SIMULATION NETWORKING AND PROTOCOL ALTERNATIVES

Technical Report

Publication Number IST-TR-90-20

M. Bassiouni, Michael Gerogiopoulos, Jack Thompson

The Institute for Simulation and Training
12424 Research Parkway, Orlando, FL 32826

April 26 & 27, 1989

University of Central Florida

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION: unclassified	1b RESTRICTIVE MARKINGS: None
2a SECURITY CLASSIFICATION AUTHORITY: N/A	3 DISTRIBUTION/AVAILABILITY OF REPORT: Approved for Public Release; Distribution is unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A	
4 PERFORMING ORGANIZATION REPORT NUMBER(S): IST-TR-90-20	5 MONITORING ORGANIZATION REPORT NUMBER(S): IST-TR-90-20
6a NAME OF PERFORMING ORGANIZATION: Institute for Simulation and Training	6b OFFICE SYMBOL (If applicable):
6c ADDRESS (City, State, and ZIP Code): 12424 Research Parkway, Suite 300 Orlando, Fl. 32826	7a NAME OF MONITORING ORGANIZATION: Project Manager for Training Devices
	7b ADDRESS (City, State, and ZIP Code): 12350 Research Parkway Orlando, Fl. 32826
8a NAME OF FUNDING/SPONSORING ORGANIZATION: DARPA/TTO	8b OFFICE SYMBOL (If applicable):
	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: N61339-88-G-0002
8c ADDRESS (City, State, and ZIP Code): 1400 Wilson Blvd. Arlington, VA. 22209	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification): Simulation Networking and Protocol Alternatives

12 PERSONAL AUTHOR(S): Bassiouni, M.; Georgiopoulos, Michael; Thompson, Jack

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Technical	FROM 7/88 TO 6/89	April 27, 1989	7

16 SUPPLEMENTARY NOTATION:

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Simulation Networks, Local Area Networks for Simulation, Ethernet, Token Ring, Virtual Token Bus, Network Performance

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

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 is also presented.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL: Mike Garnsey	22b TELEPHONE (Include Area Code): 407-380-4816
	22c OFFICE SYMBOL: AMC CM-TND-EN

DD Form 1473, JUN 86 — Previous editions are obsolete. — SECURITY CLASSIFICATION OF THIS PAGE

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.

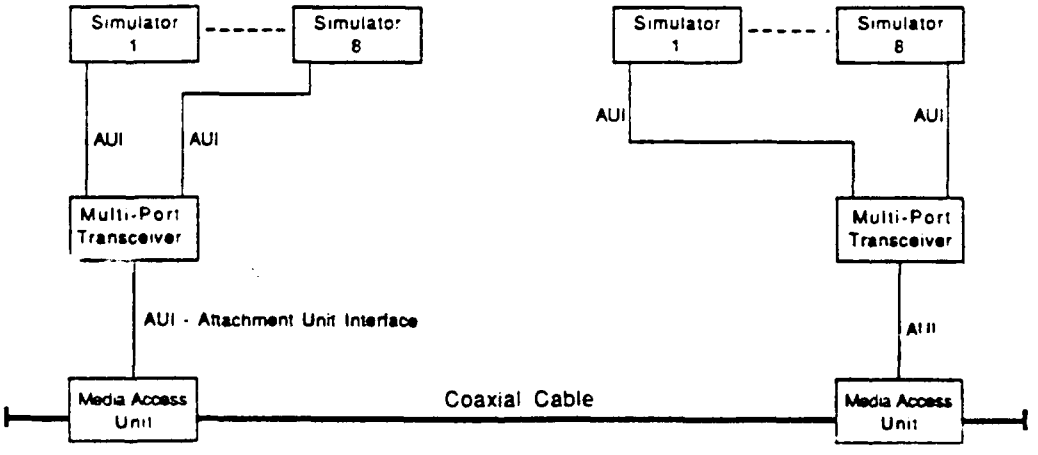

Figure 1. Bus Network Topology System Configuration

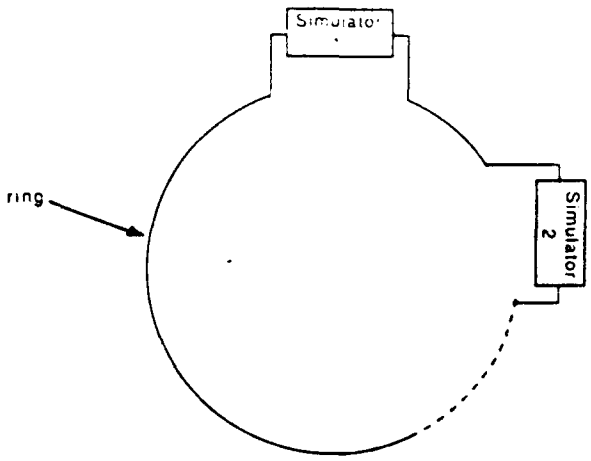

Figure 2. Ring Network Topology System Configuration

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

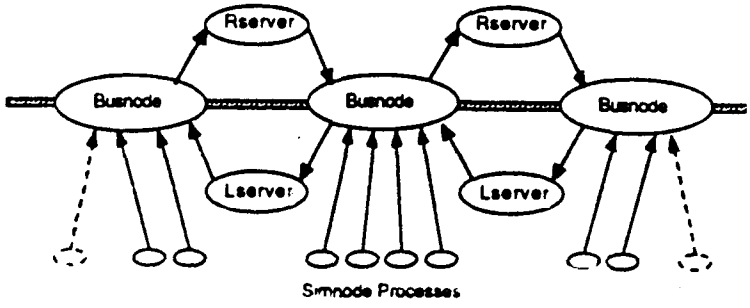

Figure 3. ETHERNET Simulation Model Process Interactions

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.