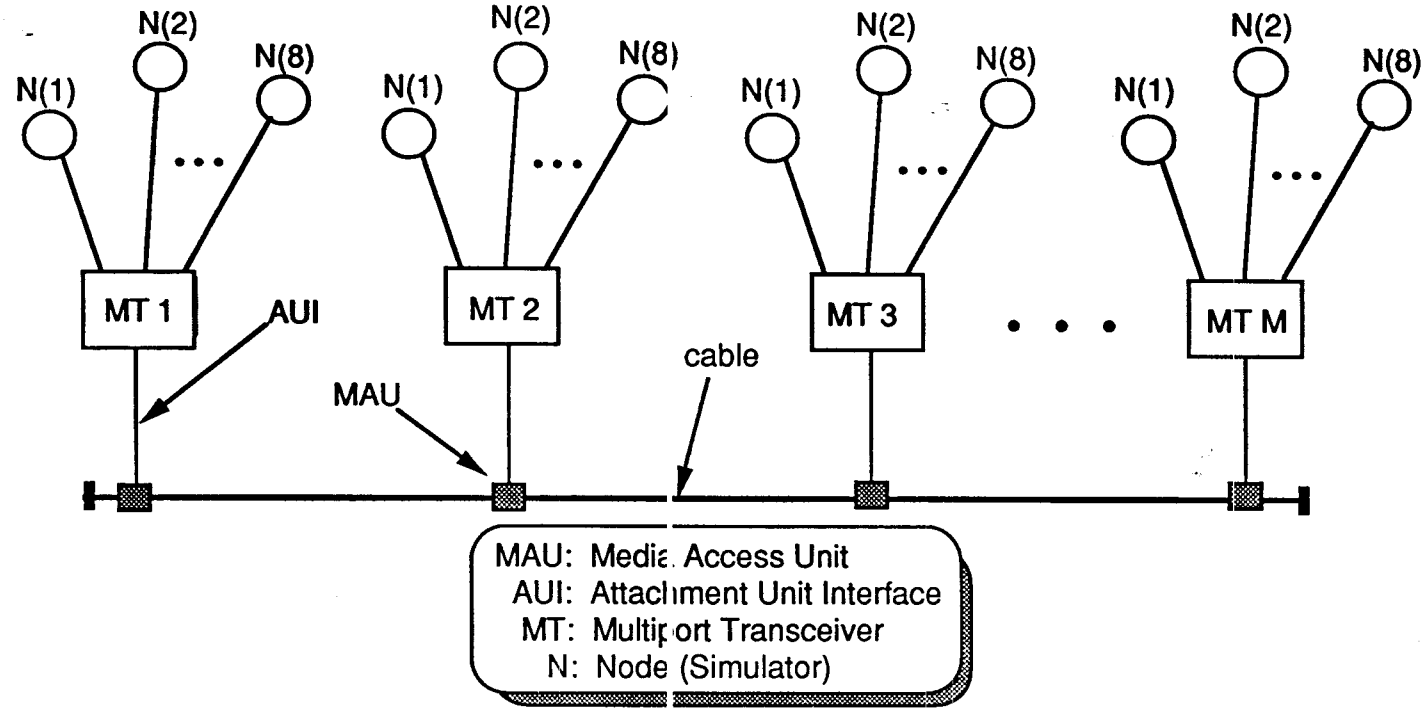


FIGURE 1: Bus Network Topology System Configuration

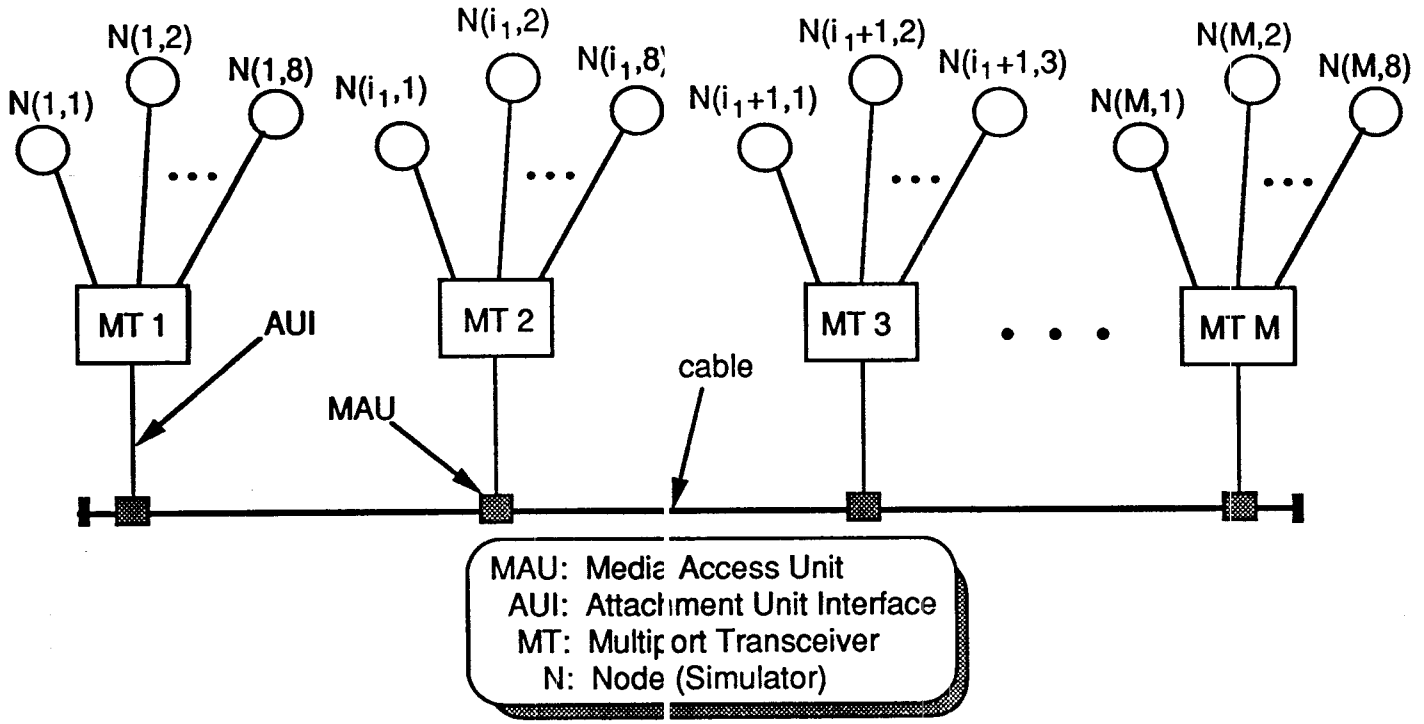


FIGURE 2: Node Identities in the GBRAM Protocol

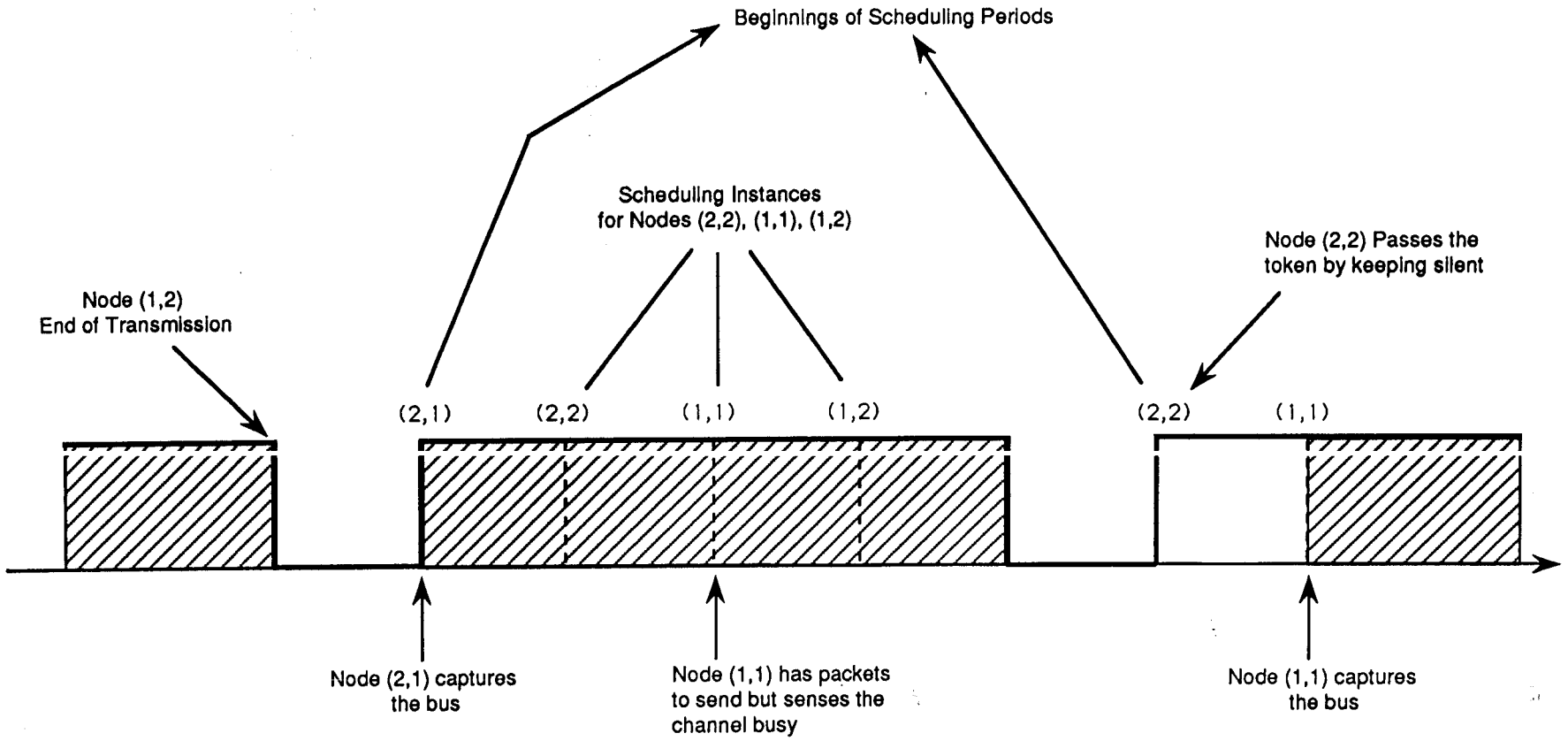


FIGURE 3: Channel Scheduling for the GBRAM Protocol in a Network of four Nodes (i.e. Nodes (1,1), (1,2), (2,1), (2,2)).

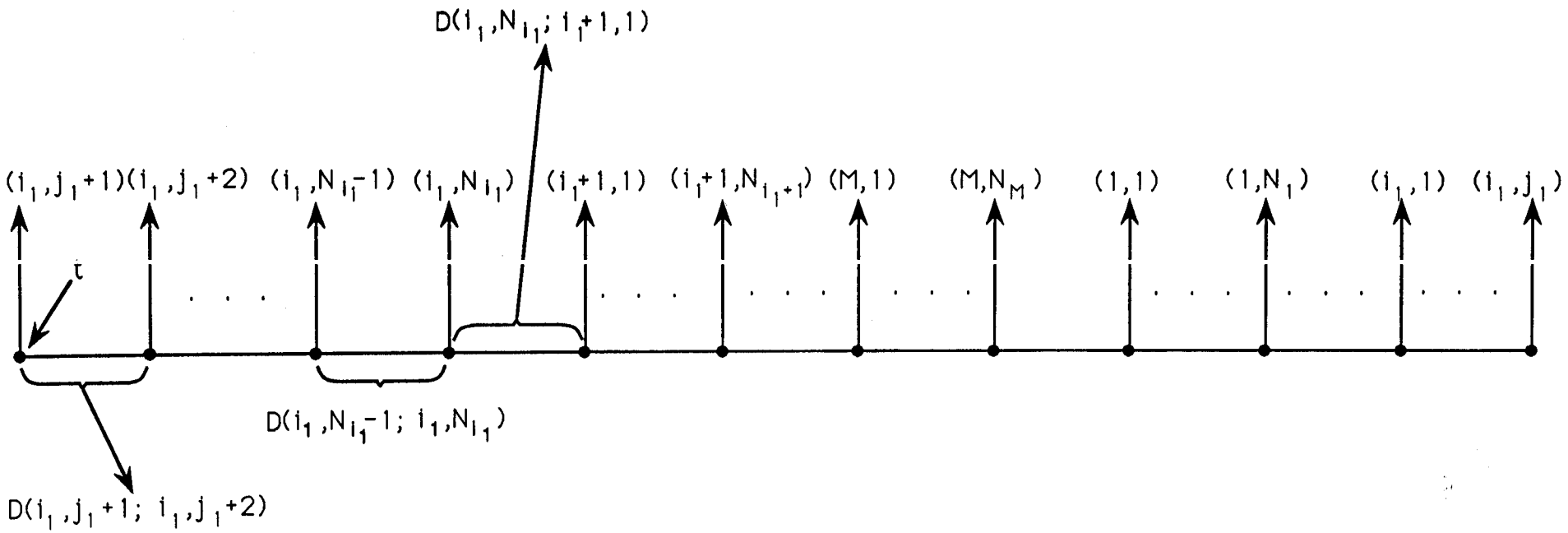


FIGURE 4: Channel Scheduling Instances for the GBRAM Protocol

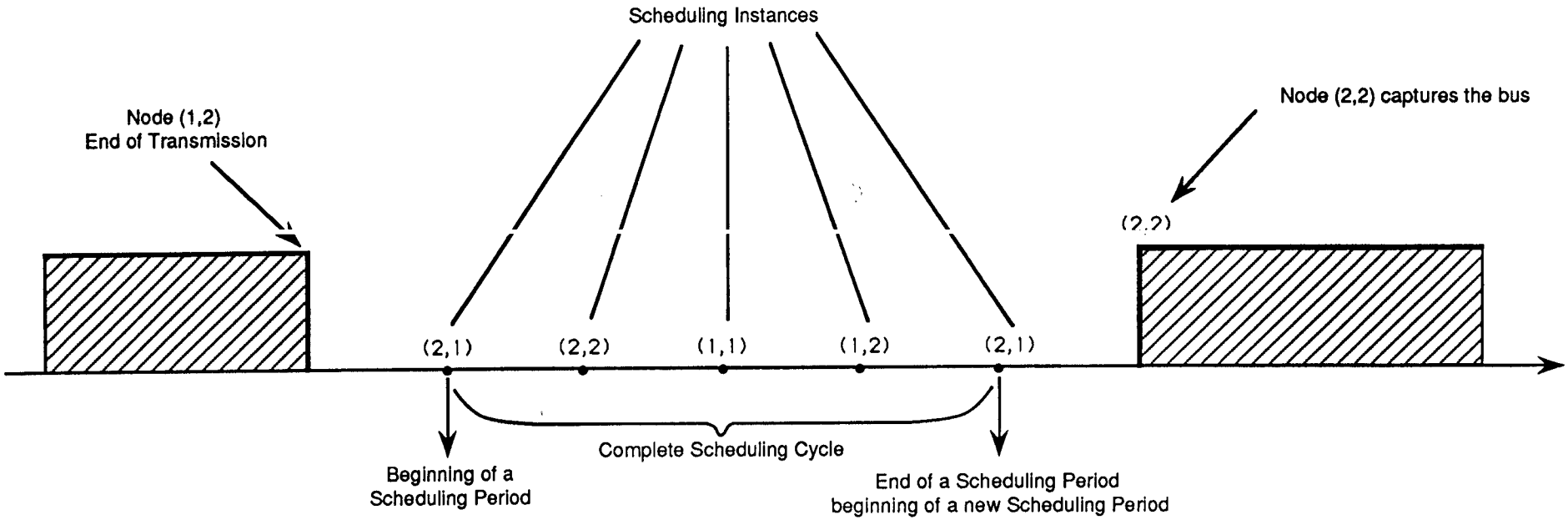


FIGURE 5: Channel Scheduling for the GBRAM Protocol in a Network of four Nodes
(i.e. Nodes (1,1), (1,2), (2,1), (2,2))

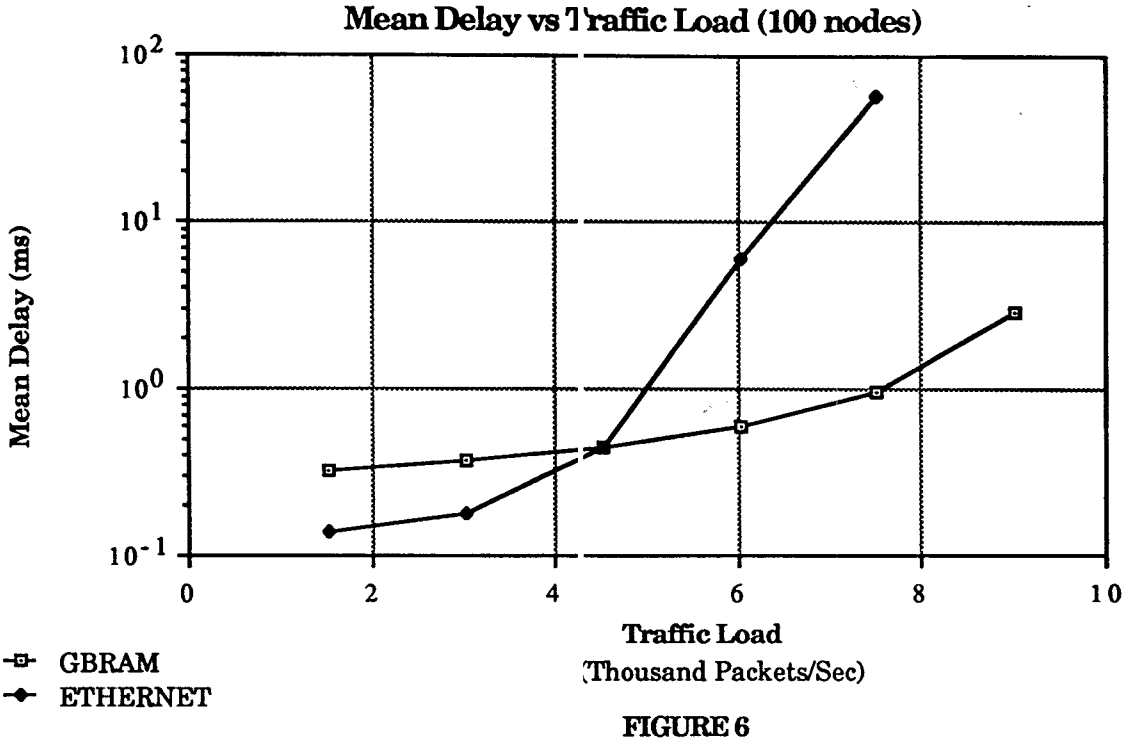


FIGURE 6

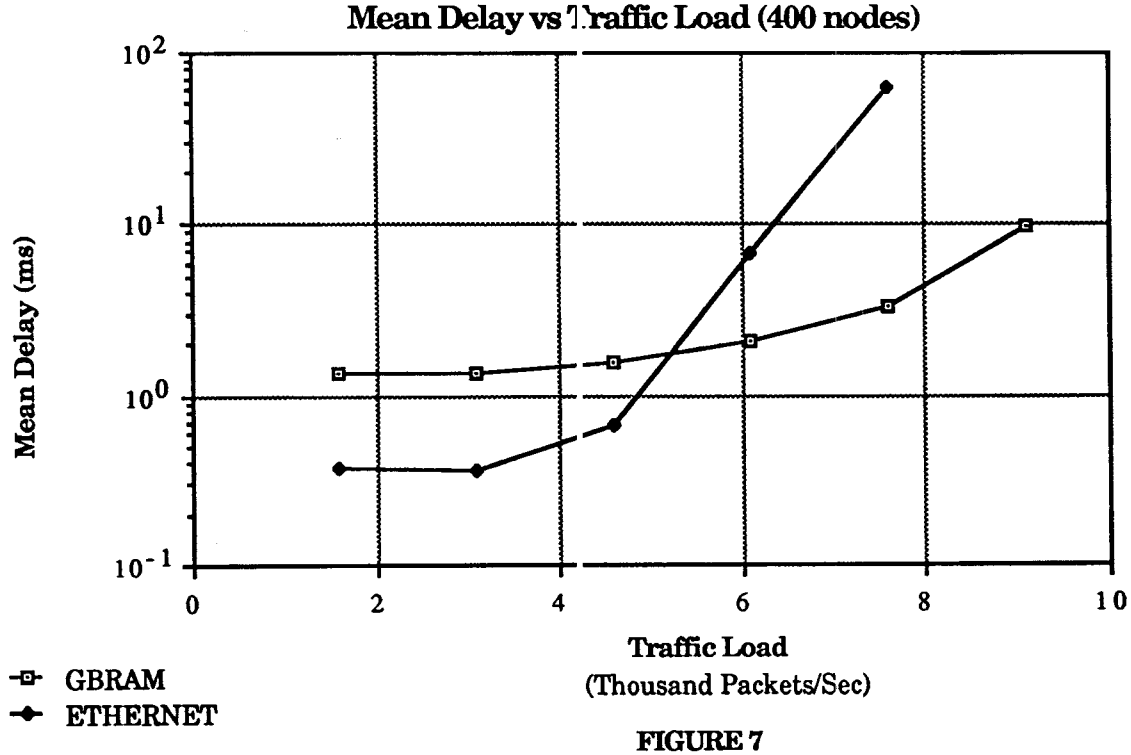


FIGURE 7

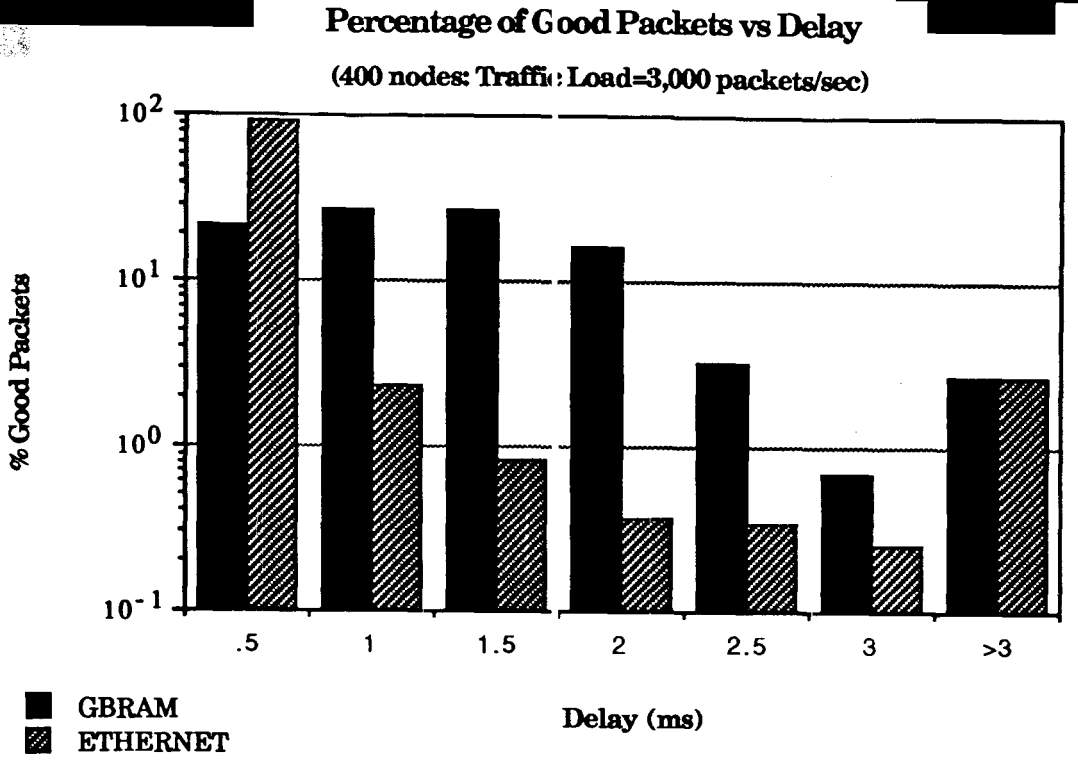


FIGURE 8

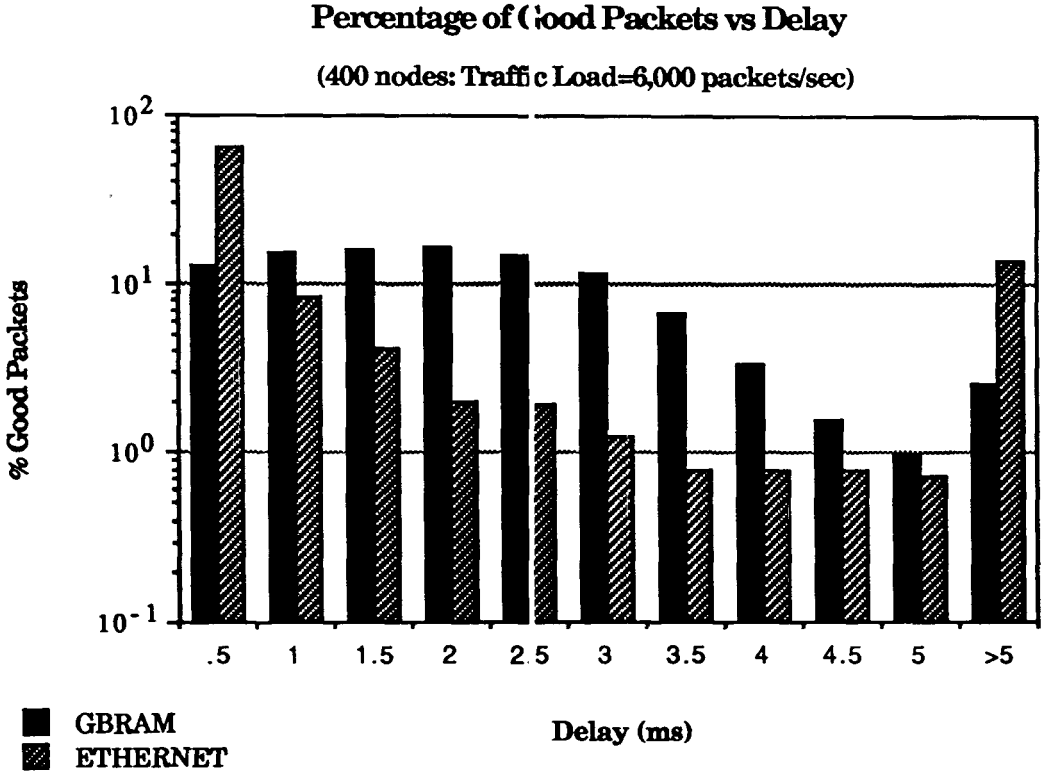


FIGURE 9

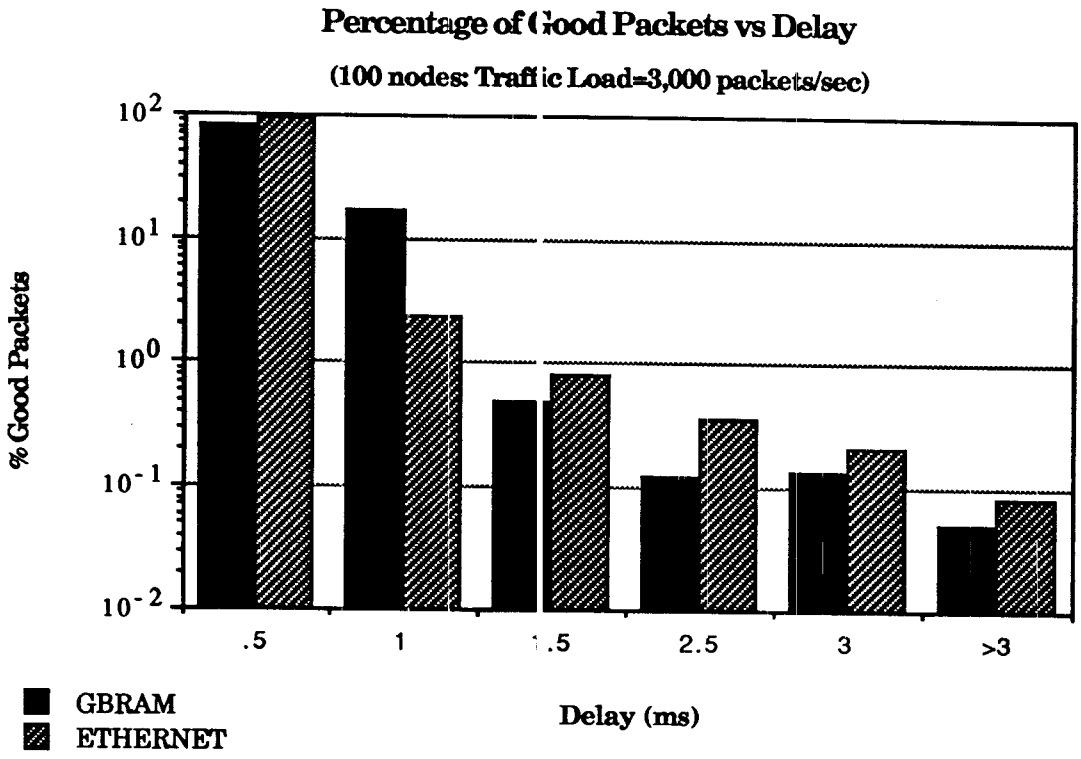


FIGURE 10

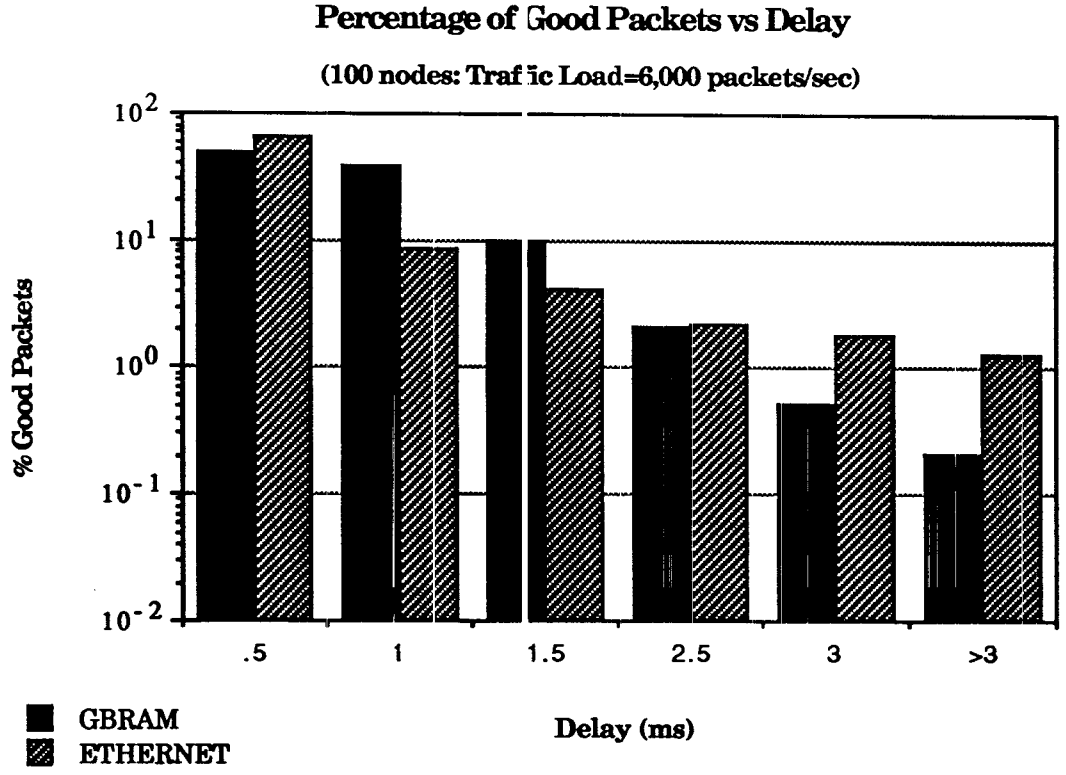


FIGURE 11

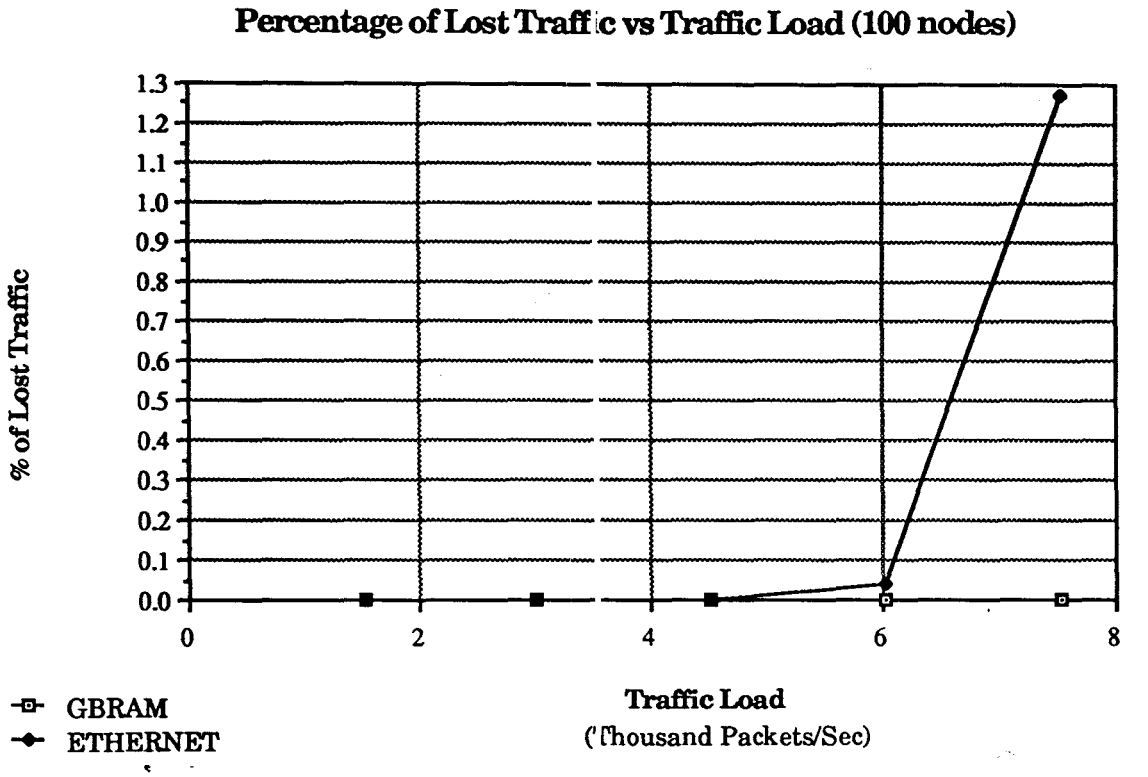


FIGURE 12

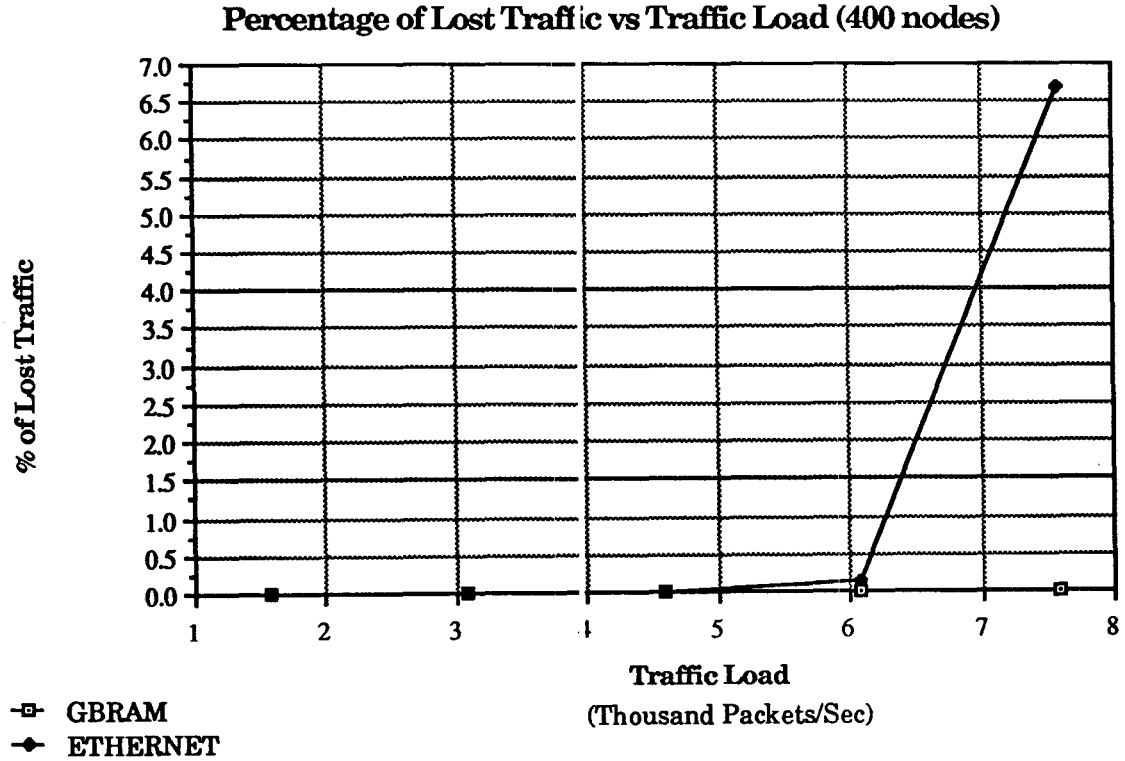


FIGURE 13

APPENDIX A: DESCRIPTION OF THE GBRAM PROTOCOL

Let **t** be a time epoch common to all the nodes; **t** corresponds to the beginning of a scheduling period. The beginning of a scheduling period corresponds to either the end of a transmission period (as perceived by the transmitting node) plus the propagation delay along the cable or to complete scheduling cycle after the beginning of the previous scheduling period (see Figure 5). Let us assume that node (i_1,j_1) is the node that transmitted last prior to time **t**. Then, an arbitrary node (i_2,j_2) is scheduled to transmit $T(i_2,j_2)$ units of time after **t**, where $T(i_2,j_2)$ is defined as follows:

$$T(i_2,j_2) = \sum_{j=j_1+}^{j_2-1} D(i_1,j;i_1,j+1)$$

$$\text{if } i_2 = i_1 \text{ and } j_1+1 \le j_2 \le N_{i_1}$$

$$T(i_2,j_2) = \sum_{j=j_1+1}^{N_{i_1}-1} D(i_1,j;i_1,j+1)+$$

$$\sum_{i=i_1}^{i_2-2}\{D(i,N_i;i+1,1)+\sum_{j=1}^{N_{i+1}-1} D(i+1,j;i+1,j+1)\}+$$

$$D(i_2-1,N_{i_2-1};i_2,1)+\sum_{j=1}^{j_2-1} D(i_2,j;i_2,j+1)$$

$$\text{if } i_1+1 \le i_2 \le M \text{ and } 1 \le j_2 \le N_{i_2}$$

and

$$T(i_2,j_2) = \sum_{j=j_1+1}^{N_{i_1}-1} D(i_1,j;i_1,j+1)+$$

$$\sum_{i=i_1}^{M-1}\{D(i,N_i;i+1,1)+\sum_{j=1}^{N_{i+1}-1} D(i+1,j;i+1,j+1)\}+$$

$$D(M,N_M;1,1)+$$

$$\sum_{i=1}^{i_2-1}\{D(i,N_i;i+1,1)+\sum_{j=1}^{N_i-1} D(i,j;i,j+1)\}+$$

$$\sum_{j=1}^{j_2-1} D(i_2,j;i_2,j+1)$$

$$\text{if } 1 \le i_2 \le i_1-1 \text{ and } 1 \le j_2 \le N_{i_2} \text{ or } i_2 = i_1 \text{ and } 1 \le j_2 \le j_1$$

APPENDIX B.1: DESCRIPTION OF THE ETHERNET SIMULATION CODE

Every node in the network has a queue where it stores the packets that arrive at its site and require access to the channel. These packets are served on a **first-come, first-serve basis**. The program utilizes two storage devices. One of them, called **stack**, stores the node packets that are at the top of the node queues. The packets in the **stack** follow the ETHERNET protocol to access the channel. All other node packets are stored in the second storage device called **buffer**.

First the program determines the propagation delays between any two nodes in the network. We consider two cases denoted as **minimum** and **maximum** delays. These cases correspond, respectively, to the best possible and the worst possible propagation delays between any two nodes in the network, as specified in the standards specification [1]. Then the program chooses the load of the channel and by utilizing a random number generator, determines the time of arrival and identity (i.e., the node they originate from) of the the generated packets.

At the beginning of the main program, we calculate CTN (the time that the state of the channel will change from idle to busy). The change of the channel state from idle to busy may originate either from a packet in the **stack** (i.e., packets that are already in the system) or from a new packet arrival. Following the determination of CTN, the program examines whether CTN is the beginning of a successful transmission or a collision event.

If a successful event occurs, the program calculates the delay of the successfully transmitted packet and stores this delay value either in the array **ARRAY (.)** (if the packet is a node packet) or in the array **MARRAY (.)** (if the packet is an MCC packet). Then, the program calculates the time **ABORT (1)** which corresponds to the time that the successful transmission ends. Furthermore, the program determines **IDLE** which is the time we can say with certainty that all nodes have seen the channel idle for at least Interframe Gap (IG) period of time after the end of the successful transmission.

If a collision event occurs, the colliding nodes are identified and their **stack** indices are saved into the array **STI (.)**. A retransmission (back-off) delay is assigned to them which is stored into the array **SRD (.)**. Then, the program calculates the time instances at which the colliding nodes stop their transmissions, and stores them in the array **ABORT (.)**. Furthermore, the program determines **IDLE**, which is the time we can say with certainty that all nodes have seen the channel idle for at least IG period of time after the end of all packet transmissions involved in the collision event. Finally, the time instances that the colliding nodes reduce their retransmission delays to zero (i.e., the time instances that they will reattempt to transmit) are calculated and stored in the array **RD (.)**.

From this point on, successful transmission and collision events are simulated simultaneously. Packets in the stack that will attempt transmission at a time prior to **IDLE** update their **RD (.)** values accordingly. New packets that arrive prior to **IDLE** are put in the **stack** with appropriate **RD (.)** values (i.e., the time instance that they will attempt to transmit). The packet among those stored in the **stack** that will transmit first (i.e., the packet with the smallest **RD (.)** value) is determined and its **stack** index is denoted by **IMIN**. At this point, the packets that have been successfully transmitted or over-collided (more than 16 successful attempts to access the channel) are discarded. After cleaning up the **stack**, we go back to the beginning of the main program where the new time **CTN** is found by comparing **RD (IMIN)** and the next new packet arrival after **IDLE**. Then, the main program is repeated and this process continues until all packets are successfully transmitted or over-collided. At the end of the program statistical results are calculated and printed out.

APPENDIX B.2 GLOSSARY OF TERMS FOR THE ETHERNET SIMULATION CODE

A(.) -> ARRAY CONTAINING THE TIME OF ARRIVALS OF NEW PACKETS

AD+RB -> PROPAGATION DELAY BETWEEN TWO NODES IN THE NETWORK LOCATED AT THE TWO ENDS OF THE CABLE

BO -> JAM SIGNAL (IN BITS) FOR MINIMUM DELAYS

BO+DF -> JAM SIGNAL FOR MAXIMUM DELAYS

C(.) -> ARRAY CONTAINING THE IDENTITIES OF NEW ARRIVALS

CO -> END TO END PROPAGATION DELAY ALONG THE CABLE

COLMC -> NUMBER OF TIMES MCC COLLIDES

CTN -> TIME THAT A NODE CHANGES THE STATE OF THE CHANNEL FROM IDLE TO BUSY

CU -> NUMBER OF PACKETS IN THE BUFFER

CULL -> MAXIMUM INDEX OF BUFFER

CULL1 -> AVERAGE INDEX OF BUFFER

D(I,J)+RB->THE PROPAGATION DELAY BETWEEN THE NODES I & J

DELAY(.)-> ARRAY CONTAINING THE DELAYS OF THE PACKETS TRANSMITTED BY THE NODES

DF -> CONSTANT EQUAL TO 16 BITS (MAXIMUM DELAYS) OR 0 BITS (MINIMUM DELAYS)

DG -> PROPAGATION DELAY BETWEEN TWO ADJACENT GROUPS OF NODES

DU+RB -> PROPAGATION DELAY BETWEEN TWO NODES IN THE SAME GROUP

GROUPS -> NUMBER OF GROUPS OF NODES

KL -> NUMBER OF PACKETS GENERATED

IDLE -> TIME AT WHICH ALL THE NODES SEE THE CHANNEL IDLE FOR AT LEAST IG UNITS OF TIME

IG -> INTERFRAME GAP(IN BITS)

IMIN -> STACK INDEX OF THE NODE WITH THE SMALLEST RD(.)

L -> TRAFFIC LOAD OF THE CHANNEL

L3 -> NUMBER OF TIMES THE STACK IS EMPTY

L4 -> NUMBER OF PACKETS TRANSMITTED BY THE NODES

L6 -> NUMBER OF COLLISIONS

LAST -> TIME ALL TRANSMISSIONS STOP

MCC -> IDENTITY OF THE MCC UNIT

MCP -> MCC PACKET LENGTH

MDELAY(.)->ARRAY CONTAINING THE DELAYS OF THE PACKETS TRANSMITTED BY THE MCC

MEAN1 -> AVERAGE DELAY OF A PACKET TRANSMITTED BY THE NODES

ML4 -> NUMBER OF PACKETS TRANSMITTED BY THE MCC

MMEAN1 -> AVERAGE DELAY OF A PACKET TRANSMITTED BY THE MCC

MP(.) -> ARRAY OF PERCENTAGES OF PACKETS TRANSMITTED BY THE MCC WITH DELAYS WITHIN CERTAIN PRESPECIFIED RANGES (e.g. (1,5),[5,10),[10-15) PACKET LENGTHS etc)

MPMI -> PERCENTAGE OF PACKETS TRANSMITTED BY THE MCC WITH DELAYS OF 1 PACKET LENGTH

MVARI -> VARIANCE OF THE DELAY OF A PACKET TRANSMITTED BY THE MCC

MX(.) -> ARRAY CONTAINING THE NUMBER OF PACKETS TRANSMITTED BY THE MCC WITH DELAYS WITHIN CERTAIN PRESPECIFIED RANGES (e.g. (1-5),[5-10),[10-15) PACKET LENGTHS etc)

MXDM -> NUMBER OF PACKETS TRANSMITTED BY THE MCC WITH DELAYS

OF 1 PACKET LENGTH

NC -> NUMBER OF COLLIDING PACKETS DURING A COLLISION EVENT

NL -> CURRENT INDEX OF THE ARRAYS CONTAINING THE IDENTITY AND TIME OF ARRIVAL OF NEW PACKETS

NS -> NUMBER OF PACKETS WAITING IN THE STACK FOR TRANSMISSION

NSCU -> AVERAGE INDEX OF BOTH STACK AND BUFFER

NSS -> MAXIMUM INDEX OF STACK

NSS1 -> AVERAGE INDEX OF STACK

OVERC -> NUMBER OF PACKETS DISCARDED BECAUSE THEY COLLIDE MORE THAN 16 TIMES

PE(.) -> ARRAY WITH PERCENTAGES OF PACKETS TRANSMITTED BY THE NODES WITH DELAYS WITHIN CERTAIN PRESPECIFIED RANGES (e.g. (1-5),[5-10),[10-15) PACKET LENGTHS etc)

PL -> PACKET LENGTH

PMI -> PERCENTAGE OF PACKETS TRANSMITTED BY THE NODES WITH DELAYS OF 1 PACKET LENGTH

RAND(0) -> UNIFORM RANDOM GENERATOR

RB -> CONSTANT EQUAL TO 3 BITS (MAXIMUM DELAYS) OR 0 BITS (MINIMUM DELAYS)

RC -> CONSTANT EQUAL TO 14 BIT (MAXIMUM DELAYS) OR 0 BITS (MINIMUM DELAYS)

RI -> CONSTANT EQUAL TO 16 BIT (MAXIMUM DELAYS) OR 0 BITS (MINIMUM DELAYS)

SA(.) ARRAY CONTAINING THE TIME OF ARRIVAL OF PACKETS IN THE STACK THAT WAIT FOR TRANSMISSION

SI(.) -> ARRAY CONTAINING THE IDENTITIES OF THE PACKETS IN THE STACK THAT WAIT FOR TRANSMISSION

SNC(.) -> ARRAY CONTAINING THE NUMBER OF COLLISIONS THAT A PACKET, WAITING IN THE STACK, HAS SUFFERED

SPI(.) -> ARRAY CONTAINING THE STACK INDEX OF THE PACKETS THAT MAY COLLIDE (POTENTIAL COLLIDERS)

STI(.) -> ARRAY CONTAINING THE STACK INDEX OF THE PACKETS THAT COLLIDE

TD(.) -> ARRAY CONTAINING THE NUMBER OF THE PACKETS TRANSMITTED BY THE MCC OR HE NODES WITH DELAYS WITHIN CERTAIN PRESPECIFIED RANGES (e.g. (1-5),[5-10),[10-15) PACKET LENGTHS etc)

TMEAN -> AVERAGE DELAY OF A PACKET TRANSMITTED BY THE NODES OR THE MCC

TP(.) -> ARRAY WITH PERCENTAGES OF PACKETS TRANSMITTED BY THE NODES OR MCC WITH DELAYS WITHIN CERTAIN PRESPECIFIED RANGES (e.g. (1-5),[5-10),[10-15) PACKET LENGTHS etc)

TPMI -> PERCENTAGE OF PACKETS TRANSMITTED BY THE NODES OR THE MCC WITH DELAYS OF 1 PACKET LENGTH

TXDM -> TOTAL NUMBER OF PACKETS TRANSMITTED BY THE NODES OR MCC WITH DELAYS OF 1 PACKET LENGTH

TVARI -> VARIANCE OF THE DELAY OF A PACKET TRANSMITTED BY THE NODES OR THE MCC

VARI -> VARIANCE OF THE DELAY OF A PACKET TRANSMITTED BY THE NODES

XD(.) -> ARRAY CONTAINING THE NUMBER OF PACKETS TRANSMITTED BY

THE NODES WITH DELAYS WITHIN CERTAIN PRESPECIFIED RANGES (e.g. (1-5),[5-10),[10-15) PACKET LENGTHS etc)

XDM -> NUMBER OF PACKETS TRANSMITTED BY THE NODES WITH DELAYS OF 1 PACKET LENGTH

APPENDIX B.3 ETHERNET SIMULATION CODE

```
      INTEGER I,J,J1,I1,NL,NS,IMIN,TEMP,PNC,NC,GENE
      INTEGER C(106001),SI(600),SNC(600),STI(500)
      INTEGER SFI(500),OVE(50),L3,L4,NSS,OVERC,MCC,MCP
      INTEGER K1,K2,K3,GROUPS,NODES,OV,NCL,L6,KL,PLL,PL1
      INTEGER SSI(40000),ML4,COLMC,CULL,L7,L9,CU
      INTEGER XI,XD(30),XDM,MX(30),MXDM,TD(30),TL4,TXDM
      REAL L
      DOUBLE PRECISION AD,BO,IG,PL,DU,DG,RB,RI,RC,CD,DF
      DOUBLE PRECISION D(402,402),RD(600),SA(600),SRD(400)
      DOUBLE PRECISION A(0:110001),DELAY(106000),ABORT(0:400)
      DOUBLE PRECISION MEAN1,CS,CTN,AVE1,VARI,NSS1,X,Y,IDLE
      DOUBLE PRECISION LAST,PE(30),CC,AA,MPL,XDMM,CULL1,NSCU
      DOUBLE PRECISION MDELAY(6000),MMEAN1,MVARI,MP(30),MPMI,PMI
      DOUBLE PRECISION MAVE1,SSA(40000),TP(30),TMEAN,TVARI,TPMI
C
      BO=32
      IG=96
      PL=1024
      A(0)=0
C
C DETERMINE IF THE DELAYS ARE MAXIMUM OR MINIMUM.
C
      DO 100  K1=1,2
         IF (K1.EQ.1) THEN
C           CONSIDERING MAXIMUM DELAYS
            DU=28.28
            RB=3
            RI=16
            RC=14
            CD=21.65
            DF=16
          ELSE
C           CONSIDERING MINIMUM DELAYS
            DU=6.68
            RB=0.00001
            RI=0
            RC=0
            CD=16.67
            DF=0
          ENDIF
          AD=CD+DU
          PLL=10
          PL1=50
C
C DETERMINE THE NUMBER OF NODES TO BE CONSIDERED.
C
      DO 200 K2=1,4
         IF (K2.EQ.1) NODES=100
```

```
          IF (K2.EQ.2) NODES=200
          IF (K2.EQ.3) NODES=300
          IF (K2.EQ.4) NODES=400
          MCC=NODES/2
C
C DETERMINE THE PROPAGATION DELAYS BETWEEN ANY PAIR OF NODES.
C
      GROUPS=NODES/8+1
      IF ((K2.EQ.2).OR.(K2.EQ.4)) GROUPS=GROUPS-1
      DG=CO/(GROUPS-1)
      DO 1 I=0,GROUPS-1
         DO 2 I1=1,8
            DO 3 J=0,GROUPS-1
               DO 4 J1=1,8
                  D(I*8+I1,J*8+J1)=IABS(I-J)*DG+DU
4              CONTINUE
3           CONTINUE
2        CONTINUE
1     CONTINUE
      DO 5 I=1,NODES
         D(I,I)=0
5     CONTINUE
C
C DETERMINE THE TRAFFIC LOAD, AS WELL AS, THE IDENTITY AND
C TIME OF ARRIVAL OF NEW PACKETS.
C
      DO 300  K3=1,5
         IF (K3.EQ.1) L=.15
         IF (K3.EQ.2) L=.3
         IF (K3.EQ.3) L=.45
         IF (K3.EQ.4) L=.6
         IF (K3.EQ.5) L=.75
         IF (K3.EQ.6) L=.9
         CC=10000000
         I=0
         MCP=0
         DO 6 I1=1,100000
            I=I+1
            A(I)=A(I-1)+(ALOG((1/(1-RAND(0)))**(1/L)))*PL
            IF (A(I).GE.CC) THEN
               AA=A(I)
               DO 113 J=1,NODES/50
                  A(I)=CC
                  C(I)=MCC
                  MCP=MCP+10
                  I=I+1
113            CONTINUE
               CC=CC+10000000
               A(I)=AA
            ENDIF
114         C(I)=RAND(0)*NODES+1
            IF (C(I).EQ.MCC) THEN
```

```
                IF ((K2.EQ.1).OR.(K2.EQ.3)) THEN
                   C(I)=NODES+1
                ELSE
                   GO TO 114
                ENDIF
             ENDIF
6          CONTINUE
           KL=I
           A(I+1)=900000000000000.0
C
        CULL=0
        CULL1=0
        NS1=0
        L7=0
        L9=0
        NSCU=0
        OV=0
        NL=1
        NS=0
        OVERI=0
        OVERC=0
        AVE1=0
        MAVE1=0
        ML4=0
        MEAN1=0
        VARI=0
        COLMC=0
        16=0
        L3=0
        NSS=0
        NSS1=0
        L4=0
        CU=0
        MPL=1024*PLL
C
C DETERMINE WHICH NODE INITIATES TRANSMISSION AND AT WHAT TIME (CTN).
C
400     IF ((NS.EQ.0).OR.(A(NL).LE.(RD(IMIN)))) THEN
           DO 8 I=1,NS
              IF (SI(I).EQ.C(NL)) THEN
                 CU=CU+1
                 SSA(CU)=A(NL)
                 SSI(CU)=C(NL)
                 NL=NL+1
                 GO TO 400
              ENDIF
8          CONTINUE
           NS=NS+1
           SA(NS)=A(NL)
           SI(NS)=C(NL)
           SNC(NS)=0
           RD(NS)=A(NL)
```

```
            PNC=1
            SPI(PNC)=NS
            CTN=A(NL)
            NL=NL+1
         ELSE
            PNC=1
            SPI(PNC)=IMIN
            CTN=RD(IMIN)
         ENDIF
         NSS1=NSS1+NS
C
C DETERMINE THE POTENTIAL COLLIDERS AMONG THE PACKETS IN THE STACK.
C
         DO 7 I=1,NS
            IF (((RD(I)).LE.(CTN+AO+RB)).AND.(I.NE.SPI(1))) THEN
               PNC=PNC+1
               SPI(PNC)=I
            ENDIF
7        CONTINUE
C
C DETERMINE THE POTENTIAL COLLIDERS AMONG THE NEW ARRIVALS.
C
10       IF (A(NL).LE.(CTN+AO+RB)) THEN
            DO 11 I=1,NS
               IF (SI(I).EQ.C(NL)) THEN
                  CU=CU+1
                  SSA(CU)=A(NL)
                  SSI(CU)=C(NL)
                  NL=NL+1
                  GO TO 10
               ENDIF
11          CONTINUE
            NS=NS+1
            SA(NS)=A(NL)
            SI(NS)=C(NL)
            SNC(NS)=0
            RD(NS)=A(NL)
            NL=NL+1
            PNC=PNC+1
            SPI(PNC)=NS
            GO TO 10
         ENDIF
C
C
C DETERMINE IF THERE IS A SUCCESSFUL TRANSMISSION OR COLLISION.
C
         DO 17 I=2,PNC
         IF ((RD(SPI(I))-RD(SPI(1))).LE.(RB+D(SI(SPI(I)),SI(SPI(1)))))
        $GO TO 22
17       CONTINUE
C
C A SUCCESSFUL TRANSMISSION TOOK PLACE.
```

```
C
C IF SUCCESSFUL TRANSMISSION(S) ORIGINATE FROM THE MCC UNIT,
C DETERMINE THE DELAYS OF THE MCC SUCCESSFULLY TRANSMITTED
C PACKET(S).
C
      IF (SI(SPI(1)).EQ.MCC) THEN
         DO 40 I=1,PLL
            ML4=ML4+1
            MDELAY(ML4)=RD(SPI(1))+I*PL-SA(SPI(1))
            MAVE1=MAVE1+MDELAY(ML4)
40       CONTINUE
         ABORT(1)=CTN+MPL
         IDLE=CTN+MPL+RI+IG+AO
         GO TO 147
      ENDIF
C
C IF THE SUCCESSFUL TRANSMISSION ORIGINATES FROM A NODE,
C DETERMINE THE DELAY OF THE SUCCESSFULLY TRANSMITTED PACKET.
C
      L4=L4+1
      DELAY(L4)=RD(SPI(1))+1024-SA(SPI(1))
      AVE1=AVE1+DELAY(L4)
      ABORT(1)=CTN+PL
      IDLE=CTN+PL+RI+IG+AO
147   OV=1
      OVE(OV)=SPI(1)
      GO TO 95
C
C A COLLISION TOOK PLACE.
C
C ORDER THE POTENTIAL COLLIDERS BASED ON THE TIME THEY ATTEMPT
C TO INITIATE THEIR TRANSMISSIONS.
C
22    DO 23 I=1,PNC-1
         X=RD(SPI(I))
         DO 24 J=I+1,PNC
            Y=RD(SPI(J))
            IF (X.GT.Y) THEN
               TEMP=SPI(I)
               SPI(I)=SPI(J)
               SPI(J)=TEMP
               X=Y
            ENDIF
24       CONTINUE
23    CONTINUE
      L6=L6+1
C
C DETERMINE THE PACKETS THAT COLLIDE AND STORE THEIR STACK INDEX
C INTO THE ARRAY STI(.) AND THEIR RETRANSMISSION DELAYS INTO
C THE ARRAY SRD(.).
C
      DO 25 J=1,PNC
```

```
            C5=RD(SPI(1))+D(SI(SPI(1)),SI(SPI(J)))
            DO 26 I=2,NC
               IF (RD(STI(I))+D(SI(STI(I)),SI(SPI(J))).LE.C5) THEN
                  C5=RD(STI(I))+D(SI(STI(I)),SI(SPI(J)))
               ENDIF
26          CONTINUE
            IF (RD(SPI(J)).LE.C5+RB) THEN
C              THE PACKET WITH STACK INDEX SPI(J) HAS COLLIDED.
               SNC(SPI(J))=SNC(SPI(J))+1
               IF (SI(SPI(J)).EQ.MCC) COLMC=COLMC+1
               IF (SNC(SPI(J)).GT.16) THEN
                  OVERC=OVERC+1
                  OV=OV+1
                  OVE(OV)=SPI(J)
               ENDIF
               IF (SNC(SPI(J)).LE.10) NCL=SNC(SPI(J))
               IF (SNC(SPI(J)).GT.10) NCL=10
               GENE=RAND(0)*2**NCL
               NC=NC+1
               SRD(NC)=GENE*512
               STI(NC)=SPI(J)
            ENDIF
25       CONTINUE

C
C DETERMINE THE TIME THAT THE COLLIDERS STOP TRANSMITTING, AND STORE
C THAT TIME INTO THE ARRAY ABORT(.). ALSO DETERMINE IDLE THAT IS THE
C TIME AT WHICH WE CAN SAY WITH CERTAINTY THAT ALL THE NODES HAVE
C SEEN THE CHANNEL IDLE FOR AT LEAST IG UNITS OF TIME AFTER THE END
C OF ALL PACKET TRANSMISSIONS INVOLVED IN THE COLLISION.
C
         ABORT(0)=0
         DO 29 I=1,NC
            IF (I.NE.1) THEN
               C5=RD(STI(1))+D(SI(STI(1)),SI(STI(I)))
            ELSE
               C5=RD(STI(2))+D(SI(STI(1)),SI(STI(2)))
            ENDIF
            DO 30 J=2,NC
               IF (I.NE.J) THEN
                  IF ((RD(STI(J))+D(SI(STI(J)),SI(STI(I)))).LT.C5) THEN
                     C5=RD(STI(J))+D(SI(STI(I)),SI(STI(J)))
                  ENDIF
               ENDIF
30          CONTINUE
            ABORT(I)=C5+RC+DF+BO
C           A USER MUST TRANSMIT AT LEAST 56 BITS
            IF ((ABORT(I)-RD(STI(I))).LT.56) ABORT(I)=56+RD(STI(I))
            IF (ABORT(I).GT.IDLE) IDLE=ABORT(I)
29       CONTINUE
         IDLE=IDLE+RI+IG+AO
C
```

```
C DETERMINE THE TIME THAT THE COLLIDERS WILL REDUCE THEIR
C RETRANSMISSION DELAY TO ZERO AND STORE IT INTO RD(STI(.)).
C
      DO 504 I=1,NC
         RD(STI(I))=ABORT(I)+GRD(I)
504   CONTINUE
C
C CONSIDER SUCCESSFUL AND COLLISION EVENTS IN UNISON
C
C IF ANY OF THE PACKETS IN THE STACK HAS REDUCED ITS RETRANSMISSION
C DELAY TO ZERO AT THE TIME INSTANCE THAT IT SEES THE CHANNEL IDLE
C FOR IG TIME, DETERMINE THE TIME THAT IT WILL ATTEMPT TO INITIATE
C TRANSMISSION AND STORE THAT VALUE INTO RD(.)
C
95    DO 31 I=1,NS
        IF (RD(I).LE.IDLE) THEN
           C5=ABORT(1)+D(SI(I),SI(SPI(1)))
           DO 32 J=2,NC
              IF ((ABORT(J)+D(SI(I),SI(STI(J)))).GT.C5) THEN
                 C5=ABORT(J)+D(SI(I),SI(STI(J)))
              ENDIF
32         CONTINUE
           IF (RD(I).LE.(C5+RI+IG)) THEN
              RD(I)=C5+RI+IG
              IF ((SI(I).EQ.SI(SPI(1))).AND.(NC.EQ.0)) RD(I)=C5+IG
           ENDIF
        ENDIF
31    CONTINUE
C
C PICK UP ALL THE PACKETS THAT ARRIVED BEFORE THE TIME INSTANCE
C 'IDLE' AND PUT THEM IN THE STACK.
C
152   IF (A(NL).LE.IDLE) THEN
         C5=ABORT(1)+D(C(NL),SI(SPI(1)))
         DO 34 I=2,NC
            IF ((ABORT(I)+D(C(NL),SI(STI(I)))).GT.C5) THEN
               C5=ABORT(I)+D(C(NL),SI(STI(I)))
            ENDIF
34       CONTINUE
         DO 38 I=1,NS
            IF (SI(I).EQ.C(NL)) THEN
               CU=CU+1
               SSA(CU)=A(NL)
               SSI(CU)=C(NL)
               NL=NL+1
               GO TO 152
            ENDIF
38       CONTINUE
         NS=NS+1
         SA(NS)=A(NL)
         SI(NS)=C(NL)
         SNC(NS)=0
```

```
            RD(NS)=C5+RI+IG
            IF ((C(NL).EQ.SI(SPI(1))).AND.(NC.EQ.0)) RD(NS)=C5+IG
            IF (A(NL).GT.(C5+RI+IG)) RD(NS)=A(NL)
            NL=NL+1
            GO TO 152
         ENDIF
C
C IF A SUCCESSFUL TRANSMISSION OCCURED GET RID OF THE SUCCESSFULLY
C TRANSMITTED PACKET. IF A COLLISION EVENT OCCURED GET RID OF THE
C OVERCOLLIDED PACKETS. IF ANY OF THE NODES THAT OVERCOLLIDES OR
C OR TRANSMITS A PACKET HAS ANOTHER PACKET WAITING IN THE BUFFER
C PUT IT IN THE STAC.
C
         DO 43 I=1,OV
            DO 44 J=1,CU
               IF (SI(OVE(I)).EQ.SSI(J)) THEN
                    NS=NS+1
                    SA(NS)=SSA(J)
                    SI(NS)=SSI(J)
                    SNC(NS)=0
                    C5=ABORT(I)+D(SSI(J),SI(SPI(1)))
                    DO 465 I1=2,NC
                       IF ((ABORT(I)+D(SSI(J),SI(STI(I)))).GT.C5) THEN
                          C5=ABORT(I)+D(SSI(J),SI(STI(I)))
                       ENDIF
465                 CONTINUE
                    RD(NS)=C5+RI+IG
                    IF ((SSI(J).EQ.SI(SPI(1))).AND.(NC.EQ.0)) RD(NS)=C5+IG
                    IF (SSA(J).GT.(C5+RI+IG)) RD(NS)=SSA(J)
                    DO 491 I1=J,CU-1
                        SSA(I1)=SSA(I1+1)
                        SSI(I1)=SSI(I1+1)
491                 CONTINUE
                    CU=CU-1
                    GO TO 46
               ENDIF
44          CONTINUE
46          DO 45 J1=OVE(I),NS-1
               SA(J1)=SA(J1+1)
               SI(J1)=SI(J1+1)
               SNC(J1)=SNC(J1+1)
               RD(J1)=RD(J1+1)
45          CONTINUE
            NS=NS-1
43       CONTINUE
C
C DETERMINE THE INDEX OF THE PACKET IN THE THE STACK WITH THE
C SMALLER RD(.)
C
          C5=RD(1)
          IMIN=1
          DO 35 I=2,NS
```

```
            IF (RD(I).LT.C5) THEN
              IMIN=I
              C5=RD(I)
            ENDIF
35      CONTINUE
C
C FIND THE MAXIMUM INDEX OF THE STACK AND OTHER DATA AND THEN
C GO TO THE BEGINNING OF THE MAIN PROGRAM (STATEMENT 400), AND
C REPEAT THE WHOLE PROCEDURE ONCE MORE, UNTIL ALL THE PACKETS
C ARE SUCCESSFULLY TRANSMITTED.
C
        IF (NSS.LT.NS) NSS=NS
        IF (CU.GT.CULL) CULL=CU
        CULL1=CULL1+CU
        IF (NS.EQ.0) L3=L3+1
        IF (CU.EQ.0) L7=L7+1
        L9=L9+1
        OV=0
        NC=0
        IF ((NS.EQ.0).AND.(NL.GT.KL)) GO TO 150
        GO TO 400
C
C CALCULATE STATISTICAL RESULTS AT THE END OF THE MAIN PROGRAM.
C
150     TL4=L4+ML4
        NSS1=NSS1/L9
        CULL1=CULL1/L9
        NSCU=NSS1+CULL1
        TMEAN=(AVE1+MAVE1)/TL4
        IF (L5.NE.0)  MEAN2=AVERA/(1024*L5)
        DO 262 I=1,30
           XD(I)=0
           MX(I)=0
           TD(I)=0
262     CONTINUE
C
C STATISTICAL RESULTS FOR THE NODE PACKETS.
C
        MEAN1=AVE1/L4
        VARI=0
        XDN=0
        DO 176 I=1,L4
           VARI=VARI+((DELAY(I)-MEAN1)**2)/(1024*1024)
           XDMM=DELAY(I)/1024
           IF (XDMM.EQ.1) XDM=XDM+1
           XI=DELAY(I)/5120+1
           IF (XI.LT.23) THEN
              XD(XI)=XD(XI)+1
           ELSE
              XI=DELAY(I)/20480.0+17.5
              IF (XI.LT.29) THEN
                 XD(XI)=XD(XI)+1
```

```
               ELSE
                  XD(29)=XD(29)+1
               ENDIF
            ENDIF
176      CONTINUE
         DO 191 I=1,29
            PE(I)=(100.0*XD(I)/L4)
191      CONTINUE
         PMI=(100.0*XDM/L4)
         VARI=VARI/(L4-1)
         MEAN1=MEAN1/1024
C
C STATISTICAL RESULTS FOR THE MCC PACKETS.
C
         MMEAN1=MAVE1/ML4
         MVARI=0
         MXDM=0
         DO 323 I=1,ML4
            MVARI=MVARI+((MDELAY(I)-MMEAN1)**2)/(1024*1024)
            XDMM=MDELAY(I)/1024
            IF (XDMM.EQ.1) MXDM=MXDM+1
            XI=MDELAY(I)/5120+1
            IF (XI.LT.23) THEN
               MX(XI)=MX(XI)+1
            ELSE
               XI=MDELAY(I)/20480.0+17.5
               IF (XI.LT.29) THEN
                  MX(XI)=MX(XI)+1
               ELSE
                  MX(29)=MX(29)+1
               ENDIF
            ENDIF
323      CONTINUE
         DO 163 I=1,29
            MP(I)=(100.0*MX(I)/ML4)
163      CONTINUE
         MPMI=(100.0*MXDM/ML4)
          MVARI=MVARI/(ML4-1)
          MMEAN1=MMEAN1/1024.0
C
C STATISTICAL RESULTS FOR THE MCC AND NODE PACKETS.
C
         TVARI=0
         TXDM=0
         J1=L4+1
         I1=0
         DO 155 I=J1,TL4
            I1=I1+1
            DELAY(I)=MDELAY(I1)
155      CONTINUE
         DO 121 I=1,TL4
            TVARI=TVARI+((DELAY(I)-TMEAN)**2)/(PL**2)
```

```
          XDMM=DELAY(I)/1024
          IF (XDMM.EQ.1) TXDM=TXDM+1
          XI=DELAY(I)/5120+1
          IF (XI.LT.23) THEN
             TD(XI)=TD(XI)+1
          ELSE
             XI=DELAY(I)/20480.0+17.5
             IF (XI.LT.29) THEN
                TD(XI)=TD(XI)+1
             ELSE
                TD(29)=TD(29)+1
             ENDIF
          ENDIF
121     CONTINUE
        DO 126 I=1,29
           TP(I)=(100.0*TD(I)/TL4)
126     CONTINUE
        TPMI=(100.0*TXDM/TL4)
        TVARI=TVARI/(TL4-1)
        TMEAN=TMEAN/1024.0
        LAST=RD(1)+PL
        IF (SI(1).EQ.MCC) LAST=RD(1)+MCP
        LAST=LAST/1024
C
C OUTPUT THE RESULTS AT THE END OF THE PROGRAM.
C
         IF (K1.EQ.1)      PRINT *,'                    MAXIMUM DELAYS'
         IF (K1.EQ.2)      PRINT *,'                    MINIMUM DELAYS'
       PRINT *,'|-------------------------------------------------------|'
       PRINT *,'|     1       1-5  5-10  10-15  15-20  20-25  25-30 30-35'
366    FORMAT (2X,I6,7I7)
       WRITE(6,366) XDM,XD(1)-XDM,XD(2),XD(3),XD(4),XD(5),XD(6),XD(7)
156    FORMAT (2X,I6,7I7)
       WRITE(6,156)MXDM,MX(1)-MXDM,MX(2),MX(3),MX(4),MX(5),MX(6),MX(7)
401    FORMAT (2X,I6,7I7)
       WRITE(6,401) TXDM,TD(1)-TXDM,TD(2),TD(3),TD(4),TD(5),TD(6),TD(7)
389    FORMAT (1X,8F7.3)
       WRITE (6,389) PMI,PE(1)-PMI,PE(2),PE(3),PE(4),PE(5),PE(6),PE(7)
166    FORMAT (1X,8F7.3)
       WRITE (6,166) MPMI,MP(1)-MPMI,MP(2),MP(3),MP(4),MP(5),MP(6),MP(7)
402    FORMAT (1X,8F7.3)
       WRITE (6,402) TPMI,TP(1)-TPMI,TP(2),TP(3),TP(4),TP(5),TP(6),TP(7)
       PRINT *,'     '
       PRINT *,'|  35-40  40-45  45-50  50-55  55-60  60-65 65-70 70-75'
377    FORMAT (1X,8I7)
       WRITE(6,377) XD(8),XD(9),XD(10),XD(11),XD(12),XD(13),XD(14),XD(15)
167    FORMAT (1X,8I7)
       WRITE(6,167) MX(8),MX(9),MX(10),MX(11),MX(12),MX(13),MX(14),MX(15)
403    FORMAT (1X,8I7)
       WRITE(6,403)TD(8),TD(9),TD(10),TD(11),TD(12),TD(13),TD(14),TD(15)
253    FORMAT (1X,8F7.3)
       WRITE(6,253) PE(8),PE(9),PE(10),PE(11),PE(12),PE(13),PE(14),PE(15)
```

```
254   FORMAT (1X,8F7.3)
      WRITE(6,254) MP(8),MP(9),MP(10),MP(11),MP(12),MP(13),MP(14),MP(15)
404   FORMAT (1X,8F7.3)
      WRITE(6,404) TP(8),TP(9),TP(10),TP(11),TP(12),TP(13),TP(14),TP(15)
      PRINT *,'    '
      PRINT *,'| 75-80  80-85  85-90  90-95  95-100 100-105 105-110'
388   FORMAT (1X,5I7,1X,2I8,I5)
      WRITE(6,388) XD(16),XD(17),XD(18),XD(19),XD(20),XD(21),XD(22)
157   FORMAT (1X,5I7,1X,2I8,I5)
      WRITE(6,157) MX(16),MX(17),MX(18),MX(19),MX(20),MX(21),MX(22)
405   FORMAT (1X,5I7,1X,2I8,I5)
      WRITE(6,405) TD(16),TD(17),TD(18),TD(19),TD(20),TD(21),TD(22)
352   FORMAT(1X,5F7.3,2X,3F7.3)
      WRITE(6,352) PE(16),PE(17),PE(18),PE(19),PE(20),PE(21),PE(22)
231   FORMAT(1X,5F7.3,2X,3F7.3)
      WRITE(6,231) MP(16),MP(17),MP(18),MP(19),MP(20),MP(21),MP(22)
406   FORMAT (1X,5F7.3,2X,3F7.3)
      WRITE(6,406) TP(16),TP(17),TP(18),TP(19),TP(20),TP(21),TP(22)
      PRINT *,'     '
      PRINT *,'|110-130 130-150 150-170 170-190 190-210 210-230   OVER'
257   FORMAT (1X,7I8)
      WRITE(6,257) XD(23),XD(24),XD(25),XD(26),XD(27),XD(28),XD(29)
232   FORMAT (1X,7I8)
      WRITE(6,232) MX(23),MX(24),MX(25),MX(26),MX(27),MX(28),MX(29)
407   FORMAT (1X,7I8)
      WRITE(6,407) TD(23),TD(24),TD(25),TD(26),TD(27),TD(28),TD(29)
358   FORMAT (1X,7F8.3)
      WRITE(6,358) PE(23),PE(24),PE(25),PE(26),PE(27),PE(28),PE(29)
233   FORMAT (1X,7F8.3)
      WRITE(6,233) MP(23),MP(24),MP(25),MP(26),MP(27),MP(28),MP(29)
408   FORMAT (1X,7F8.3)
      WRITE(6,408) TP(23),TP(24),TP(25),TP(26),TP(27),TP(28),TP(29)
      PRINT *,' '
      PRINT *,'|-------------------------------------------------------|'
      PRINT *,'| NUMBER OF NODES                                =',NODES
      PRINT *,'| TRAFFIC LOAD                                   =',L
      PRINT *,'| NUMBER OF PACKETS MCC GENERATES                =',MCP
      PRINT *,'| NUMBER OF TIMES MCC COLLIDES                   =',COLMC
      PRINT *,'| SUCCESSFULLY TRANSM. PACKETS-MCC               =',ML4
      PRINT *,'| AVERAGE TRANSMISSION DELAY-MCC                 =',MMEAN1
      PRINT *,'| VARIANCE TRANSMISSION DELAY-MCC                =',MVARI
      PRINT *,'| OVERCOLLIDED PACKETS                           =',OVERC
      PRINT *,'| SUCCESSFULLY TRANSMITTED PACKETS-NODES         =',L4
      PRINT *,'| AVERAGE TRANSMISSION DELAY-NODES               =',MEAN1
      PRINT *,'| VARIANCE TRANSMISSION DELAY-NODES              =',VARI
      PRINT *,'| AVERAGE TRANSMISSION DELAY-TOTAL               =',TMEAN
      PRINT *,'| VARIANCE TRANSMISSION DELAY-TOTAL              =',TVARI
      PRINT *,'| MAXIMUM INDEX OF STACK                         =',NSS
      PRINT *,'| AVERAGE INDEX OF STACK                         =',NSS1
      PRINT *,'| MAXIMUM INDEX OF BUFFER                        =',CULL
      PRINT *,'| AVERAGE INDEX OF BUFFER                        =',CULL1
      PRINT *,'| AVERAGE INDEX OF BOTH STACK AND BUFFER         =',NSCU
```

```
      PRINT *,'| TIME ALL TRANSMISSIONS STOP              =',LAST
      PRINT *,'| NUMBER OF COLLISIONS                    =',L6
      PRINT *,'|-------------------------------------------------|'
300   CONTINUE
200   CONTINUE
100   CONTINUE
      STOP
      END
```

APPENDIX C.1: DESCRIPTION OF THE GBRAM SIMULATION CODE

First, the program determines the propagation delays between any two nodes in the network. We consider two cases denoted as **minimum** and **maximum** delays. These cases correspond to, respectively, the best possible and the worst possible propagation delays between any two nodes in the network, as specified in the standards specification [1]. Then, the program chooses the load of the channel and, by utilizing a random number generator, determines the time of arrival and identity (i.e., the node they originate from) of the generated packets. The nodes in the network have indices 1, 2, 3, ...

The main program calculates the packet that will change the state of the channel from idle to busy. We call this packet **the packet**. This is done by considering all the packets that arrived prior to a complete scheduling cycle after the current time (**TIME**). From these packets, the program considers the packets that arrive prior to their scheduling instance and picks the packet with the smallest scheduling instance. This is **the packet**. If no packet initiates transmission in the first complete scheduling cycle after **TIME**, the program considers all packets that arrived prior to two complete scheduling cycles after **TIME**, and so on.

Through the above process, the identity of **the packet** (**LAST**) and the time (**TIME 1**) that **the packet** will initiate transmission are determined. Furthermore, the delay of **the packet** is saved into the array **DELAY (.)** if it is a node packet and into the array **MDELAY (.)** if it is an MCC packet. Next, the new value of **TIME** is computed by adding to **TIME 1** the length of **the packet** and the end to end propagation delay along the cable and subtracting from **TIME 1** the propagation delay between nodes **LAST** and **LAST+1**. This way, the scheduling instance of any node in the first scheduling cycle after **TIME** is found by adding to **TIME** the difference between the scheduling instances of the node and the node that transmitted last (**LAST**).

Finally, the packet is eliminated from the system and the main program is repeated until all generated packets are successfully transmitted. Then, statistical results are calculated and printed out.

APPENDIX C.2 GLOSSARY OF TERMS FOR THE GBRAM SIMULATION CODE

A(.) -> ARRAY CONTAINING THE TIME OF ARRIVALS OF NEW PACKETS
C(.) -> ARRAY CONTAINING THE IDENTITIES OF NEW ARRIVALS
CO -> END TO END PROPAGATION DELAY ALONG THE CABLE
D(I,J) -> THE TIME DIFFERENCE BETWEEN THE SCHEDULING INSTANCES OF NODES I & J
DELAY(.)-> ARRAY CONTAINING THE DELAYS OF THE PACKETS TRANSMITTED BY THE NODES
DG -> PROPAGATION DELAY BETWEEN TWO ADJACENT GROUPS OF NODES
DU -> PROPAGATION DELAY BETWEEN TWO NODES IN THE SAME GROUP
GROUPS -> NUMBER OF GROUPS OF NODES
IDL -> TIME THE CHANNEL WAS IDLE
INDEX -> INDEX OF THE PACKET WITH PRIORITY FROM THE NEW ARRIVALS
KL -> NUMBER OF PACKETS GENERATED
L -> TRAFFIC LOAD OF THE CHANNEL
L4 -> NUMBER OF PACKETS TRANSMITTED BY THE NODES
LAST -> IDENTITY OF THE NODE THAT INITIATED THE LAST TRANSMISSION
MCC -> IDENTITY OF THE MCC UNIT
MDELAY(.)->ARRAY CONTAINING THE DELAYS OF THE PACKETS TRANSMITTED BY THE MCC
ML4 -> NUMBER OF PACKETS TRANSMITTED BY THE MCC
MMEAN1 -> AVERAGE DELAY OF A PACKET TRANSMITTED BY THE MCC
MP(.) -> ARRAY WITH PERCENTAGES OF PACKETS TRANSMITTED BY THE MCC WITH DELAYS WITHIN CERTAIN PRESPECIFIED RANCES (e.g. (2,5),[5-10),[10,15) PACKET LENGTHS etc)
MPMI -> PERCENTAGE OF PACKETS TRANSMITTED BY THE MCC WITH DELAYS BETWEEN 1 & 2 PACKET LENGTHS
MVARI -> VARIANCE OF THE DELAY OF A PACKET TRANSMITTED BY THE MCC
MX(.) -> ARRAY CONTAINING THE NUMBER OF PACKETS TRANSMITTED BY THE MCC WITH DELAYS WITHIN CERTAIN PRESPECIFIED RANGES (e.g. (2-5),[5-10),[10-15) PACKET LENGTHS etc)
MXDM -> NUMBER OF PACKETS TRANSMITTED BY THE MCC WITH DELAYS BETWEEN 1 & 2 PACKET LENGTHS
NL ->CURRENT INDEX OF THE ARRAYS CONTAINING THE IDENTITY AND TIME OF ARRIVAL OF NEW PACKETS
PE(.) -> ARRAY WITH PERCENTAGES OF PACKETS TRANSMITTED BY THE NODES WITH DELAYS WITHIN CERTAIN PRESPECIFIED RANGES (e.g. (2-5),[5-10),[10-15) PACKET LENGTHS etc)
PL -> PACKET LENGTH
PMI -> PERCENTAGE OF PACKETS TRANSMITTED BY THE NODES WITH DELAYS BETWEEN 1 & 2 PACKET LENGTHS
RAND(0) -> UNIFORM RANDOM GENERATOR
TD(.) -> ARRAY CONTAINING THE NUMBER OF PACKETS TRANSMITTED BY THE MCC OR THE NODES WITH DELAYS WITHIN CERTAIN PRESPECIFIED RANGES (e.g. (2-5),[5-10),[10-15) PACKET LENGTHS etc)
TIME -> TIME THAT THE NODE 'LAST' TERMINATED ITS TRANSMISSION

PLUS THE END TO END PROPAGATION DELAY ALONG THE CABLE MINUS THE PROPAGATION DELAY BETWEEN THE NODE 'LAST' AND 'LAST'+1

TMEAN -> AVERAGE DELAY OF A PACKET TRANSMITTED BY THE NODES OR THE MCC

TP(.) -> ARRAY WITH PERCENTAGES OF PACKETS TRANSMITTED BY THE NODES OR MCC WITH DELAYS WITHIN CERTAIN PRESPECIFIED RANGES (e.g. (2-5),[5-10),[10-15) PACKET LENGTHS etc)

TPMI -> PERCENTAGE OF PACKETS TRANSMITTED BY THE NODES OR MCC WITH DELAYS BETWEEN 1 & 2 PACKET LENGTHS

TXDM -> TOTAL NUMBER OF PACKETS TRANSMITTED WITH DELAYS BETWEEN 1 & 2 PACKET LENGTHS

TVARI -> VARIANCE OF THE DELAY OF A PACKET TRANSMITTED BY THE NODES OR THE MCC

VARI -> VARIANCE OF THE DELAY OF A PACKET TRANSMITTED BY THE NODES

XD(.) -> ARRAY CONTAINING THE NUMBER OF PACKETS TRANSMITTED BY THE NODES WITH DELAYS WITHIN CERTAIN PRESPECIFIED RANGES (e.g. (2-5),[5-10),[10-15) PACKET LENGTHS etc)

XDM -> NUMBER OF PACKETS TRANSMITTED BY THE NODES WITH DELAYS BETWEEN 1 & 2 PACKET LENGTHS

APPENDIX C.3: GBRAM SIMULATION CODE

```
      INTEGER K1,K2,GROUPS,NODES,I,J,K3,KL,L4,NL1,TL4
      INTEGER C(101001),NL,LAST,I1,J1,TXDM,TD(30)
      INTEGER XD(30),XI,XDMM,XDM,INDEX
      INTEGER PLL,PL1,MCC,MCP,MXDM,ML4,MX(30)
      REAL L
      DOUBLE PRECISION DU,CO,DG,PL,TIME,MEAN,DDD
      DOUBLE PRECISION A(0:101003),D(401,401),DELAY(110000)
      DOUBLE PRECISION VARI,PE(30),PMI,TIME1,SSS,IDL
      DOUBLE PRECISION CC,AA,MDELAY(20000),MMEAN1,MVARI,MP(30)
      DOUBLE PRECISION MPMI,TMEAN,TVARI,TP(30)
      A(0)=0
      PL=1024
      DO 100 K1=1,2
      IF (K1.EQ.1) THEN
C        CONSIDERING MAXIMUM DELAYS
         DU=31.28
         CO=21.65
      ELSE
C        CONSIDERING MINIMUM DELAYS
         DU=6.68
         CO=16.67
      ENDIF
      PLL=10
      PL1=50
C
C DETERMINE THE NUMBER OF NODES TO BE CONSIDERED.
C
      DO 110 K2=1,4
         NODES=100*K2
         MCC=NODES/2
C
C DETERMINE THE TIME DIFFERENCE BETWEEN THE SCHEDULING
C INSTANCES OF THE NODES GIVEN BY J*8+I1 AND J*8+J1.
C
      GROUPS=NODES/8+1
      IF ((K2.EQ.2).OR.(K2.EQ.4)) GROUPS=GROUPS-1
      if ((K2.EQ.1).OR.(K2.EQ.3)) NODES=NODES+1
      DG=CO/(GROUPS-1)
      DDD=2*CO+DU*NODES
      DO 1 I=0,GROUPS-1
       DO 2 I1=1,8
        DO 3 J=0,GROUPS-1
         DO 4 J1=1,8
          IF (((J*8+J1)-(I*8+I1)).GT.0) THEN
           D(I*8+I1,J*8+J1)=IABS(I-J)*DG+DU*((J*8+J1)-(I*8+I1))
          ELSE
```

```
            D(I*8+I1,J*8+J1)=DDD-(IABS(I-J)*DG+IABS((J*8+J1)-
     $       (I*8+I1))*DU)
           ENDIF
4         CONTINUE
3        CONTINUE
2       CONTINUE
1      CONTINUE

C
C DETERMINE THE TRAFFIC LOAD, AS WELL AS, THE
C IDENTITY AND TIME OF ARRIVAL OF THE NEW PACKETS.
C
       DO 120 k3=1,6
          L=K3*.15
          NODES=100*K2
          CC=10000000
          I=0
          MCP=0
          DO 6 I1=1,100000
             I=I+1
             A(I)=A(I-1)+(ALOG((1/(1-RAND(0)))**(1/L)))*PL
             IF (A(I).GE.CC) THEN
                AA=A(I)
                DO 113 J=1,NODES/50
                   A(I)=CC
                   C(I)=MCC
                   MCP=MCP+10
                   I=I+1
113             CONTINUE
                CC=CC+10000000
                A(I)=AA
             ENDIF
114          C(I)=RAND(0)*NODES+1
             IF (C(I).EQ.MCC) THEN
                IF ((K2.EQ.1).OR.(K2.EQ.3)) THEN
                   C(I)=NODES+1
                ELSE
                   GO TO 114
                ENDIF
             ENDIF
6         CONTINUE
          KL=I
          IF ((K2.EQ.1).OR.(K2.EQ.3)) NODES=NODES+1
          A(I+1)=99000000000.0
          A(I+2)=99900000000.0
C
C MAIN PROGRAM.
C
       L4=0
       NL=1
       TIME=0
       LAST=1
```

```
        MEAN=0
        ML4=0
        MMEAN1=0
        IDL=0
C
C FIND THE NODE WITH PRIORITY(i.e.,THE NODE WHICH TRANSMITS THE
C PACKET) AND THE TIME IT WILL INITIATE TRANSMISSION.
C
15      NL1=NL
        I1=(A(NL)-TIME)/DDD
        IF (I1.LT.0) I1=0
        INDEX=0
        SSS=TIME+(I1+1)*DDD
16      TIME1=SSS+DU
10      IF (A(NL1).LE.SSS) THEN
           IF (A(NL1).LE.TIME+I1*DDD+D(LAST,C(NL1))) THEN
              IF (TIME1.GT.I1*DDD+D(LAST,C(NL1))) THEN
                 INDEX=NL1
                 TIME1=I1*DDD+D(LAST,C(NL1))
              ENDIF
           ENDIF
           NL1=NL1+1
           GO TO 10
        ENDIF
        IF (INDEX.EQ.0) THEN
           NL1=NL
           I1=I1+1
           SSS=SSS+DDD
           GO TO 16
        ENDIF
        TIME1=TIME1+TIME
        LAST=C(INDEX)
C
C CALCULATE THE DELAY OF THE SUCCESSFULLY TRANSMITTED PACKET
C (i.e., THE PACKET).
C
        IF (C(INDEX).EQ.MCC) THEN
           DO 40 I=1,PLL
           ML4=ML4+1
             MDELAY(ML4)=TIME1+I*PL-A(INDEX)
             MMEAN1=MMEAN1+MDELAY(ML4)
40         CONTINUE
           TIME=TIME1+10*PL+CO-D(LAST,LAST+1)
           GO TO 32
        ENDIF
        L4=L4+1
        DELAY(L4)=TIME1+PL-A(INDEX)
        MEAN=MEAN+DELAY(L4)
        TIME=TIME1+PL+CO-D(LAST,LAST+1)
        IF (LAST.EQ.NODES) TIME=TIME1+PL+CO-D(LAST,1)
C
C GET RID OF THE SUCCESSFULLY TRANSMITTED PACKET(i.e., THE PACKET).
```

```
C
32      IF (INDEX.NE.NL) THEN
             A(INDEX)=A(INDEX-1)
             C(INDEX)=C(INDEX-1)
             INDEX=INDEX-1
             GO TO 32
        ENDIF
        NL=NL+1
        IF (NL.GT.KL) GO TO 150
        GO TO 15
C
C CALCULATE STATISTICAL RESULTS AT THE END OF THE MAIN PROGRAM.
C
150     TL4=L4+ML4
        TMEAN=(MEAN+MMEAN1)/TL4
        DO 80 I=1,30
           XD(I)=0
           MX(I)=0
           TD(I)=0
80      CONTINUE
C
C STATISTICAL RESULTS FOR THE NODE PACKETS.
C
        MEAN=MEAN/L4
        VARI=0
        XDM=0
        DO 81 I=1,L4
           VARI=VARI+((DELAY(I)-MEAN)**2)/(PL**2)
           XDMM=DELAY(I)/1024
           IF (XDMM.EQ.1) XDM=XDM+1
           XI=DELAY(I)/5120+1
           IF (XI.LT.23) THEN
              XD(XI)=XD(XI)+1
           ELSE
              XI=DELAY(I)/20480.0+17.5
              IF (XI.LT.29) THEN
                 XD(XI)=XD(XI)+1
              ELSE
                 XD(29)=XD(29)+1
              ENDIF
           ENDIF
81      CONTINUE
        DO 82 I=1,29
           PE(I)=(100.0*XD(I)/L4)
82      CONTINUE
        PMI=(100.0*XDM/L4)
        VARI=VARI/(L4-1)
C
C       STATISTICAL RESULTS FOR THE MCC PACKETS.
C
        MMEAN1=MMEAN1/ML4
        MVARI=0
```

```
      MXDM=0
      DO 133 I=1,ML4
         MVARI=MVARI+((MDELAY(I)-MMEAN1)**2)/(PL**2)
         XDMM=MDELAY(I)/1024
         IF (XDMM.EQ.1) MXDM=MXDM+1
         XI=MDELAY(I)/5120+1
         IF (XI.LT.23) THEN
            MX(XI)=MX(XI)+1
         ELSE
            XI=MDELAY(I)/20480.0+17.5
            IF (XI.LT.29) THEN
               MX(XI)=MX(XI)+1
            ELSE
               MX(29)=MX(29)+1
            ENDIF
         ENDIF
133   CONTINUE
      DO 134 I=1,29
         MP(I)=(100.0*MX(I)/ML4)
134   CONTINUE
      MPMI=(100.0*MXDM/ML4)
      MVARI=MVARI/(ML4-1)
C
C STATISTICAL RESULTS FOR THE MCC AND NODE PACKETS.
C
      TVARI=0
      TXDM=0
      J1=L4+1
      I1=0
      DO 91 I=J1,TL4
         I1=I1+1
         DELAY(I)=MDELAY(I1)
91    CONTINUE
      DO 92 I=1,TL4
         TVARI=TVARI+((DELAY(I)-TMEAN)**2)/(PL**2)
         XDMM=DELAY(I)/1024
         IF (XDMM.EQ.1) TXDM=TXDM+1
         XI=DELAY(I)/5120+1
         IF (XI.LT.23) THEN
            TD(XI)=TD(XI)+1
         ELSE
            XI=DELAY(I)/20480.0+17.5
            IF (XI.LT.29) THEN
               TD(XI)=TD(XI)+1
            ELSE
               TD(29)=TD(29)+1
            ENDIF
         ENDIF
92    CONTINUE
      DO 94 I=1,29
         TP(I)=(100.0*TD(I)/TL4)
94    CONTINUE
```

```
      TPMI=(100.0*TXDM/TL4)
      TVARI=TVARI/(TL4-1)
      MEAN=MEAN/1024.0
      MMEAN1=MMEAN1/1024.0
      TMEAN=TMEAN/1024.0
      IDL=(TIME/1024)-L4
C
C OUTPUT THE RESULTS AT THE END OF THE PROGRAM.
C
      IF (K1.EQ.1) PRINT *,'                    MAXIMUM DELAYS'
      IF (K1.EQ.2) PRINT *,'                    MINIMUM DELAYS'
      PRINT *,'|-------------------------------------------------------|'
      PRINT *,'|   1-2     2-5   5-10   10-15   15-20   20-25   25-30 30-35'
83    FORMAT (2X,I6,7I7)
      WRITE (6,83) XDM,XD(1)-XDM,XD(2),XD(3),XD(4),XD(5),XD(6),XD(7)
101   FORMAT (2X,I6,7I7)
      WRITE(6,101) MXDM,MX(1)-MXDM,MX(2),MX(3),MX(4),MX(5),MX(6),MX(7)
121   FORMAT (2X,I6,7I7)
      WRITE(6,121) TXDM,TD(1)-TXDM,TD(2),TD(3),TD(4),TD(5),TD(6),TD(7)
84    FORMAT (1X,8F7.3)
      WRITE (6,84) PMI,PE(1)-PMI,PE(2),PE(3),PE(4),PE(5),PE(6),PE(7)
102   FORMAT (1X,8F7.3)
      WRITE(6,102) MPMI,MP(1)-MPMI,MP(2),MP(3),MP(4),MP(5),MP(6),MP(7)
122   FORMAT (1X,8F7.3)
      WRITE(6,122) TPMI,TP(1)-TPMI,TP(2),TP(3),TP(4),TP(5),TP(6),TP(7)
      PRINT *,'  '
      PRINT *,'|  35-40   40-45   45-50   50-55   55-60 60-65 65-70 70-75'
85    FORMAT (1X,8I7)
      WRITE(6,85) XD(8),XD(9),XD(10),XD(11),XD(12),XD(13),XD(14),XD(15)
103   FORMAT (1X,8I7)
      WRITE(6,103) MX(8),MX(9),MX(10),MX(11),MX(12),MX(13),MX(14),MX(15)
123   FORMAT (1X,8I7)
      WRITE(6,123) TD(8),TD(9),TD(10),TD(11),TD(12),TD(13),TD(14),TD(15)
86    FORMAT (1X,8F7.3)
      WRITE(6,86) PE(8),PE(9),PE(10),PE(11),PE(12),PE(13),PE(14),PE(15)
104   FORMAT (1X,8F7.3)
      WRITE(6,104) MP(8),MP(9),MP(10),MP(11),MP(12),MP(13),MP(14),MP(15)
124   FORMAT (1X,8F7.3)
      WRITE(6,124) TP(8),TP(9),TP(10),TP(11),TP(12),TP(13),TP(14),TP(15)
      PRINT *,'    '
      PRINT *,'| 75-80   80-85   85-90   90-95   95-100 100-105 105-110'
87    FORMAT (1X,5I7,1X,2I8,I5)
      WRITE(6,87) XD(16),XD(17),XD(18),XD(19),XD(20),XD(21),XD(22)
105   FORMAT (1X,5I7,1X,2I8,I5)
      WRITE(6,105) MX(16),MX(17),MX(18),MX(19),MX(20),MX(21),MX(22)
125   FORMAT (1X,5I7,1X,2I8,I5)
      WRITE(6,125) TD(16),TD(17),TD(18),TD(19),TD(20),TD(21),TD(22)
88    FORMAT (1X,5F7.3,2X,3F7.3)
      WRITE(6,88) PE(16),PE(17),PE(18),PE(19),PE(20),PE(21),PE(22)
106   FORMAT (1X,5F7.3,2X,3F7.3)
      WRITE(6,106) MP(16),MP(17),MP(18),MP(19),MP(20),MP(21),MP(22)
126   FORMAT (1X,5F7.3,2X,3F7.3)
```

```
      WRITE(6,126) TP(16),TP(17),TP(18),TP(19),TP(20),TP(21),TP(22)
      PRINT *,'    '
      PRINT *,'¦110-130 130-150 150-170 170-190 190-210 210-230 OVER'
89    FORMAT (1X,7I8)
      WRITE(6,89) XD(23),XD(24),XD(25),XD(26),XD(27),XD(28),XD(29)
107   FORMAT (1X,7I8)
      WRITE(6,107) MX(23),MX(24),MX(25),MX(26),MX(27),MX(28),MX(29)
127   FORMAT (1X,7I8)
      WRITE(6,107) TD(23),TD(24),TD(25),TD(26),TD(27),TD(28),TD(29)
90    FORMAT (1X,7F8.3)
      WRITE(6,90) PE(23),PE(24),PE(25),PE(26),PE(27),PE(28),PE(29)
108   FORMAT (1X,7F8.3)
      WRITE(6,108) MP(23),MP(24),MP(25),MP(26),MP(27),MP(28),MP(29)
128   FORMAT (1X,7F8.3)
      WRITE(6,128) TP(23),TP(24),TP(25),TP(26),TP(27),TP(28),TP(29)
      PRINT *,'    '
      PRINT *,'-------------------------------------------------------¦'
      PRINT *,'¦ NUMBER OF NODES                                  =',NODES
      PRINT *,'¦ LOAD OF CHANNEL                                  =',L
      PRINT *,'¦ SUCCESFULLY TRANSM. PACKETS-MCC                  =',ML4
      PRINT *,'¦ AVERAGE TRANSMISSION DELAY-MCC                   =',MMEAN1
      PRINT *,'¦ VARIANCE TRANSMISSION DELAY-MCC                  =',MVARI
      PRINT *,'¦ SUCCESFULLY TRANSMITTED PACKETS-NODES            =',L4
      PRINT *,'¦ AVERAGE TRANSMISSION DELAY-NODES                 =',MEAN
      PRINT *,'¦ VARIANCE     >>          >>  >>>                 =',VARI
      PRINT *,'¦ SUCCESFULLY TRANSMITTED PACKETS-TOTAL            =',TL4
      PRINT *,'¦ AVERAGE TRANSMISSION DELAY-TOTAL                 =',TMEAN
      PRINT *,'¦ VARIANCE     >>          >>    >>                =',TVARI
      PRINT *,'¦ TIME ALL TRANSMISSIONS STOP                      =',TIME/1024
      PRINT *,'¦ TIME THE CHANNEL WAS IDLE                        =',IDL
      PRINT *,'¦ NUMBER OF PACKETS GENERATED                      =',100000+MCP
      PRINT *,'¦-------------------------------------------------------¦'
120   CONTINUE
110   CONTINUE
100   CONTINUE
      STOP
      END
```

APPENDIX D: LITERATURE REVIEW

The references below are a result of an extensive literature review associated with the project and are available upon request.

[1] ANSI/IEEE - International Standard 8802/3 "Carrier Sense Multiple Access with Collision detection (CSMA/CD) Access Method and Physical Layer Specification," IEEE Computer Society Press, 1985.

[2] R.M. Metcalfe and D.R. Boggs, "Ethernet Distributed Packet Switching for Local Computer Networks," Communication Ass. Comput. Mach., Vol. 19, no 7, pp. 395-403, 1976.

[3] T.T. Liu, L.Li and W.F. Franta, "A Decentralized Conflict-Free Protocol, GBRAM for Large Scale Local Networks," Computer Network Symposium Proceedings, pp. 39-54, Dec. 1981.

[4] ANSI/IEEE - International Standard 8802/5 "Token Ring Access," IEEE Computer Society Press, 1985.

[5] T.A. Gonsalves, "Measured Preformance of the Ethernet," Advances in Local Area Networks, edited by K. Kummerle, J.O. Limb and F.A. Tobagi, IEEE Press 1987, pp. 383-410.

[6] K. Kummerle and M. Reiser, "Local Area Networks - Major Technologies and Trends," in Local Area Networks, edited by K. Kummerle, J.O. Limb and F.A. Tobagi, IEEE Press 1987, pp. 2-27.

[7] W. Bux, F.H. Closs, K. Kummerle, H.J. Keller and H.R. Mueller, "Architecture and Design of a Reliable Token Ring Network," in Local Area Networks, edited by K. Kummerle, J.O. Limb and F.A. Tobagi, IEEE Press 1987, pp. 67-80.

[8] J.O. Limb and C. Flores, "Description of FASNET - A Unidirectional Local Area Communications Network," in Local Area Networks, edited by K. Kummerle, J.O. Limb and F.A. Tobagi, IEEE Press 1987, pp. 190-205.

[9] M.R. Finley Jr., "Optical Fibers in Local Area Networks," in Local Area Networks, edited by K. Kummerle, J.O. Limb and F.A. Tobagi, IEEE Press, 1987, pp. 224-243.

[10] J.D. Detreville and W.D. Sincoskie, "A Distributed Experimental Communications Systems," in Local Area Networks, edited by K. Kummerle, J.O. Limb and F.A. Tobagi, IEEE Press, 1987, pp. 533-542.

[11] D. Friedman and V. Haimo. SIMNET Network Performance. BBN Report 6711. BBN Communications Corp., Cambridge, MA, Jan 1988.

[12] A. Pope. The SIMNET Network and Protocols. BBN Report Number 6787. BBN Laboratories, Inc., Cambridge, MA May 1988.

[13] D.C. Miller, A.Pope and R.M. Waters, "Long Haul Networking of Simulators," Proceedings 10th Interservice/Industry Training Systems Conference, Nov 29 - Dec 1 1988, Orlando, Fl , pp. 577-582.

[14] I. Chlamtac, W.R. Franta and K.D. Levin, "BRAM The Broadcast Recognizing Access Method " IEEE Trans. on Communications, Vol. COM-27, No. 8, August 1989, pp. 1183-1189.

[15] F.T. Andrews, Jr., "ISDN 83," special issue on ISDNs, IEEE Communications Magazine, Vol. 22, No. 1, Jan 1984, pp 6-10.

[16] D.J. Kostas, "Transition to ISDN - An Overview," Special issue on ISDNs, IEEE Comm Mag., Vol. 22, No 1, Jan 1984, pp. 11-17.

[17] N.Q. Duc and E.K. Chew, "ISDN Protocol Architecture," IEEE Comm. Mag., Vol. 23, No.3 March 1985, pp. 15-22.

[18] M. Ross, "Design Approaches and Performance Criteria for Integrated Voice/Data Switching," Proc. IEEE, Vol. 65, No.9 Sept. 1977, pp. 1283-1295.

[19] M. Schwartz. Telecommunication Networks. Addison and Wesley, 1987.

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

[21] H.A. Taha. Simulation Modeling and SIMNET. Prentice-Hall, 1988.

APPENDIX E

SIMULATION NETWORKING AND PROTOCOL ALTERNATIVES

PRESENTED AT THE INSTITUTE FOR SIMULATION AND TRAINING'S
SYMPOSIUM FOR
INTERACTIVE NETWORKED SIMULATION FOR TRAINING

APRIL 1989
ORLANDO, FL

SIMULATION NETWORKING AND PROTOCOL ALTERNATIVES

M. Bassiouni, Department of Computer Science
M. Georgiopoulos, Department of Electrical Engineering
J. Thompson, Institute for Simulation and Training

Graduate Student Assistants: S. Chaterjee, M. Chiu and N. Christou

University of Central Florida
Orlando, FL 32816

ABSTRACT

In this paper, we focus on the implementation of an efficient local area network (LAN) which will be used to interconnect simulation training devices. In particular, we present preliminary efforts in modeling and analyzing the performance of three different network protocol access methods: CSMA/CD (Carrier Sense Multiple Access with Collision Detection), Virtual Token-Passing Bus Access Protocols and Token-Ring Access. A detailed discussion of the advantages and disadvantages of the above access protocols and anticipated results are also presented.

INTRODUCTION

The networking of simulation training devices departs from the traditional use of computer networks whose purpose is to allow for the sharing of computing resources among multiple computers. In the application of networking simulators, the network is used almost exclusively for communication of process state information between training devices engaged in the training exercise.

There are many inherent limitations to using a network in this application. For example, as the number of simulators on the network and workload per simulator increases, there will be a deterioration in throughput and degradation of other performance measures. If throughput delays become significant, the effectiveness of a real-time training simulation may be overly compromised due to the time-critical response requirements in the simulation of true-to-life, action-requiring training scenarios. Depending upon communication protocols, there may also be an increase in the frequency of retransmissions and lost or distorted messages. The magnitude of this problem is functionally related to how data is distributed throughout the system, and the soundness of the network access and internal network protocols.

Various choices exist for the implementation of a local area network (LAN), (e.g. transmission medium, topology, access protocols, etc.) to interconnect simulation devices. In this paper, we present efforts in modeling and analyzing the performance of three different network protocol access methods. In particular, the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) such as ETHERNET (ANSI/IEEE 802.3 Standards [1,2]), the Virtual Token-Passing bus protocols such as the Generalized Broadcast Recognizing Access Methods (GBRAM) [3], and Token-Ring Access protocols (ANSI/IEEE 802.5 Standards [4,5]) are examined.

SYSTEM MODEL

Our system consists of a complex web of armor, fixed and rotary wing aircraft, and air-defense simulated vehicles linked together via a Local Area Network (LAN) to create a simulated world in which war-gaming can be conducted. In our system, combat forces and their commanders must move, shoot, communicate and navigate just as they do in a real battle. Hence, a tremendous amount of information must be exchanged among the simulators in real-time if a realistic battle scenario is to be created.

Local Area Networks can be characterized by the following factors:

- transmission medium (coaxial cable, twisted pair, optical fiber)
- modulation scheme (baseband, broadband)
- wiring scheme (bus or ring)
- medium-access control schemes (random-access or controlled-access).

We intend to investigate the capability of three LAN's to interconnect the simulators. Two of these LAN's are **bus networks**, which utilize baseband transmission to send messages over a coaxial cable. The medium-access control schemes for one is the ETHERNET protocol [2] and for the other is Generalized Broadcast Recognizing Access Method (GBRAM) protocol [3]. The third LAN is a **ring network**, which utilizes baseband transmission to send messages over a fiber optic cable. Its medium-access control scheme is a token passing protocol.

In the ETHERNET protocol, if a simulator, or other node, has a packet ready to transmit onto the network, it monitors the network to determine whether any transmissions are in progress. If a transmission is in progress, the network is said to be "busy", otherwise, it is "idle". If the node finds the network busy, transmission of the data packet is deferred. When it finds the network idle, packet transmission is initiated. If multiple nodes attempt to transmit at the same time, their transmissions interfere, or collide. The collision is acknowledged by each transmitting node sending out a bit sequence onto the network referred to as a "jam-signal". After the jam-signal has been transmitted, the nodes involved in the collision schedule a retransmission attempt at a randomly selected time in the future.

In the GBRAM protocol, the nodes employ a "virtual-token" scheme in which each node gains network access (the virtual token) at a unique time which is determined by a decentralized scheduling function, hence avoiding collisions completely.

The Token-ring access protocol is even more straight-forward. A node gains the right to transmit onto the network when it detects and captures a free token passing on the network medium. The token is a control signal that circulates on the medium following each information transfer. Any node, upon detection of a free token, may capture the token, set it to busy, and then send its packet. Upon completion of transmitting its data, and after appropriate checking for proper operation, the node generates and transmits a "free token" which begins circulating around the network and provides other nodes the opportunity to gain network access.

Bus Network Topology System Configuration

The system configuration corresponding to the bus network topology is shown in Figure 1. In this CSMA/CD (ETHERNET) implementation, up to eight simulators or other types of nodes can be connected through an ETHERNET multi-port transceiver to a single point on the ETHERNET coaxial cable, via a media-access unit (vampire tap). A single coaxial cable is available to link all simulators together.

Some important parameters pertaining to the implementation of the ETHERNET and the GBRAM protocols are as follows:

- the time that it takes for a message to traverse the medium
- the time elapsed from the instant the coaxial cable becomes idle or becomes busy until the node "realizes" that the cable is idle or busy
- the time elapsed from the moment that a transmitting node realizes that it is involved in a collision until it generates the first bit of the jam-signal

Ring Network Topology System Configuration

The system configuration corresponding to the ring network topology is shown in Figure 2. A ring network consists of a closed sequence of individual point-to-point (node-to-node) links. For efficient operation, the token protocol dictates a minimal delay per station, and the ability to change a single bit in the data stream (e.g. the token) "on-the-fly". An important parameter pertaining to the implementation of a token ring protocol is the time it takes for the data to propagate through a node on the network.

Node Traffic Generation

Each node generates a certain amount of traffic into the network. In the simulation of the network traffic, some of the options for the packet inter-arrival time at a node site are:

- Exponential - the traffic generated by the simulator is a Poisson process
- Fixed with a specified percentage of "jitter" - a fixed time, plus or minus a random time within the specified percentage of the fixed time
- Uniformly distributed in a specified interval
- Trace-driven - the traffic used to drive the network is a trace of real network traffic data.

One of the nodes in our network operates differently from ordinary simulator units. It produces network packets for a large quantity of different types of simulated vehicles. It transmits the data packets for a portion of its simulated vehicles at regular intervals. Hence, its traffic can be characterized as periodic.

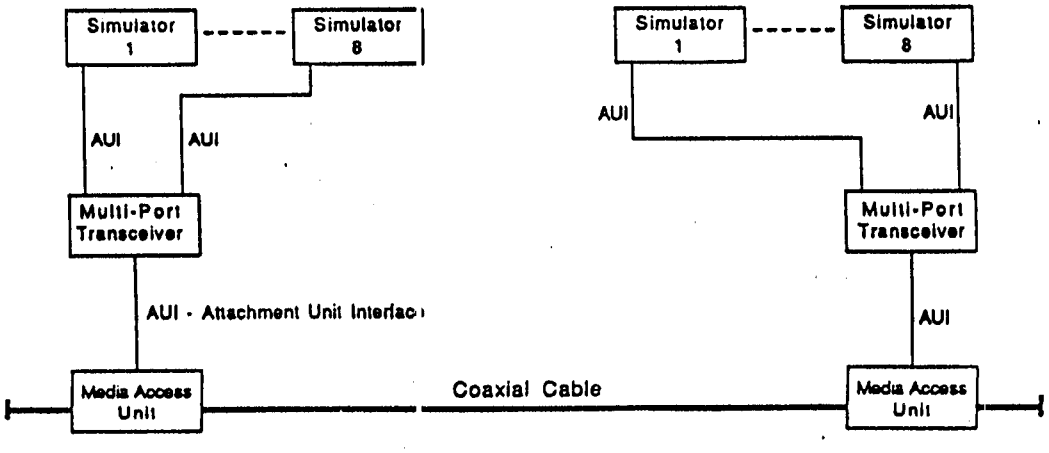


Figure 1. Bus Network Topology System Configuration

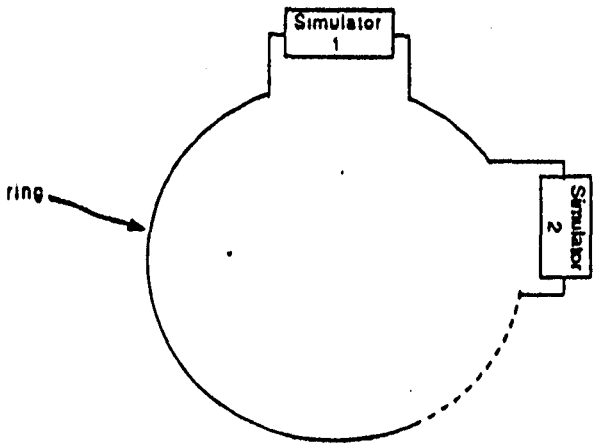


Figure 2. Ring Network Topology System Configuration

THE ETHERNET SIMULATION MODEL

In this section, we give a high-level description of the simulation model used in evaluating and predicting the performance of the CSMA/CD implementation of ETHERNET. The simulation model is written in Concurrent-C (an extension of the C programming language with concurrent programming facilities based on the "rendezvous" concept). The powerful synchronization and concurrency aspects of Concurrent-C [6] have provided us with a notationally convenient and conceptually elegant tool for modeling the parallel activities of the simulation network nodes and the underlying networking layer.

The process interaction model of Concurrent-C has been used in our simulation to map the different entities and activities of the simulated network to corresponding Concurrent-C processes. The following process types are the major generic entities used in our simulation. Figure 3 gives a block diagram showing the interactions among these different processes.

- Process **Simnode** is used to represent a vehicle simulator on the network. A process of this type is created for each such simulator.
- Process **Busnode** is used to represent the point of contact of each network node with the ETHERNET bus (coaxial cable). A process of this type is created for each such point of contact on the bus.
- Process **Lserver** is used to implement and control the flow of data (packets and jam signals) in the direction from right to left for each network node. A process of this type is created for each network node.
- Process **Rserver** is analogously defined for traffic flowing in the direction from left to right.
- Process **Scheduler** is used to order time events and control the sequencing of activities of the entire simulation.

Typically, eight simulators connect to the coaxial transmission cable at a single point via a multi-port transceiver. Each of the simulators is modeled as a **Simnode** process. A **Busnode** process for each point of contact is created to receive and transmit local traffic from any one of the eight network nodes, as well as retransmit any external messages arriving at the node. For this purpose, we use two separate processes called **Rserver** and **Lserver**. The **Rserver** process implements the transfer of data from its left **Busnode** process to its right **Busnode** process. This transmission is actually simulated by calling the **Scheduler** process to wait for the propagation delay (the time needed for the message to travel from one network node to the next). The **Lserver** similarly carries data signals from the right **Busnode** to its left neighbor. The **Busnode** process detects collisions of transmitted data by checking for the existence of local traffic, left traffic or right traffic.

The Simnode Process

This process is the source of local traffic. It generates packets according to a specified input method (e.g. using traces of real data or random stochastically generated inter-arrival times such as exponential, uniform, fixed with jitter, etc.). Upon arrival of a local packet, the **Simnode** process makes a request to the corresponding **Busnode** process in order to transmit the new packet. This is done by calling a specific transaction in the **Busnode** process as illustrated by the code presented later. At this point, the **Busnode** process checks for a carrier flag. If the flag has been off for at least the inter-frame gap, the **Simnode** process can proceed with its transmission. If the carrier flag is on, the **Simnode** process must wait for the inter-frame gap

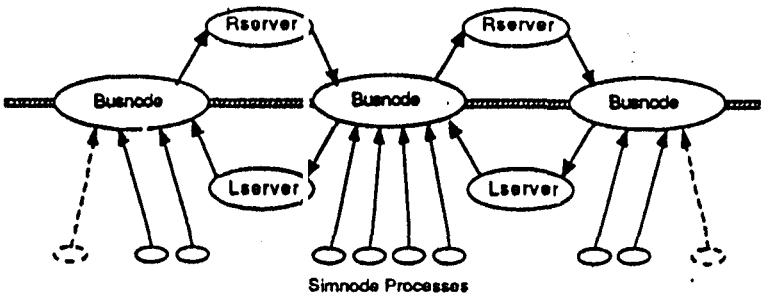


Figure 3. ETHERNET Simulation Model Process Interactions

and then retry its transmission. When a collision is detected during transmission, the **Simnode** process generates and transmits a jam signal and increments the collision counter. This is followed by invoking a back-off algorithm for retransmission. A packet is discarded after 16 unsuccessful transmission attempts. The specification and major activities of the **Simnode** process are described by the following code:

```
Process spec Simnode (process sched s,
                      process bus bid,
                      long meanlit, name_t name)

Process body Simnode (s, bid, meanlit, name)
  /* Initialization phase */
    c_setname(c_mypid(), name.str);
    s.adduser(); bid.addProd();

  /* Main processing phase */
    while (not done) do
    /* get arrival time */
    t=erand(meanlit)
    /* call Scheduler to wait for arrival */
    loc_traffic.arrive = s.wait(s.reqDelay(t));
    /* attempt transmission */
    while ((dt = bid.transReq(c_mypid())!= 0)
       s.transDelay(dt);
    /* code for collision check and
    subsequent backoff algorithm */
    collision_handler (collis_counter);
    /* Termination phase */
    statistic_fun();
```

The Busnode Process

The **Busnode** process acts like a server process ready to accept transaction calls from the local **Simnode** processes, the **Lserver** processes or the **Rserver** processes. The **Busnode** is responsible for detecting collisions and it continuously monitors the carrier flag to see if it is busy. In the case of a collision, the **Busnode** process calls the **Scheduler** to awaken the transmitting **Simnode** process which then stops transmission and sends the jam signal. The following code gives the Concurrent C specification of the **Busnode** process.

```
Process spec Busnode (process sched s,
                      process Rserver idr,
                      process Lserver idl,
                      process Simnode name)

  /*transactions to change producer count */
  trans void addProd(), dropProd();
  /* transactions to change consumer count */
  trans void addCons(), dropCons();
  /* transaction to handle right to left traffic: */
  trans put_FRTL(type); /* type can be start,
                           completion or jam */
  /* transaction to handle left to right traffic: */
  trans put_FLTR(type);
  /* transaction to transmit local traffic */
  trans done(type);
  /* transaction to accept requests */
  trans trans_req(send_id); /* for Simnode */
  trans takereq(); /* for Rserver & Lserver */
```

The Rserver and Lserver Processes

These processes transmit the traffic delivered to the **Busnode** process by any transmitting **Simnode** process to the left and/or right. The specification and body of the **Rserver** process are given below.

```
Process spec Rserver(process sched s,
      process Busnode inbus,
      process Busnode outbus, Process Simnode
name)

  Process body Rserver(s, inbus, outbus, name)
  typedef struct /* data submitted by Simnode */
    { /* time of arrival */
    long arrive;
    /* Packet length */
    int packet_length;
    /* No. of update messages per sec */
    int update_num;
    /* No. of attempts to transmit */
    int attempt_index;
      } local_traffic

  /* Initialization phase */
    c_setname(c_mypid(), name.str);
    s.adduser(); inbus.addCons();
    outbus.addProd();
  /* Main processing phase */
    while (takereq(1)) {
    /* wait for propagation delay */
    t = arrivaltime + propagation delay;
    ts = s.wait(s.reqDelay(t));
    /* deliver message */
    put_FLTR(type);
    }
```

The Scheduler Process

Delays in the simulated network (such as transmission delays) are handled by the **Scheduler** process. This process maintains the simulated clock and advances it appropriately. For each delay request from a process, the **Scheduler** determines the time when the process needs to be reactivated and saves this time in an "activation request" list. When all processes are waiting, the scheduler picks the next process to run, advances the simulated clock and reactivates the process. The simulated clock advances only when all processes are waiting; thus any (non-delay) computation done by a process takes place in zero simulated time. At any given moment, each client process is in one of the following three states:

- Waiting: for an explicit delay request from the **Scheduler**;.
- Active: computing in zero simulated time;
- Passive: waiting for an event other than a delay request from the **Scheduler**.

The specification and the body of the Scheduler are given below.

```
process spec sched()
/* return current simulated time */
trans long now();
/* request a delay */
trans long reqDelay(long);
/* wait for a reqDelay */
trans long wait(long);
/* add or delete client process */
trans void adduser(), dropuser();
/*change client to new state */
trans void passive(), active()

/* handle collision */
trans void collision(id);
typedef struct {
/* structure describing a delay request */
long ts; /* time stamp */
int next; /* index of next entry or -1 */
/* number of clients waiting for this time */
int nwait;
} reqent;
static reqent Rtab MAXREQ ;
static int Ifree; /* first free entry */
/* Ihead is entry with lowest timestamp */
static int Ihead = -1;

process body sched()
  int nclients, nactive, i;
long curtive = 0;
/* initialization phase */
c_setname(c_mypid(),"sched");
rqInit();
accept adduser() nclients = nactive = 1
/* main processing phase: accept requests while
clients exist */
while (nclients >0)
{
select
accept adduser() nclients += 1; nactive += 1;
or
accept dropuser() nclients -= 1; nactive -= 1;
or
accept passive()  nactive -=1;
or
accept active()  nactive +=1
or
accept now()  treturn curtime;
or
accept reqDelay(x)
nactive -= 1; treturn (addreq(curtime+x));
or
accept jam(id)
change timestamp of record with this id
}

/* If all clients are waiting, find the first event */
/* and allow all clients waiting for it to proceed */
if (nactive == 0 && Ihead != -1)
curtime = RtabIhead.ts;
nactive = RtabIhead.nwait;
While (--RtabIhead.nwait >=0)
accept wait(key) such that (key == Ihead)
treturn curtime;
```

In addition to the above entities, several other auxiliary processes/routines are used to collect/print statistics and appropriate performance measures, perform consistency checks, print error messages, create and initialize all required processes, and start/terminate the concurrent simulation. The software system is written in a modular fashion with emphasis on ease-of-modification and the use of parameterized values that facilitate the testing of a wide range of network characteristics and the simulation of different load conditions and different network parameters.

PERFORMANCE CHARACTERISTICS OF MEDIUM-ACCESS PROTOCOLS

In the real-time networking of simulators, two performance aspects are of particular interest: the delay-throughput characteristic of the medium-access control schemes, and network system behavior under heavy traffic loads. These characteristics will be considered for the general classes of contention and non-contention (token passing) protocols.

Contention Protocols

Contention protocols such as ETHERNET perform well in environments with a large number of bursty (ratio of average to high traffic is small) users. For its reliable operation, however, the ETHERNET bus protocol requires that a transceiver must be capable of detecting the weakest other transmitter on the network during its own transmissions, and of distinguishing the signals from other transmitters from the echoes of its own transmitter. Because of this, the use of high-quality coaxial cable is required to cover longer distances and a limitation on the maximum distance which can be covered by a single segment network cable is imposed.

An advantage of the bus structure (where ETHERNET and GBRAM operate) over the ring structure is that users attached to the bus are passive units, while users connected to the ring are active units. An immediate consequence of this observation is that if a node on the ring breaks down it can bring the entire network system down. This is highly unlikely to happen in the bus configuration.

A disadvantage of contention protocols is there is no guarantee of packet delivery time due to the undeterministic nature of contention and collision/back-off.

Token Passing Protocols

An advantage of token passing protocols is that they are much less sensitive to increased transmission rates and smaller packet lengths compared to contention protocols . and they operate more efficiently with longer length cables than the contention protocols [5]. Furthermore, since token passing protocols are conflict free, a maximum packet delivery can be guaranteed for a given number of users, making them desirable protocols for real-time applications.

Token-Ring LAN's [5] offer other advantages including the following:

- Because of its point-to-point connection property, rings readily accommodate the use of optical fiber as a transmission medium. In addition to offering reduced size and weight, and enhanced safety features, optical fiber also offers very high signal bandwidth (100 Mbps for fiber token-rings).

- Token-rings easily provide a priority-based scheme for packet transmission across the network. This is because the token has bits indicating the priority assigned to it, thereby providing multiple levels of access to the ring. In simulator networking this means that it will be possible to assign priorities to the different types of messages in order to optimize real-time performance and visual display at peak load conditions.

- The technological advantages enjoyed by bus topologies to date are about to disappear. Inevitably, VLSI technology and other near-term advances will soon be supplying the industry with ring chips and off-the-shelf ring attachments at the same low cost as bus chips. This low cost, combined with their reliability and ease of configuration and implementation, will make token-ring LAN's a very promising tool for simulator networking.

CONCLUSIONS

In this paper we have described an ongoing effort to model and evaluate the performance of three different network protocol access methods suitable for networking of simulation training devices: a contention access method based on the CSMA/CD (ETHERNET) protocol, and two contention-free methods based on Virtual Token Bus Access such as GBRAM and Token-Ring Access protocols. The system models pertaining to the above three access methods were addressed and a high-level description of a detailed simulation software system implemented for evaluating the performance of an ETHERNET scheme was given.

The models developed for the three access methods will enable us to perform a comparison study and evaluate different design decisions. Some of the numerical performance measures that will be gathered by the models are:

- The overall throughput of the network.
- The utilization of the transmission medium.
- The collision ratio for contention access.
- The average delay time per packet.
- The average ratio of lost packets (data loss rate).
- The relationship of the number of nodes on the network and the above parameters.
- The effect of packet arrival rates on network performance.

The models developed under this effort offer a very flexible tool for the evaluation and analysis of important classes of networking schemes that can be used to interconnect large numbers of real-time simulation training devices. Further investigations will be carried out to perform a comparison study of the three access methods and to evaluate different design decisions aimed at improving the overall throughput and enhancing the capability of simulation networks.

ACKNOWLEDGEMENTS

This work is supported by the U.S. Army Program Manager for Training Devices (PM TRADE) under Broad Agency Announcement # 88-01. The opinions expressed herein are those of the authors' and not necessarily those of the U.S. Government. The authors would like to thank J. Cadiz and E. Stadler for their help in obtaining and analyzing network traffic data and preparing this report.

REFERENCES

[1] ANSI/IEEE - International Standard 8802/3 "Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specification", IEEE Computer Society Press, 1985.

[2] Metcalfe, R. M. and Boggs, D. R., "Ethernet Distributed Packet Switching for Local Computer Networks", Communication Ass. Comput. Mach., Vol. 19, no. 7, pp. 395-403, 1976.

[3] Liu, T. T., Li, L. and Franta, W. R. "A Decentralized Conflict-Free Protocol, GBRAM for Large Scale Local Networks", Computer Network Symposium Proceedings, pp. 39-54, Dec. 1981.

[4] ANSI/IEEE - International Standard 8802/5 "Token Ring Access", IEEE Computer Society Press, 1985.

[5] Dixon, R., Strole, N. and Markov, J. "A token ring network for local data communication", IBM System Journal, Vol. 22, 1983, pp.62-74.

[6] Gehani, N. and Roome, W. "Concurrent C" Technical Report, AT&T Bell Laboratories, 1986.

ABOUT THE AUTHORS

M. A. Bassiouni received his Ph.D. degree in Computer Science from Pennsylvania State University in 1982. He is currently an Associate Professor of Computer Science at the University of Central Florida, Orlando. His current research interests include computer networks, distributed systems, databases, and performance evaluations. He has authored several papers and has been actively involved in research on local area networks, concurrency control, data encoding, I/O measurements and modeling, schemes of file allocation and user interfaces to relational database systems. Dr. Bassiouni is a member of the IEEE Computer Society, the Association for Computing Machinery, and the American Society for Information Science.

M. Georgiopoulos received his Ph.D degree in Electrical Engineering from the University of Connecticut in 1986. He is currently an Assistant Professor in the Department of Electrical Engineering and Communication Sciences at the University of Central Florida, Orlando. His current research interests include multi-user communication theory, communication networks, computer networks, and spread spectrum communications. Dr. Georgiopoulos is a member of IEEE and the Technical Chamber of Greece.

J. Thompson received his BS degree in Electrical Engineering from the University of Central Florida in 1978. He is currently a Research Associate for the Institute for Simulation and Training (IST) at the University of Central Florida, Orlando. Mr. Thompson has technical responsibility for all IST research activities involving computer networking.

APPENDIX F

REAL TIME SIMULATION NETWORKING: NETWORK MODELING AND PROTOCOL ALTERNATIVES

TO BE PRESENTED AT THE
11TH INTERSERVICE/INDUSTRY TRAINING SYSTEMS CONFERENCE

NOVEMBER 1989
FT. WORTH, TX

Real Time Simulation Networking: Network Modeling and Protocol Alternatives

M. Bassiouni, Department of Computer Science
M. Georgiopoulos, Department of Electrical Engineering
J. Thompson, Institute for Simulation and Training

Graduate Student Assistants: S. Chaterjee, M. Chiu and N. Christou

University of Central Florida
Orlando, FL 32816

ABSTRACT

In this paper, we present the findings of a comparison study using predictive detailed simulation models for three different network protocol access methods: Carrier Sense Multiple Access with Collision Detection (ANSI/IEEE 802.3 STD), Token-Passing Bus Access (ANSI/IEEE 802.4 STD) and Token-Ring Access (ANSI/IEEE 802.5 STD). Discussions of network performance, the implications of the results of the comparison study, and the insight gained from this project for improving real-time simulation networking are presented.

INTRODUCTION

A local area network (LAN) is a geographically confined communication system that uses a shared transmission medium. Various choices usually exist for the main ingredients of LAN (i.e., transmission medium, topology, access protocols, etc.), with each exhibiting advantages and providing benefits that depend on the objectives of the LAN. The ability to model, analyze and evaluate the impact of these choices on network performance is essential to ensuring maximum utilization of the LAN.

One of the pioneering LAN's for connecting computers was a bus-based ETHERNET developed by Xerox Corp. in the early 1970's. The contention access method used by each node in ETHERNET is based on a pre-emptive protocol of first listening for network activity and then broadcasting the message onto the network. If a collision with another message occurs, each sender (node) backs-off from transmitting their message for a random period of time and then attempts the transmission again. This access technique is known as Carrier Sense Multiple Access with Collision Detection (CSMA/CD) [1, 2]. Standards for CSMA/CD protocols such as ETHERNET are known as IEEE 802.3 standards approved by the American National Standards Institute (ANSI).

The networking of real-time, interactive simulation training systems departs from the traditional use of a computer network whose function is to provide sharing of computing resources among multiple users (nodes) on the network. When used to interconnect real-time training simulators, the network is used almost exclusively for communication of process state information between the simulators engaged in the training exercise.

There are many inherent limitations to using a network in this application. For example, as the number of simulators on the network and the workload per simulator increases, there may be a deterioration in throughput and a degradation of other network performance parameters. If throughput delays become too large, for example, the effectiveness of a real-time training simulation may be overly compromised due to the time-critical response requirements in the simulation of true-to-life, action-requiring training scenarios. Depending upon the network communication protocol being used, there may also be an increase in the frequency of retransmitted and lost or distorted messages.

Recently, there has been a tremendous interest in LAN's implemented using the non-contention class of network protocols known as Token-Passing protocols. Two schemes falling under this class are Token-Ring and Virtual Token-Bus protocols . In a Token Ring LAN, a distinctive bit sequence, called a token, is passed from one node to another in order to signify the availability of the network medium for the transmission of data for that node. Possession of the token by the node gives it,and only it, permission to transmit across the network, as opposed to having all nodes contend for this privilege. In a Virtual Token-Bus LAN, a virtual, or imaginary token, is passed from user to user thus providing access to the network. This virtual token is actually a predetermined instant in time when each user knows it is its turn to access the network. Each of these three protocols will be discussed in detail in later sections.

The primary goal of this research effort has been to develop predictive/analytical models for network performance of three LAN configurations operating under real-time, interactive simulation/training constraints. Two of these LAN's are **bus networks** which utilize baseband transmission to send messages over a coaxial cable which is common to all users. The medium-access control schemes for the first is ETHERNET which is a member of the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol family and for the second is the Generalized Broadcast Recognizing Access Method (GBRAM) [3] which is a member of the Virtual Token-Bus protocol family. The third LAN is a **ring network**, which sends its messages over either coaxial or fiber optic cable. Its medium-access control scheme is the Token-Ring [4] protocol.

NETWORK MEDIUM-ACCESS PROTOCOLS

ETHERNET

CSMA/CD protocols, including ETHERNET, are characterized by their distributed network control whereby each node on the network determines its own channel access time based only on information available from the common network channel (bus). When a node is ready to transmit a message onto the network, it first monitors the network bus to determine whether any other transmissions from other nodes are in progress. If the node senses the network channel to be busy, it simply waits for the channel to become idle

before attempting to transmit its message. Once the channel is sensed to be idle, the node waits a pre-specified amount of time to assure the channel is clear and then begins transmitting its message. During its own transmission, the node also monitors the channel in order to detect whether its message is interfering (colliding) with messages from other nodes. If a collision is detected, each node involved in the collision transmits a bit sequence onto the network known as a **jam signal**, after which each node involved in the collision waits (backs-off) for a randomly generated amount of time before reattempting its transmission.

The performance of contention protocols is directly related to how efficiently nodes avoid collisions and handle retransmissions. The problem of data collisions is directly related to the network traffic load.

Token-Ring

In Token Passing protocols [4, 5], which the Token-Ring LAN is a member, there is no contention for the network channel because only one node at a time is allowed to access the channel. In Token-Ring LAN's this is accomplished by arranging the nodes in a serial ring configuration such that the network channel actually passes through each node. The **token** is a control signal that circulates around the channel. An individual node gains the right to transmit onto the network when it first detects, and then captures a **free token** passing on the channel. Once a node captures the **free token**, it changes it to a **busy token** and begins transmitting its message onto the network. Upon completing its message transmission, the node generates and transmits a **free token** which begins circulating around the network channel, thus providing other nodes the opportunity to gain access to the network.

Generalized Broadcast Recognizing Access Method (GBRAM)

The GBRAM protocol is also a member of the Token Passing protocol family. It differs significantly,however, from the Token-Ring protocol. In the GBRAM, rather than each node having to capture the **free token** from the network to gain transmission access, an imaginary (**virtual**) **token** is passed from node to node achieving the same result. The **virtual token** scheme provides each node access to the network at a unique time instant which is determined by a decentralized scheduling function.

NETWORK SYSTEM CONFIGURATION MODELS

Bus Network

The bus network configuration (including both ETHERNET & GBRAM) is shown in Figure 1. In this implementation, up to eight nodes can be connected through a multi-port transceiver to a single point on the coaxial cable, via a media-access unit. A single coaxial cable links all nodes together.

Ring Network

The ring network configuration (Token-Ring) is shown in Figure 2. The ring network consists of a closed sequence of individual point-to-point (node-to-node) connections.

NETWORK PROTOCOL COMPUTER SIMULATIONS

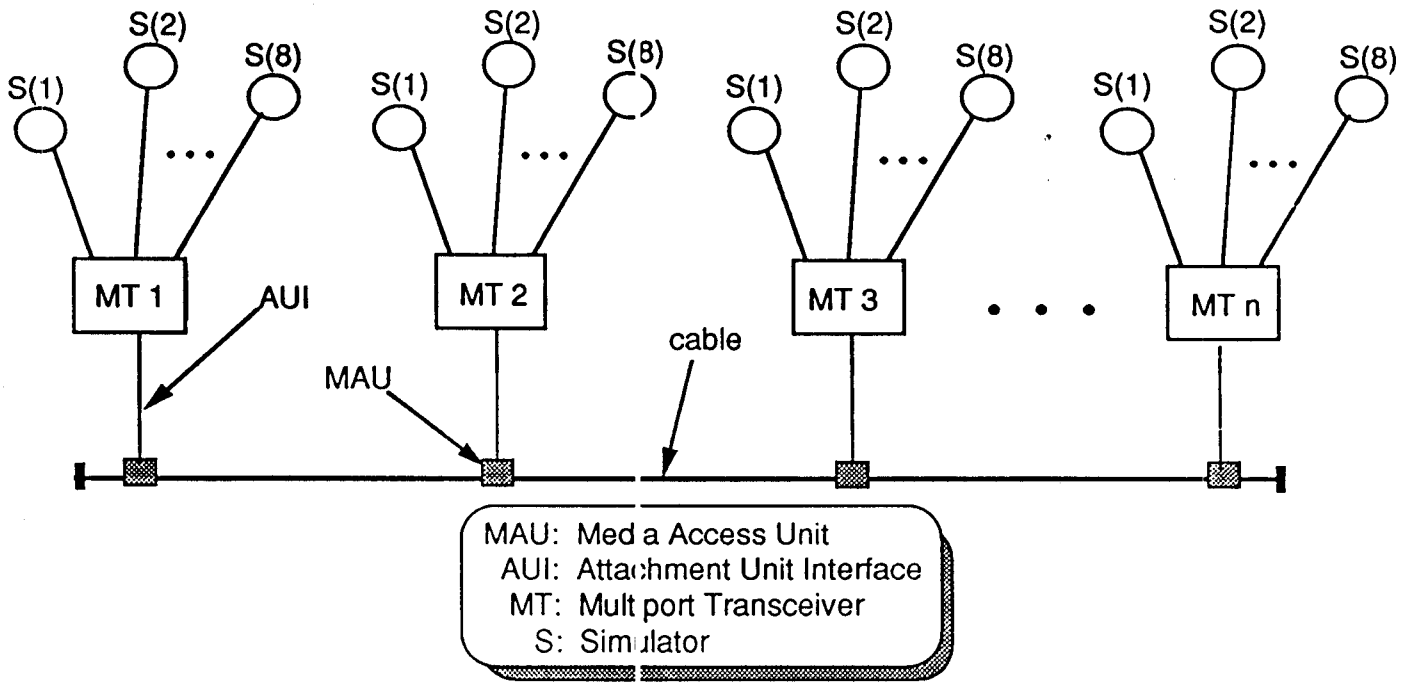


FIGURE 1. Bus Network Topology System Configuration

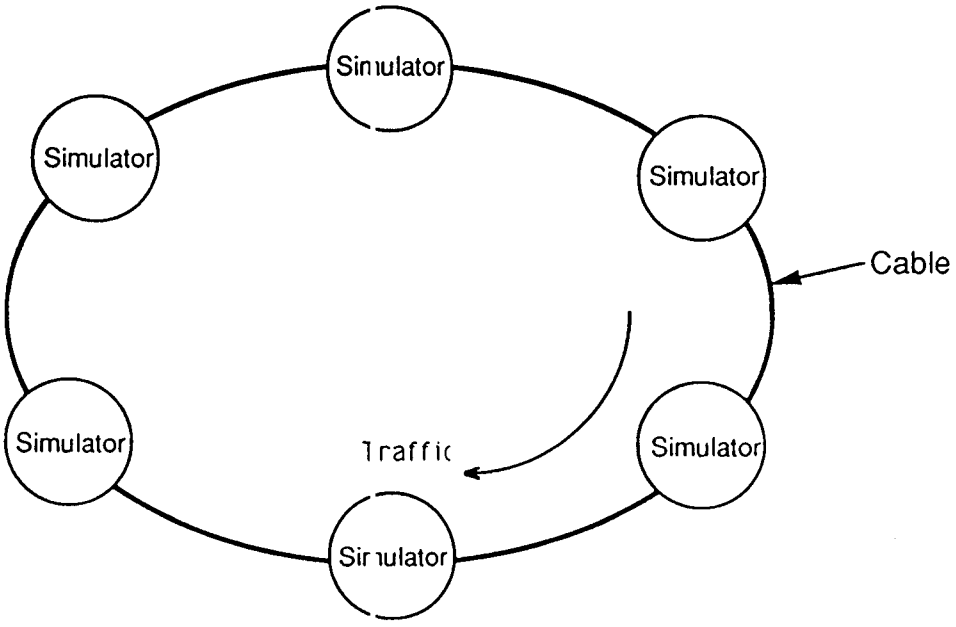


FIGURE 2. Ring Network Topology System Configuration

Simulation Models

The simulation models for both the bus and ring LAN topologies are written in **Concurrent C** (an extension of the **C** language with concurrent programming facilities based on the "rendezvous" concept). The powerful synchronization and concurrency aspects of **Concurrent C** [6] have provided us with a notationally convenient and conceptually elegant tool for modeling the parallel activities of LAN nodes and the underlying networking layer.

A functional diagram of the simulation model for the bus topology is shown in Figure 3. The following process types are the major generic entities used in our simulation for the bus structure.

Process **Simnode** is used to represent a vehicle simulator on the network. A process of this type is created for each such simulator. This process is the source of local traffic and is capable of generating packets according to a specified input method (e.g., using traces of real data or random stochastically generated inter-arrival times such as exponential, uniform, fixed with jitters, etc.)

Process **Busnode** is used to represent the point of contact of each network node with the bus (coaxial cable). A process of type **Busnode** is created for each such point of contact on the bus. Upon receiving a transmission request from a **Simnode** process, the **Busnode** attempts to fulfill the request based on the medium access protocol of the LAN. For example, in the CSMA/CD case, the **Busnode** process checks for a carrier flag and will allow transmission only if the flag has been off for at least the interframe gap. If a collision is detected during transmission, the **Simnode** process sends a jam signal and increments the collision counter. This is followed by invoking a back-off algorithm for retransmission.

Processes **Lserver** and **Rserver** are used to simulate the propagation delay and control the flow of data packets and jam signals in the direction form right to left and from left to right, respectively, for each network node. A pair of these processes is created for each network node.

Process **Scheduler** is used to order time events and control the sequencing of activities of the entire simulation.

Figure 4 shows a block diagram showing the simulation model for the ring topology. Process **Simnode** is responsible for generating the load (data packets) on the ring. Process **Ringnode** monitors the ring traffic and implements the token-based medium access protocol. Process **Server** is used to simulate the propagational delay between each pair of LAN nodes. As in the bus model, the simulation of the ring structure uses a **Scheduler** process to control the sequencing of activities.

DISCUSSION

Token Ring vs Contention Access

A token passing ring is a LAN with a loop topology in which a token is passed around the network in a round-robin fashion from one node to the next. Contention for transmission is resolved by stipulating that only the node currently in possession of the token is allowed to transmit a frame or

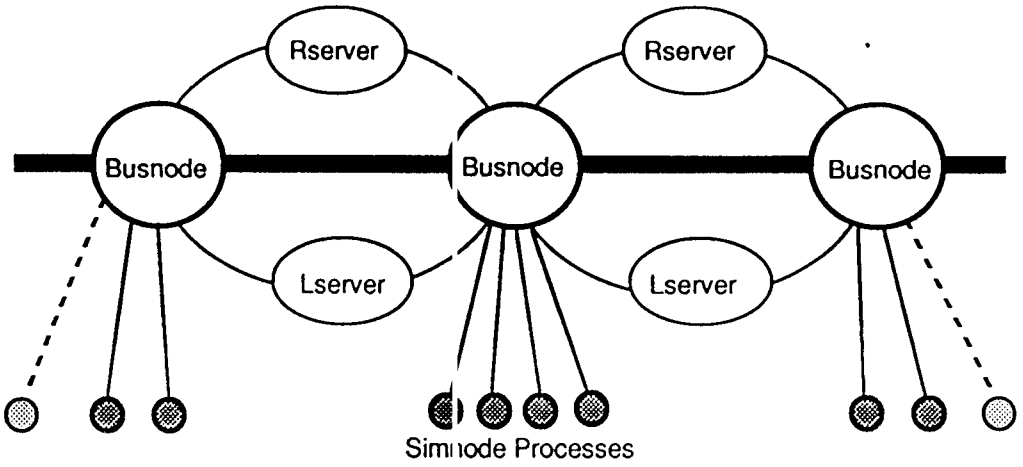


Figure 3. Simulation Model for Bus Topology Networks

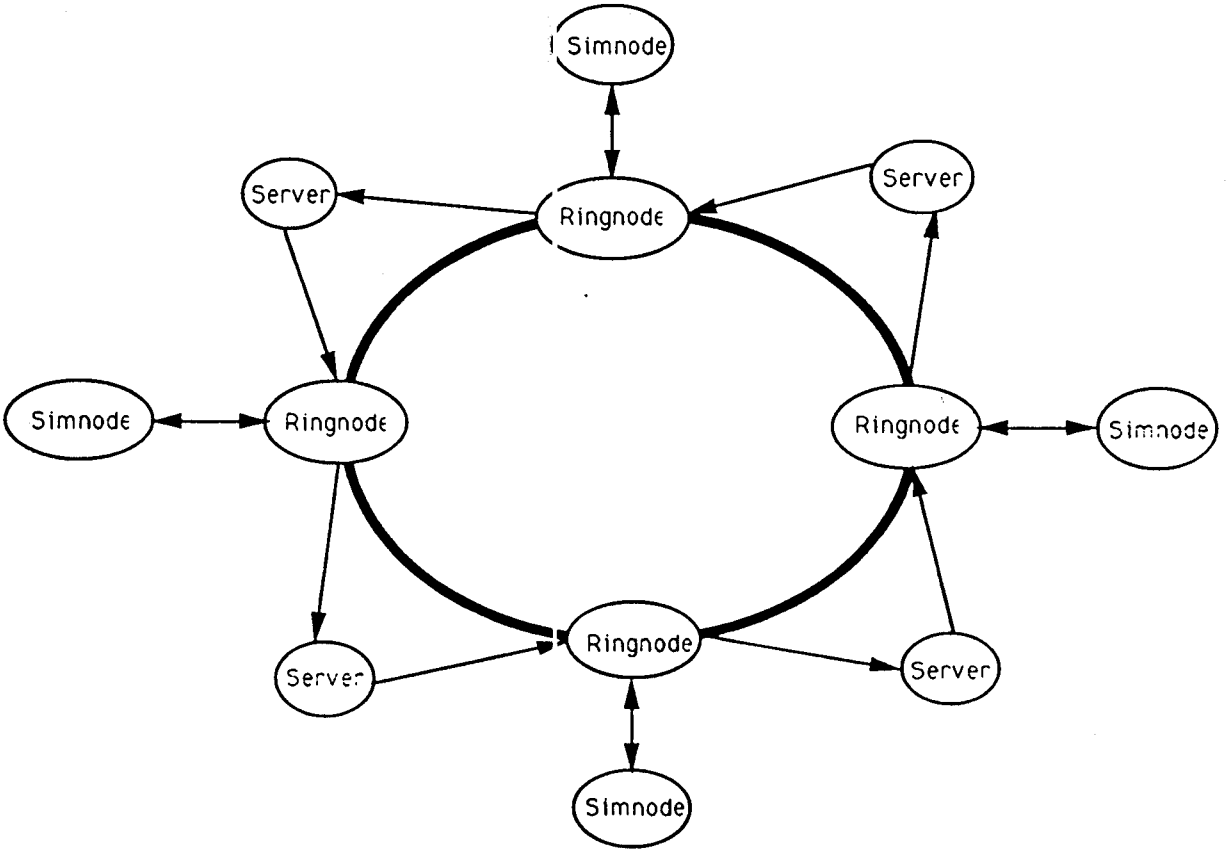


Figure 4. Simulation Model for Ring Topology Networks

a sequence of frames onto the ring. When the transmission is finished, the token is passed to the node downstream which then gets a chance to transmit. Since there is a single token on the ring, only one node can be transmitting at a time. Other (non-transmitting) nodes, however, continuously receive the bit stream, examine it and repeat it (i.e., place it on the medium to the next station). A station repeating the bit stream may copy it into local buffers or modify some control bits as appropriate.

In general, Token-Ring LAN's are much less sensitive to increased transmission rates and smaller packet sizes compared to contention protocols (e.g., ETHERNET with CSMA/CD). Since token-rings are collision free, a maximum packet delay can be guaranteed for a given number of stations. Thus, the real-time requirements of applications having high traffic loads (e.g., networks with large number of simulation training devices) can be handled more gracefully by using a contention-free ring scheme.

Because of its point-to-point connection property, rings readily accommodate the use of optical fiber as a transmission medium. In addition to offering reduced size, weight and enhanced safety features, optical fiber also offers very high signal bandwidth. One very promising implementation of ring networks using optical fiber is the Fiber Distributed Data Interface (FDDI). FDDI is a 100 Mbits/sec token-ring LAN protocol that is rapidly becoming accepted as the premier high speed LAN standard [7]. With its embedded extensibility to support even higher speeds (500 to 1,000 Mbits/sec), FDDI is poised to become the dominant high-end LAN of the 1990's. The paradigm for FDDI topology is known as a "dual counter-rotating ring of trees". The physical layer topology consists of independent, full-duplex, point-to-point physical connections, while the logical layer consists of one or two rings. The FDDI Medium Access Control (MAC) protocol provides data services similar to those of the IEEE 802.5 token-rings. An extension to FDDI (FDDI II) is currently being investigated to add isochronous data transmission capabilities to the network, thus enabling it to handle both voice and data. FDDI technology will eventually provide the simulation/training industry with powerful real-time LAN's capable of interconnecting an unprecedented number of stations.

Another promising feature of Token-Rings is that they provide a priority-based scheme for packet transmission across the network. In the ANSI/IEEE 802.5 ring implementation, the passing token has three bits indicating the current priority level of the ring (this gives a total of 8 priority levels). A station that captures the token can only transmit packets whose priority is equal to or higher than the priority of the passing token. The ANSI/IEEE 802.5 protocol also provides mechanisms that enable stations to request/change the priority of the passing token. In simulation networks, this means that it will be possible to assign levels of priority to different types of messages which may be beneficial in attempting to optimize real-time system performance, especially under peak load conditions.

On the other hand, token rings are outperformed by bus topology LAN's in certain areas. One main advantage of the bus structure over ring LAN's is the reliability of network operation following a node failure. In general, bus-based LAN's are more resistant to network crashes due to node failures since the propagation of messages on the bus does not require the participation of any given node. Failure of a station on the ring structure, however, can bring the entire LAN down. This problem has been considerably reduced by the increased reliability of today's ring chips and off-the-shelf ring attachments. Furthermore, new fiber optic ring devices use optical bypass switches in order to allow inactive (off-line) stations to pass the traveling data-carrying light waves directly from one neighboring node to the next without active power and with little or no degradation of the optical signal.

Bus-based ETHERNET LAN's have enjoyed economic advantages because of their widespread use in the past two decades. These advantages, however, are about to disappear since VLSI technology, fiber optics, and other near-term advances will soon be supplying the market with ring chips and devices at the same low cost as comparable bus products. Also, hardware support for FDDI is rapidly growing and the projected increase in development/installation investments in FDDI are expected to drive down the cost of FDDI hardware considerably.

GBRAM vs Contention Access

The GBRAM LAN protocol implementation shares the same bus topology as the ETHERNET implementation (see Fig. 1). The nodes connected onto the network via the same multi-port transceiver belong to the same group and each node within the group has a unique identity. This node identity scheme plays an important part in the assignment of channel access time slots for each node. Every node on the network perceives the channel state under the GBRAM as consisting of cycles of **scheduling** and **transmission** periods. Roughly speaking, the end of a transmission period designates the beginning of a scheduling period and the end of a scheduling period signals the beginning of the next scheduling period. During a scheduling period, every node gains the right to access the network channel starting with the node whose identity sequentially follows the node who transmitted last.

GBRAM avoids collisions by scheduling different users at unique time instances. The time interval between two successive scheduling instances depends on the physical location of the users who are allowed access to the network channel at these instances. In fact, it is equal to the propagation delay between the two users who are scheduled to transmit at these two unique instances. In the GBRAM, therefore, the physical location of each user on the network is extremely important in calculating the network's scheduling algorithm.

It has been observed that in large scale simulation networks not all users are active at all times. Consider, for example, a vehicle simulator which is active at the beginning of a battle, but is destroyed by enemy fire during the simulation. These inactive users must be taken out of the token passing sequence list

in order to reduce the number of wasted idle slots which will be scheduled for the network. Hence, there must be a procedure to sign-off users from the network. This procedure might be implemented as follows: an active user signs off by broadcasting, at its scheduled transmission time instant, a sign-off packet which would be read by all other active users who would, in turn, update their scheduling sequence accordingly.

The successful operation of the version of the GBRAM presented in this paper depend on the fact that all the users know a common time epoch. This common time epoch corresponds to the beginning of a scheduling period. In our version of GBRAM, the beginning of a scheduling period corresponds to either the end of a transmission period (as perceived by the transmitting user) plus the propagation delay along the cable or a complete scheduling cycle after the beginning of the previous scheduling period. It is obvious that the common time epoches can be determined by any user who observes the channel state at all times and knows its propagation delay from any other user in the network. Note that contention protocols require only that users observe the state of the channel at all times. There are other versions of GBRAM [3], however, that do not require the users to have a complete knowledge of the network topology. These versions of GBRAM will not perform as well as the GBRAM version considered in this paper.

NETWORK SIMULATION RESULTS

ETHERNET Simulated Performance

The network protocol simulation models described earlier have been used to gather information about the performance of local area networks used for real-time training, under various conditions of node (simulator) placement on the network, traffic load levels and packet scheduling policies. One configuration considered in our analysis is unique to the application of local area networks to the interconnection of simulation / training devices. In this configuration, an optimization is made to reduce the load on the network. An explanation of this optimization for the ETHERNET case is given below.

Upon a state change, a simulator (node) on the network sends the information concerning its new state to other nodes on the LAN. Each new state results in the generation of a new data packet at the **application layer** (i.e.,.at the node level). The packet is then submitted to the **data link layer** in order to start the process of its transmission. In ETHERNET, only one packet per node is delivered for transmission at a time. Other packets are normally queued up at the application level waiting for the end of the ongoing transmission attempt. In this context, the arrival of a new packet (carrying the most current state of the node) simply replaces the previous packet (stored at the application layer) which represents a now outdated state condition. The discarding of the outdated packet helps speed up the transmission of the most current state of the node. Notice that the packet already submitted to the ETHERNET data link layer is under the control of ETHERNET protocol (board) which is

not accessible from the application layer and, therefore, is not affected by new packet arrivals.

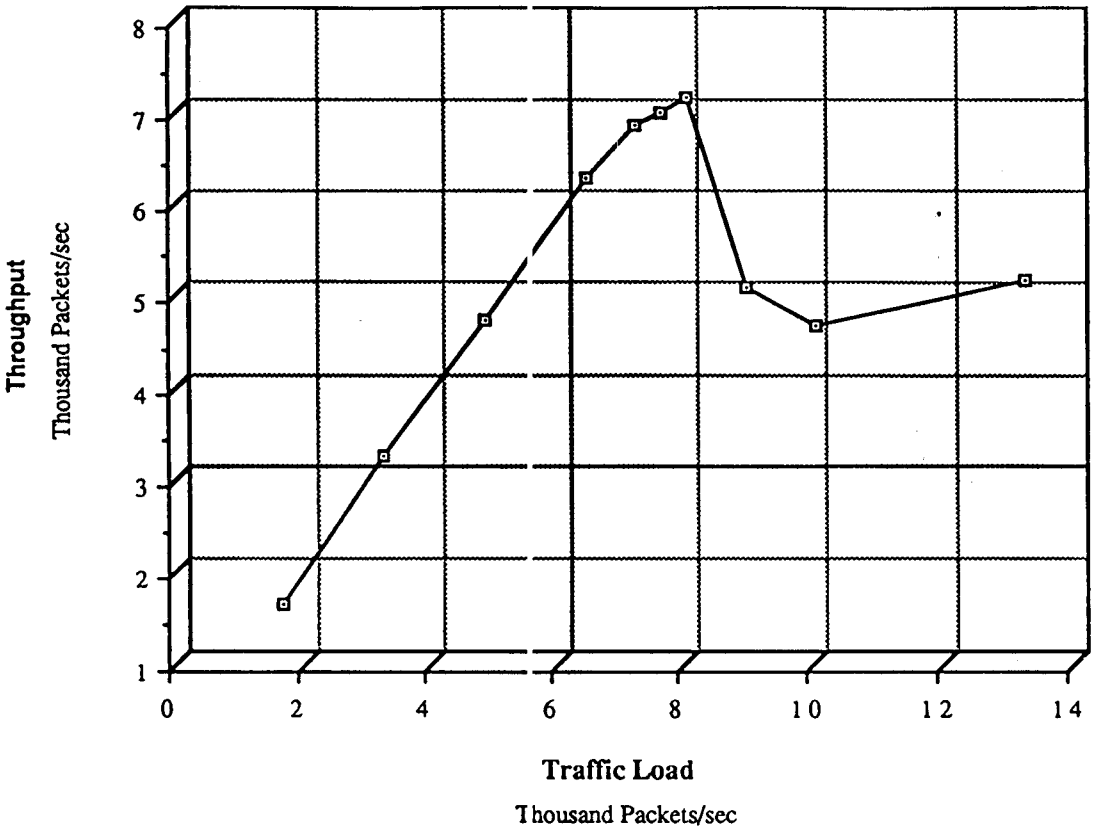
The performance of this ETHERNET configuration is given in Graphs 1 & 2, and Table 1. Graph 1 gives the relationship between the throughput of the LAN and the total initial traffic load from all simulators (i.e., before any discarding at the application level). Graph 2 gives the relationship between the total initial load and the packets discarded by ETHERNET as a result of exceeding the maximum count for transmission attempts (16) due to excessive collisions. Notice that Graph 2 gives the packets discarded by the ETHERNET protocol and not the obsolete packets discarded at the application level. Statistics about the average transmission delay and average/maximum number of transmission attempts are given in Table 1.

At low traffic load levels, the effect of collisions is small and no packets are discarded due to excessive collision counts. All packets submitted to the data link layer at such low loads eventually get transmitted successfully and the throughput of the network is equal to the total traffic load minus the obsolete packets discarded at the application level. As the traffic load increases, more collisions occur and the average number of transmission attempts per packet (and consequently the average packet delay) increases (see table 1). Since a packet is thrown away (by ETHERNET) if its transmission fails 16 consecutive times, the growing collision rate eventually results in the loss of some packets. At some point, the network becomes overwhelmed by the collision overhead and less LAN bandwidth becomes available for actual packet transmission. This is the reason for the decline in the ETHERNET LAN throughput even though the traffic load continues to increase. Since the actual throughput is dependent on both the number of obsolete packets discarded by the application layer and the colliding packets discarded by ETHERNET boards, the performance of the network may show some slight variations in actual throughput; but otherwise will stay in the saturation throughput level it has attained.

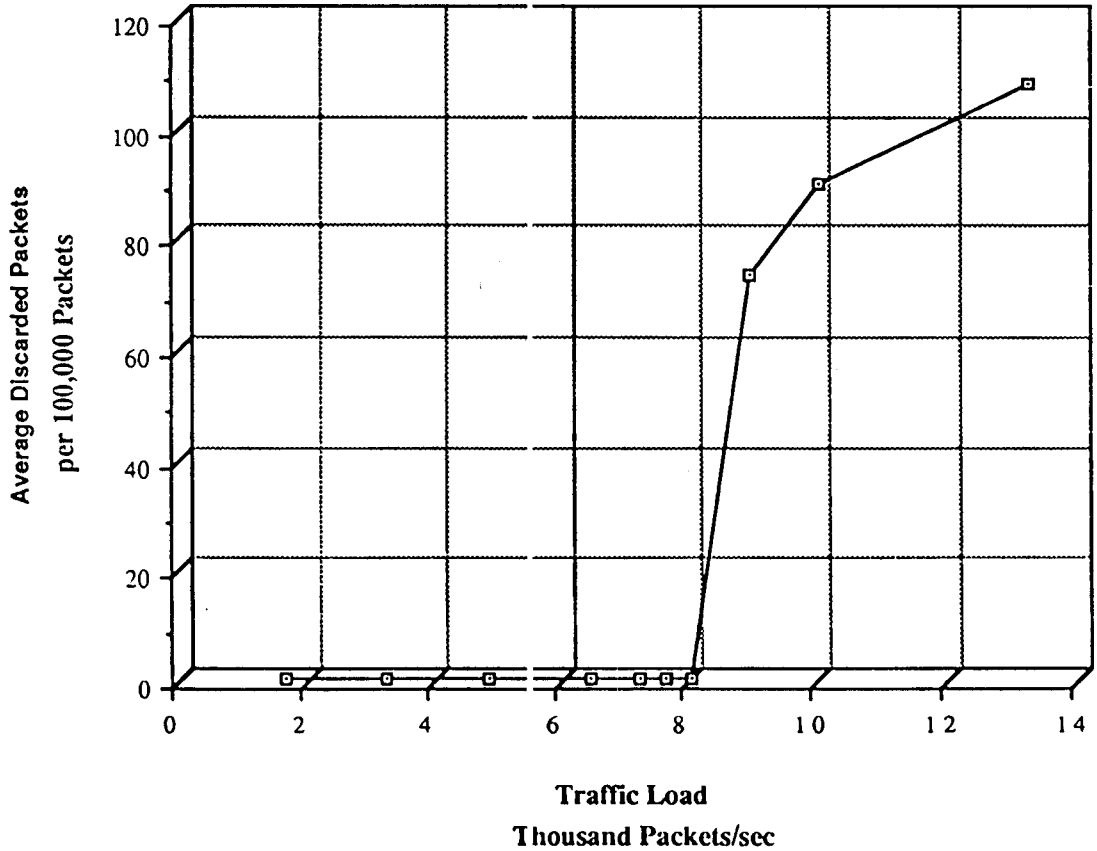
Under the above relatively high traffic condition, 0.1% of the packets submitted for transmission are totally lost (discarded after 16 unsuccessful attempts to transmit). This is a typical behavior of the CSMA/CD protocol. At low and moderate loads, CSMA/CD is quite satisfactory as evidenced by the graphs of Graphs 1 and 2. The low probability of collision under such light loads is the primary reason for the excellent performance of the CSMA/CD. As the load on the network starts to increase, more collisions between packets occur and hence more overhead is spent to resolve collisions, jam the bus after each detected collision, and reschedule packets for later retransmission. At some point, the performance of ETHERNET collapses (i.e., throughput decreases and the percentage of lost packets increases as load increases) due to the excessive overhead of managing collisions.

Token Ring Simulated Performance

Token ring LAN's exhibit quite different behavior as compared to the ETHERNET. In some versions of token ring protocols, a transmitting



Graph 1



Graph 2

Traffic Load (Thousand Packets/sec)	Average Transmissior Delay (millisec)	Average Number of Transmission Attempts per Pack	Maximum Number of Attempt
1.60	0.111	1.015	2
3.20	0.152	1.161	5
4.80	0.223	1.342	9
6.40	0.383	1.767	9
7.20	0.710	2.226	10
7.60	1.234	2.563	13
8.00	1.877	2.597	15
8.90	2.441	3.144	16
10.00	2.925	3.135	16
13.20	3.748	3.618	16

TABLE 1.

station recreates the free token and puts it onto the ring as soon as it finishes packet transmission. In IEEE 802.5 rings, however, a transmitting station checks to see if its address (affixed at the header of the transmitted packet) has returned to it (indicating a complete cycle around the ring). Only after receiving this address is the station allowed to transmit the free token onto the ring, thus allowing other stations an opportunity to transmit. This latter protocol is more conservative (from a reliability point of view) but imposes extra time overhead for token management. At low network traffic loads, the IEEE 802.5 token ring protocol causes more transmission delays for packets than the CSMA/CD counterpart. Unlike collision handling, the overhead of token management is largely independent of the LAN load. Therefore, the throughput of token-ring LAN's continues to increase as the traffic load increases. No degradation in performance at high loads is exhibited by token rings in contrast to the ETHERNET LAN.

GBRAM Simulated Performance

Preliminary GBRAM simulation results indicate that GBRAM should perform well for medium to high input traffic loads, but may be inferior to contention protocols for light to medium input traffic loads. Let us try to justify this observation based on the description of the GBRAM protocol presented in the previous section.

Consider the case where there is only one out of a total of 100 users that generates traffic onto the network and that, for the majority of its transmissions, this user has one packet in its buffer every time a scheduling period of the GBRAM protocol starts. Let us assume that the propagation delay between two users in the same group is 30 bits, the propagation delay along the cable is 20 bits, and the packet length is 1000 bits. This is a case of light input traffic. It is easy to see that GBRAM induces an average and maximum packet delay of (30x100+20)/2 + 1,000 = 2,510 bits and (30x100+20) + 1,000 = 4,020 bits, respectively. Every contention protocol under the aforementioned light input traffic conditions, induces an average and a maximum packet delay of 1000 bits (the packet length).The performance difference widens as the number of the users in the network increases.

Suppose now that all 100 users in the network are active. Each one of them has exactly one packet to transmit at the beginning of a GBRAM scheduling period. This corresponds to a case of high input traffic load. Now GBRAM induces an average and a maximum packet delay of approximately (1000x100)/2 + 1000 = 51,000 bits and (1000x100) + 1000 = 101,000 bits, respectively. The length of the packet was once again taken to be equal to 1000 bits. The aforementioned input traffic load is approximately equal to 100%. Contention protocols attain a throughput smaller than 100% even under ideal conditions (i.e. small end-to-end propagation delay/packet length ratio). As a result, contention protocols are unstable (experience unbounded packet delays) for the above high input traffic scenario.

The above discussion, although simplified, verifies our point that there will be a region of input traffic loads (light to medium) where contention protocols outperform GBRAM and a region of input traffic loads (medium to

high) where GBRAM outperforms contention protocols The cutoff point depends on the total number of users in the network and increases as the number of users in the network increases. The exact cutoff point will be determined by simulation.

CONCLUSIONS

In this paper we have described an ongoing effort to model and evaluate the performance of three different network protocol access methods suitable for networking of simulation training devices: a contention access method based on the CSMA/CD (ETHERNET) protocol, and two contention-free methods based on Virtual Token Bus Access such as GBRAM and Token-Ring Access protocols. The system models pertaining to the above three access methods were addressed and a high-level description of a detailed simulation software system implemented for evaluating the performance of these protocols was given.

The models developed for the three access methods will enable us to perform a comparison study and evaluate different design decisions. Some of the numerical performance measures that are being gathered by the models are:

- The impact of traffic loading on network throughput
- The utilization of the transmission medium
- The distribution of delay times of transmitted packets.

The models developed under this effort offer a very flexible tool for the evaluation and analysis of important classes of networking schemes that can be used to interconnect large numbers of real-time simulation training devices. Further research is being conducted which is focusing on implementing these two alternate protocols in a hardware/software test bed with the ultimate goal of enhancing the capability of simulation networks.

ACKNOWLEDGEMENTS

This work is supported by the U.S. Army Program Manager Training Devices (PM TRADE) under Broad Agency Announcement # 88-01. The authors would like to thank J. Cadiz and E. Stadler for their help in obtaining and analyzing network traffic data and preparing this report.

REFERENCES

[1] ANSI/IEEE - International Standard 8802/3 "Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specification", IEEE Computer Society Press, 1985.

[2] Metcalfe, R. M. and Boggs, D. R., "Ethernet Distributed Packet Switching for Local Computer Networks", Communication Ass. Comput. Mach., Vol. 19, no. 7, pp. 395-403, 1976.

[3] Liu, T. T., Li, L. and Franta, W. R. "A Decentralized Conflict-Free Protocol, GBRAM for Large Scale Local Networks", Computer Network Symposium Proceedings, pp. 39-54, Dec. 1981.

[4] ANSI/IEEE - International Standard 8802/5 "Token Ring Access", IEEE Computer Society Press, 1985.

[5] Dixon, R., Strole, N. and Markov, J. "A token ring network for local data communication", IBM System Journal, Vol. 22, 1983, pp.62-74.

[6] Gehani, N. and Roome, W. "Concurrent C" Technical Report, AT&T Bell Laboratories, 1986.

[7] Marrin, K. "Emerging standards, hardware and software light the way to FDDI", Computer Design, Vol 28, No. 7, April 1989, pp. 51-57.

ABOUT THE AUTHORS

M. A. Bassiouni received his Ph.D. degree in Computer Science from Pennsylvania State University in 1982. He is currently an Associate Professor of Computer Science at the University of Central Florida, Orlando. His current research interests include computer networks, distributed systems, databases, and performance evaluations. He has authored several papers and has been actively involved in research on local area networks, concurrency control, data encoding, I/O measurements and modeling, schemes of file allocation and user interfaces to relational database systems. Dr. Bassiouni is a member of the IEEE Computer Society, the Association for Computing Machinery, and the American Society for Information Science.

M. Georgiopoulos received his Ph.D degree in Electrical Engineering from the University of Connecticut in 1986. He is currently an Assistant Professor in the Department of Electrical Engineering and Communication Sciences at the University of Central Florida, Orlando. His current research interests include multi-user communication theory, communication networks, computer networks, and spread spectrum communications. Dr. Georgiopoulos is a member of IEEE and the Technical Chamber of Greece.

J. Thompson received his BS degree in Electrical Engineering from the University of Central Florida in 1978. He is currently a Research Associate for the Institute for Simulation and Training (IST) at the University of Central Florida, Orlando. Mr. Thompson has technical responsibility for all IST research activities involving computer networking.