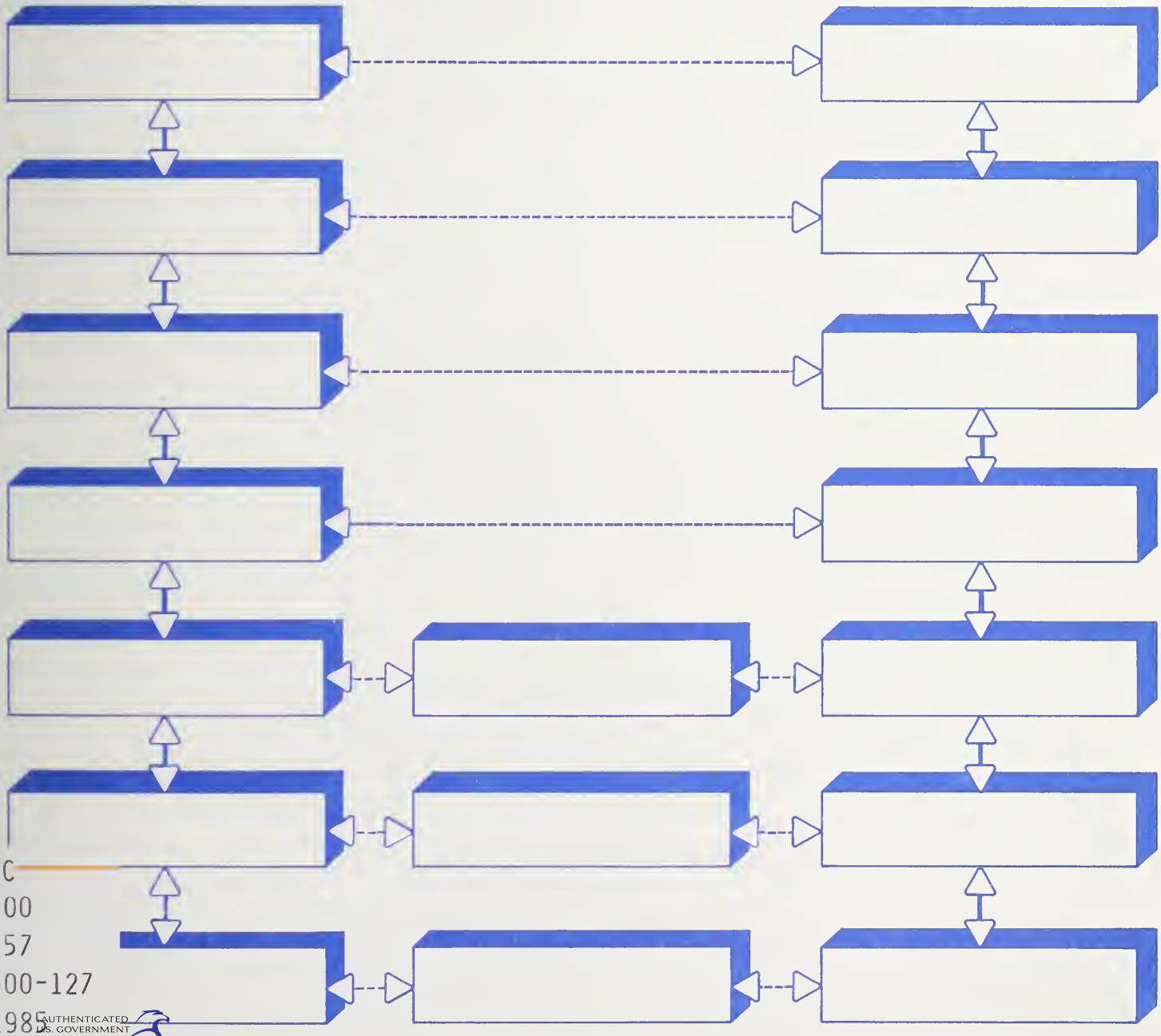
U.S. Department of Commerce

National Bureau of Standards

Computer Science and Technology

NBS Special Publication 500-127

Workshop on Analytic and Simulation Modeling of IEEE 802.4 Token Bus Local Area Networks



The National Bureau of Standards[1] was established by an act of Congress on March 3, 1901. The Bureau's overall goal is to strengthen and advance the nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, the Institute for Computer Sciences and Technology, and the Center for Materials Science.

The National Measurement Laboratory

Provides the national system of physical and chemical measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; provides advisory and research services to other Government agencies; conducts physical and chemical research; develops, produces, and distributes Standard Reference Materials; and provides calibration services. The Laboratory consists of the following centers:

- Basic Standards[2]
- Radiation Research
- Chemical Physics
- Analytical Chemistry

The National Engineering Laboratory

Provides technology and technical services to the public and private sectors to address national needs and to solve national problems; conducts research in engineering and applied science in support of these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the ultimate user. The Laboratory consists of the following centers:

- Applied Mathematics
- Electronics and Electrical Engineering[2]
- Manufacturing Engineering
- Building Technology
- Fire Research
- Chemical Engineering[2]

The Institute for Computer Sciences and Technology

Conducts research and provides scientific and technical services to aid Federal agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Institute consists of the following centers:

- Programming Science and Technology
- Computer Systems Engineering

The Center for Materials Science

Conducts research and provides measurements, data, standards, reference materials, quantitative understanding and other technical information fundamental to the processing, structure, properties and performance of materials; addresses the scientific basis for new advanced materials technologies; plans research around cross-country scientific themes such as nondestructive evaluation and phase diagram development; oversees Bureau-wide technical programs in nuclear reactor radiation research and nondestructive evaluation; and broadly disseminates generic technical information resulting from its programs. The Center consists of the following Divisions:

- Inorganic Materials
- Fracture and Deformation[3]
- Polymers
- Metallurgy
- Reactor Radiation

[1]Headquarters and Laboratories at Gaithersburg, MD, unless otherwise noted; mailing address Gaithersburg, MD 20899.
[2]Some divisions within the center are located at Boulder, CO 80303.
[3]Located at Boulder, CO, with some elements at Gaithersburg, MD.

Computer Science and Technology

NBS Special Publication 500-127

Workshop on Analytic and Simulation Modeling of IEEE 802.4 Token Bus Local Area Networks

Held at
National Bureau of Standards
Gaithersburg, Maryland
April 29-30, 1985

Robert Rosenthal, Editor

Systems and Network Architecture Division
Center for Computer Systems Engineering
Institute for Computer Sciences and Technology
National Bureau of Standards
Gaithersburg, MD 20899

U.S. DEPARTMENT OF COMMERCE
Malcolm Baldrige, Secretary

National Bureau of Standards
Ernest Ambler, Director

Issued June 1985

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer utilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in this series should complete and return the form at the end of this publication.

Library of Congress Catalog Card Number: 85-600556

National Bureau of Standards Special Publication 500-127
Natl. Bur. Stand. (U.S.), Spec. Publ. 500-127, 268 pages (June 1985)
CODEN: XNBSAV

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1985

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402

TABLE OF CONTENTS

Acknowledgements

A workshop is only as useful at the participants make it.
Thanks are due to the people who came to work.

Allen-Bradley
Anatoly Moldovansky
747 Alpha Dr.
Highland Hts., OH 44143
(216) 449-6700

Atomic Energy of Canada
Tony Capel
275 Slater St.
Ottawa K1A1E5
Canada
(613) 236-6444

Autotote LTD
Fred Sylvanus
100 Belleuue Rd.
Newark, DE 19711
(302) 737-4300

Bell Communications Research
George Change
6 Corporate Pl.
Piscataway, NJ 08854
(201) 981-3879

Boeing Computer Services
Ed Nugent
M/S 9C-23
P.O. Box 24346
Seattle, WA 98124
(206) 575-5444

Comm Power
Stephen Dunford
26560 Agoura Rd. #101
Calabasas, CA 91302
(818) 880-5511

Contel Information Systems
Gerald Boxer
11781 Lee Jackson Highway
Fairfax, VA 22033
(703) 385-4300 ext. 122

Contel Information Systems
Peter Malpass
11781 Lee Jackson Highway
Fairfax, VA 22033
(703) 385-4300 ext. 124

Digital Equipment Corp.
Tom McGowan
146 Main St.
ML05-2/E50
Maynard, MA 01754-2571
(617) 493-2648

Foxboro Corp.
Ed Shaughnessy
Dept. 351
38 Neponset St.
Foxboro, MA 02035

General Electric CR&D
M.E. Ulug
Bldg. KWC Room 308B
P.O. Box 8
Schenectdy, NY 12301

General Motors Corp.
APMES
Kester Fong
Manufacturing Bldg. A/MD-39
30300 Mound Rd.
Warren, MI 48090-9040
(313) 492-1586

General Motors Corp.
General Motors Tech. Center
A/MD-39
David Greenstein
30300 Mound Rd.
Warren, MI 48090-9040
(313) 492-1581

General Motors Corp.
APMES A/MD-39
Gary Workman
12 Mile & Mound Rd.
Warren, MI 48090
(313) 575-0632

GMI
Juan R. Pimentel
1700 W. Third Ave.
Funt, MI 48502
(313) 762-7992

Hewlett Packard
Glenn Talbott
800 Foothills Blvd.
Roseville, CA 95678
(916) 786-8000

Industrial Networking Inc./Ungermann-Bass Inc.
Jade Chien
3990 Freedom Circle
Santa Clara, CA 95050
(408) 496-0969

Industrial Technology Institute
John F. Meyer
P.O. Box 1485
Ann Arbor, MI 48106
(313) 763-0579

Industrial Technology Institute
K.H. Muralidhar
P.O. Box 1485
Ann Arbor, MI 48106
(313) 763-0579

Litton Amecom
Chuck Dawson
5115 Calvert Rd.
College Park, MD 20740
(301) 454-9865

Litton Amecom
Tim McEllibot
5115 Calvert Rd.
College Park, MD 20740
(301) 454-9705

Litton Amecom
Fred Kalmus
5115 Calvert Rd.
College Park, MD 20740
(301) 454-9340

Motorola Semiconductor
Product Sector
Bruce Loyer
3102 N. 56th St.
Phoenix, AZ 85018
(602) 952-3454

Motorola Semiconductor Israel Ltd.
Orly Kremien
147 Blalik St.
Ramat-Gan 52523
Israel
(03) 7538258

National Bureau of Standards
Jeff Bader
Bldg.225/B217
Gaithersburg, MD 20899
(301) 921-3516

National Bureau of Standards
Charles Hartmann
Bldg. 225/B217
Gaithersburg, MD 20899
(301) 921-3516

National Bureau of Standards
Karen Hsing
Bldg. 225/B217
Gaithersburg, MD 20899
(301) 921-3516

National Bureau of Standards
Anastase Nakassis
Bldg. 225/B217
Gaithersburg. MD 20899
(301) 921-3516

National Brueau of Standards
Steve Ritzman
Bldg. 225/B217
Gaithersburg, MD 20899
(301) 921-2601

National Bureau of Standards
Robert Rosenthal
Bldg. 225/B217
Gaithersburg, MD 20899
(301) 921-3516

National Bureau of Standards
Curtis Royster
Bldg. 225/B217
Gaithersburg, MD 20899
(301) 921-3516

National Bureau of Standards
Dan Stokesberry
Bldg. 225/B217
Gaithersburg, MD 20899
(301) 921-3516

National Bureau of Standards
Robert Toense
Bldg. 225/B217
Gaithersurg, MD 20899
(301) 921-3516

Siemens AG
Dr. Detler Leisengang
Balanstr 73
8 Munich 80
West Germany
49-89-4144 2493

Syscon Corp.
Joseph P. Brazy
1000 Thomas Jefferson St., N.W.
Washington, DC 20007
(202) 342-4000

Syscon Corp.
Charles Croft
1000 Thomas Jefferson St., N.W.
Suite 300
Washington, DC 20007
(202) 342-4546

Sytek Corp.
Joseph Rickert, Jr.
6500 Rock Spring Dr.
Suite 100
Bethesda, MD 20817
(301) 530-5100

Tektronix Corp.
Andy Luque
131 N.E. McCartney
Bend, OR 97771
(503) 923-4423

Western Digital
Heinz Jauch
2445 McCabe Way
Irvine, CA 92714
(714) 863-0102

Western Digital
Kelly McClellan
2445 McCabe Way
Irvien, CA 92714
(714) 863-0102 ext. 341

Special thanks are due to Ms. Mary Lou Fahey whose secretarial talents and local arrangement expertise made the workshop possible.

Workshop Chairman - Mr. George Change, Bell Communications Research

Program Chairman - A. K. A. Sastry, Rockwell International

NBS Liaison - Robert Rosenthal, National Bureau of Standards

Keynote:

Robert Rosenthal
National Bureau of Standards

Robert Rosenthal,
Manager Local Area Networking
Institute for Computer Sciences and Technology
National Bureau of Standards
Gaithersburg, MD 20899

ANALYTIC AND SIMULATION MODELING OF IEEE 802.4 TOKEN BUS

Token bus technology is anticipated for use by national and international organizations seeking standard local area network solutions for factory, laboratory and process control automation applications. Several token passing technologies have been described; but, only one emerging token bus standard, the IEEE 802.4 Token Bus, currently includes broadband facilities.

The demand for highly reliable token bus networks that support prioritized and deterministic access with robust error detection and recovery has sparked enormous interest in the Federal Government and in the private sector. The National Bureau of Standards has joined forces with researchers from industry and academia to study the behavior and characteristics of token bus technology.

Our approach is to organize a consortium of researchers including guest workers, faculty appointments, co-op students and staff members that regularly meet to develop experimental plans and laboratory testbed facilities, to conduct experiments, to collect and analyze data and to organize and sponsor workshops that report experimental findings. One of our research objectives is to develop predictive analysis techniques for IEEE 802.4 networks. Analytic and simulation modeling of these networks help to build competence and confidence in this new technology.

Our Workshop goals are to encourage modeling of the IEEE 802.4 specifications. Revision F, an ISO Draft International Standard or DIS, has several documented problems; so, Revision G is being planned. Many of these documented problems have been discovered using the models described in these proceedings. These proceedings also make available additional information about the behavior and performance characteristics of Token Bus. Our hope is that these proceedings will impact Revision G of the IEEE 802.4 Standard as it makes its way to International Standard status in ISO.

Our work in predicitive analysis of IEEE 802.4 Token Bus technology is just beginning. Researchers are now planning experiments for the NBS Token Bus Test Bed Facility. Empirical data that is acquired, reduced, and analyzed at NBS will be compared with predictions from these analytic and simulation modeling tools. Use of these tools will lead to a better understanding of token bus behavior, optimized parameter and timer settings for large and small network configurations and increased performance in factory, laboratory and process control environments.

Session I:

Simulation Approaches

Chairperson: Dan Stokesberry
National Bureau of Standards

PERFORMANCE SIMULATION OF THE IEEE TOKEN BUS PROTOCOL USING SIMAN

Juan R. Pimentel
Industrial Technology Institute
and
GMI Engineering & Management Institute

ABSTRACT

The SIMAN simulation language is used to simulate the performance of the physical and data link layers of a local area network suitable for manufacturing. The protocol standards specified by the IEEE project 802.4 has been chosen for the study. A detailed network queueing model is developed and implemented using the process view approach provided by SIMAN. Simulation results are shown in terms of average number of frames awaiting transmission, average response time, and medium utilization versus traffic intensity.

INTRODUCTION

The IEEE (token bus) protocols are computer communications protocols suitable for industrial local area networks (LAN). The token bus protocols are being developed by the IEEE projects 802.4 and 802.2, and are having a large impact for manufacturing and process control networks. According to the ISO nomenclature, the IEEE standards correspond to the lower two layers of the OSI reference model (i.e. physical and data link layers). General Motors (GM) has chosen the token bus protocol as part of a set of

specifications for a network to support manufacturing applications, referred to as the "Manufacturing Automation Protocol" (MAP). The success of MAP is evident from the interest of the GM divisions to use MAP, the interest of the computer vendor community to provide products compatible with MAP, and the interest of other companies to adopt MAP as a standard.

Local area networks for manufacturing applications are relatively new. From a network user's viewpoint, the most important problems in designing and installing industrial local area networks are: network topology design, development of application programs, and network performance evaluation. The network topology design problem consists in specifying the layout of all network components (e.g. stations, splitters, headends, amplifiers, gateways, bridges) for a specific installation. Techniques for network topology design usually minimize network costs satisfying constraints on throughput, response time, network expansion and others. Once a network is in place and operational, programs must be developed to support their intended application. For example, a program that does production scheduling on several manufacturing cells must utilize the network to obtain information from and send scheduling policies to the various cells that integrate the manufacturing facility. Another issue of interest to a network user is the actual operating characteristics (i.e. performance) of the network. Because of cost considerations it is desirable to estimate network performance before the network is installed or before a proposed change is implemented.

This paper concentrates on the performance evaluation of the IEEE 802.4 and 802.2 protocols using simulation models. The simulation language to be used is SIMAN which is an application language originally designed for the

simulation of manufacturing systems. SIMAN provides three approaches for simulation: process, discrete event, and a mixed process - discrete event. The network simulation presented in this paper is based on the process approach.

Several performance simulation studies have been made for token bus protocols. Rahimi and Jelatis (Rahimi, 1983) developed a detailed model intended for protocol verification. They simulated the state transition diagrams that characterize the protocols. However, they did not present performance results useful to a network's user. Their model was implemented using SIMPAS which is a set of PASCAL routines designed to support discrete event simulation (Raymond, 1981). Chen (Chen, 1984) presented a discrete event simulation and concentrated on performance issues such as fairness and stability versus traffic intensity. The model presented considered only one queue at the MAC sublayer and the model implementation was done in PASCAL.. Dahmen et. al. (Dahmen et al, 1984) compared the performance of token bus and CSMA/CD type of access mechanisms for a "session oriented" traffic. Their model also considered only one queue. The model implementation was done using FORCASD which is a FORTRAN based simulation package that supports simulation constructs derived from Petri-Nets, with results shown in terms of response times versus traffic intensity. Sweeton (Sweeton, 1983) also developed a comparison of token bus and CSMA/CD protocols in terms of worst case percent of load carried and worst case transaction time versus traffic intensity. The model used a simplified transport protocol and two priority queues for the MAC sublayer. Archambault (Archambault, 1984) developed a model that includes four queues. He studied the effects of packet length, target rotation time and number of stations on network utilization, rotation time, waiting time, and

queue lengths. Our approach uses an application language (SIMAN) instead of a general purpose language such as PASCAL as the model implementation tool.

SIMAN

We use the "process orientation" approach provided by SIMAN. The process orientation is based on constructing a model by depicting the functional operations of the systems as a block diagram (Pedgen, 1982). The block diagram is a linear top down sequence of blocks which represent specific process functions such as time delays and queues. Objects within the system that engage in activities are called **entities**. For our model, the data frames (i.e. the protocol data units at the LLC sublayer) are treated as entities. Two other terms useful in a SIMAN simulation are variables and attributes. Variables are the characteristics of the system which are global in nature and are not related to a specific entity. The number of frames waiting at the various queues is an example of a SIMAN variable. Attributes refer to characteristics of a specific entity. For example, the frame length is an attribute of a frame entity.

SIMAN is implemented in FORTRAN for transportability reasons. Hence, the SIMAN interface to user written routines is very simple. The interface is helpful for the simulation of events or blocks that SIMAN cannot support directly but could be easily coded separately as a FORTRAN routine. The advantage provided by SIMAN is that all simulation related activities (such as keeping track of simulated time) are handled automatically. Thus the analyst can concentrate on constructing a simulation model as opposed to designing a program that implements this model.

SIMAN does not provide all the flexibility required to handle the simulation of a complex system such as the IEEE protocols. There are two alternatives for the solution of this problem: a) to use a combined process and discrete event approach and b) to augment the process approach with user written FORTRAN routines.

The process approach of SIMAN handles events effectively. However, it lacks flexibility to handle complex mechanisms encountered in a token bus model such as the selection of queues and the delay contributed by many different sources. Consequently, the second alternative was chosen. Complex queue selection rules are implemented using the FORTRAN UR function which returns an integer indicating the queue number selected. The SIMAN statement QPICK:UR:Q1,Q2,...; selects the URth queue in the indicated order. Involved delay calculations are best implemented using the UF function. The statement DELAY:UF; delays entities arriving to this block by the quantity DF.

SIMULATION METHODOLOGY

Performance simulation techniques in computer networks are useful for protocol design, protocol verification, and network design. Protocol design consists in developing specific algorithms (e.g. protocols) for the information exchange at the same layer of two communicating entities. By simulating the performance of the algorithm one can suggest more appropiate protocols for the particular network. Protocol verification consists in exercising a specific protocol implementation to insure that it conforms according to the specification. Network design consists in specifying the layout of all network components (e.g. stations, splitters, taps, headends,

amplifiers, gateways, and bridges) and all other network parameters (speed, timers, buffers, and priorities) for a specific application. The corresponding model for network design must be concerned with network user requirements. Simulation models for protocol design and protocol verification are more detailed than models for network topology design. This is because protocol design and protocol verification simulation models need to capture detailed mechanisms and interactions (e.g. state transition diagrams) that characterize the protocol.

Whereas models for protocol design and protocol verification take an internal view of the protocols, models for network design take an external view of the protocol. Network design models need only to capture those mechanisms and interactions that are meaninful to a network user (e.g. response time). Thus models for network design are less detailed and consider a different set of variables and performance measures than models for protocol design and verification. However, models for network design must consider all OSI layers as well as the application process. In contrast, models for protocol design and verification typically consider only the specific layer for the intended protocol. The above discussion suggests that simulation models for network design tend to be a large conglomerate of somewhat simple and small modules. The performance simulation model presented next is useful for network design. SIMAN is an effective simulation language for network design applications.

THE SIMULATION MODEL

A queueing model is a mathematical model used to represent a physical system in which there is contention for resources. The evolution of the

system over time is done by using probability functions. To fully describe a queueing model it is necessary to describe the following [LAVE83]:

i) Customer arrival

ii) Service demand for each customer

iii) Service rate of each server

iv) Sequence of service centers visited by each customer

v) Order in which customers in each service center are served

When modeling the lower two layers of the IEEE 802 standard, customers are the frames to be transmitted over the communication channel. The server is one channel of a broadband coaxial cable. A service center is composed of one queue and the shared resource. The access control machine (ACM) determines which frame should be transmitted next. The customer arrival is characterized by a Poisson process with arrival rate λ. The service demand is the amount of service required by a frame at the MAC - physical layer interface given in bits (i.e. frame length). The server rate ν is the speed at which the physical layer can send data into the medium (i.e. data rate). The **service time** per frame is defined as the ratio of the expected value of the service demand and the service rate.

$$\text{Service time} = E[S]/\nu$$

The sequence of service centers visited is unique and simple since there is only one server. The sequence would be different if other communication model layers are taken into account in the simulation model because they would introduce other servers. (e.g. the session layer may require a service to be provided by a CPU which acts as another server).

The order in which the frames are served is detailed in the IEEE 802.4 specification (IEEE, 1984). The following is a summary of the algorithm used to determine which frame will be selected next for service. Additional details can be found in the above reference. Each station maintains four queues at the medium access control (MAC) sublayer. Let q(i), q(i-1), q(i-2), q(i-3) represent the four queues, where $4 \leq i \leq N$ and N is the total number of queues in the network. There is a priority for the order of service of the queues with q(i) having the highest priority, and q(i-3) having the least priority. Associated with each queue there are four timers THT, TRT4, TRT2, TRT0 for q(i), q(1-1), q(i-2), q(i-3) respectively. THT is the high priority token hold timer, TRT4, TRT2, and TRT0 are the target rotation timers for access classes 4, 2, and 0 respectively. In addition, a variable TCT which represents the last token circulation time is also needed for the selection algorithm. The timers are used to select one queue with a priority scheme described next. First, q(i) is examined and allowed to send frames for as long as THT. The next queue to be examined is q(i-1) which could send messages as long as TRT4 is greater than TCT. Likewise, q(i-2) is examined next and could send frames as long as TRT2 is greater than TCT. Finally, q(i-3) is examined and allowed to send messages as long as TRT0 is greater than TCT. According to the IEEE specification the token is passed from station to station. Our model assumes that the token is passed from queue to queue which is compatible with the protocol specification.

The model assumptions are detailed below.

Physical layer

- The network layout corresponds to a broadband, single cable system for a facility with dimensions less than 1000 feet (see Fig. 1).

- Data rate = 10 Mbps.

- cable propagation delay = L/(0.75c); L is the cable length and c is the speed of light.

- Head end delay = 7 μsec.

- Transmitter modem delay = 0.5 μsec.

- Receiver modem delay = 3 μsec.

MAC sublayer

- Four queues for four classes of service are provided.

- The length of the preamble is 32 bits.

- Source and destination addresses have a length of six bytes.

- It is assumed that the network operates with no token maintenance functions such as addition of stations to the logical ring, deletion of stations, multiple tokens, rejected tokens, failed station or receiver, and lost tokens.

LLC sublayer

- This sublayer provides class I type operation only. Class I does not require acknowledments nor flow control or error recovery. In addition, it is assumed that the only command used is UI (unnumbered information).

- Frames arrive to the LLC sublayer following a Poisson process.

The nomenclature introduced by Sauer et. al. (Sauer et. al, 1984) is used to represent the model. Figure 2 depicts the queueing model suitable for simulating the token transfer mechanism as described above. The switch time Tn is defined in figure 3 and corresponds to the time from which the token is received by any station until the token is received by the next station in the logical ring.

The terms described next are with reference to figure 3. Queueing delay is the elapsed time from when the frame arrives to when the token is received by the present station. Station delay is the elapsed time from when the token is received by the present station to when the station starts sending the message. Ttp is the transmission path delay and depends solely on the distance between a source and a destination station (going through the headend). Tmess is the time taken to send a frame and depends only on the total number of bits in the frame (data plus overhead). Ttf is the token frame delay which is equal to Tmess when only control (overhead) bits are transmitted. Trw is the delay involved in waiting for the duration of a response window. The definitions of the other terms are self explanatory from figure 3. Stations do not send solicit successor frames every time

they get the token. The parameter intersolicit_count determines how often a station shoud send solicit successor frames. The model considers this by normalizing the elapsed time from when a solicit successor frame is sent to when the token is received by the next station.

The model in figure 2 has been implemented in SIMAN. The process view for simulation provided by SIMAN treats an entity (frame) as it goes through the following stages: frame generation, queueing, token seizing, delay, and frame release.

Frame arrivals are implemented by the SIMAN statement CREATE:EX(1,2):MARK(20);. This statement creates entities with interarrival times given by an exponential distribution whose parameters are in parameter set number 1, with stream number 2, and assigns attribute number 20 as the arrival time of the entity. The arrived frames wait on queues for the token with the statement QUEUE,1; which queues frames arriving to the statement block in file number 1. Frames can attempt to access the token by means of the statement SEIZE:TOKEN;. If the resource is busy (i.e. another station has the token) the frame is forced to wait in its corresponding queue. The service rate of the server is coded in a user function UF as DELAY:UF;. The function UF calculates the delay of a frame from the time the token is received by the highest priority queue of a station until the token is received by the highest priority queue of the next station in the logical ring. The delay UF is a function of the message length, data rate, and the various delay introduced by the different hardware elements of the network (e.g. modem and headend delays). The statement OPICK:UR:Q1,Q2,Q3; implements the order in which the different queues are served. UR is a user rule coded as a separate FORTRAN function. Allowing the user to define its own rule is advantageous since SIMAN does not have the capability of having a complicated rule for serving customers as the one specified by the 802.4 protocol. The function UR contains all the logic necessary to handle the priority mechanisms for the four queues discussed earlier. The statement

RELEASE:TOKEN; frees the server. The statement TALLY:A(1),INT(20):DISPOSE; records statistics for variable INT in file 20, corresponding to frames with attribute A(1). The block modifier DISPOSE causes the departing frames to be destroyed.

Channel utilization is obtained using the DSTAT element of the experimental frame. For example, DSTAT:1,NR(2),UTIL; causes some statistics of the number of busy units of resource 2 be recorded using the label UTIL. The FILTER element of SIMAN is used to generate batches for output analysis. Finally, the INTERVALS element is used to generate confidence intervals on the observations generated by FILTER.

Some statistics (average, standard deviation, minimum and maximum values) are readily available from the SIMAN program for the following variables: number of frames awaiting transmission per queue, the response time per queue (see figure 3) and channel utilization. The network performance is shown as a series of graphs of the average number of frames awaiting transmission, the average response time and channel utilization versus traffic intensity. Traffic intensity is given by:

$$\rho = \lambda \frac{S}{\nu}$$

where, λ is the overall network frame interarrival rate, S is the message length and ν is the medium signalling data rate.

RESULTS

Several experiments were conducted using the model reported here. The method of batch means available in the SIMAN output processor is used to obtain 95% confidence intervals. Typically, the number of batches truncated at the beginning of the simulation varies from 1 to 6 and the number of batches kept varies from 5 to 10. Typically, a batch contains results for 6 seconds of simulated time. For all experiments, frames arrive according to a Poisson process. Even though the simulator can be configured easily to include a variable number of stations, all results presented here corresponds to four stations. Station delay is assumed to be 100 microseconds.

Fig. 4 depicts the response time for the different access classes. The number in parenthesis are the target rotation timer values for the different access classes in units of 10 microseconds (i.e. a value of 500 represents 5 milliseconds). Timer values shown correspond to typical values if one wishes to use the access classes in a manner that decreases performance as one utilizes lower priority classes for message transmission. Fig. 5 shows the effect of increasing the timer value for access class 4 to 2000. The effect is that as one increases the timer value for access 4, while maintaining the other timer values constant, access class 4 behaves as the highest priority access class (class 6). This effect is also true for priority classes 2 and 0. The effect of 3 sets of timers (see table I) on response time is depicted in Fig. 6. Response time for class 6 increases in a similar manner if any of the timers for classes 4,2, or 0 increase while the other timer values are held constant.

Channel utilization and information utilization (data bits only) are shown in Figs. 7 and 8 for timer set 1. As depicted in Fig. 9, information utilization is relatively insensitive to changes in timer values for access classes 4,2, or 0. Queue lengths shown in Figs. 10 and 11 correlate with results for response times shown in Figs. 4 and 5 for the same parameter values. Sensitivity of changes of timer values on queue length is shown on Figs. 12 and 13.

ACKNOWLEDEMENTS

The author wishes to thank P. Dougherty for his efforts in programming and running the experiments and to K.H. Muralidhar for helpful discussions.

TABLE I

Timer rotation values in units of 10 microseconds

	THT	TRT4	TRT2	TRT0
Timer set 1	500	600	700	800
Timer set 2	500	2000	700	800
Timer set 3	500	600	700	2000

REFERENCES

Bryant, R. M., "SIMPAS 5.0 User Manual," University of Wisconsin-Madison, Computer Sciences Technical Report No. 456, Nov. 1981.

Chien J. Y., "Performance analysis of the 802.4 token bus media access control protocol,", Factory Floor Communications Workshop, GM technical center, sept. 1984.

Dahmen N., Killat U., and Stecher R., "Performance analysis of token bus and CSMA/CD protocols derived from FORCASD simulation runs," Proc. of the 2nd. Int. Symp. on the performance of computer communication systems, Zurich, March, 1984.

IEEE PROJECT 802.4, "Token-Passing Bus Access Method and Physical Layer Specifications," Draft F, July 1984.

Lavenberg, S.S., Editor, COMPUTER PERFORMANCE MODELING HANDBOOK, Academic Press, New York, 1983.

MAP task force, "Manufacturing Automation Protocol," General Motors technical center, Warren, Michigan, April, 1984.

Pedgen C. D., "INTRODUCTION TO SIMAN," Systems Modeling Corporation, State College, Pennsylvania, 1982.

Rahimi S. K., and Jelatis G. D., "Simulation Modeling of IEEE 802 token bus medium access control protocol," Proc. of the workshop on performance and evaluation of local area networks, pp. 127-154, march, 1983.

Sauer C. H., MacNair E. A., and Kurose J. F., "Queueing Network Simulations of Computer Communication," IEEE Journal on selected areas in

communications, Vol. 2, pp. 203-219, Jan. 1984.

Sweeton D. C., "Simulation results for factory floor networks," Eleventh International Purdue Workshop on industrial computer systems, Lafayette, Indiana, Oct. 1983.

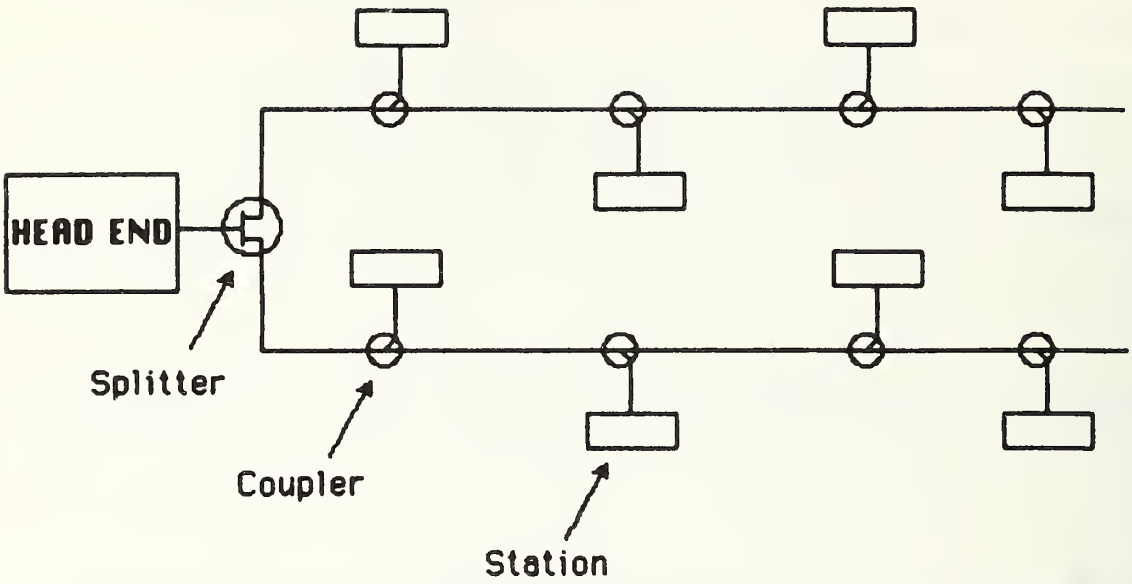


Figure 1. Network layout for a small broadband single cable system.

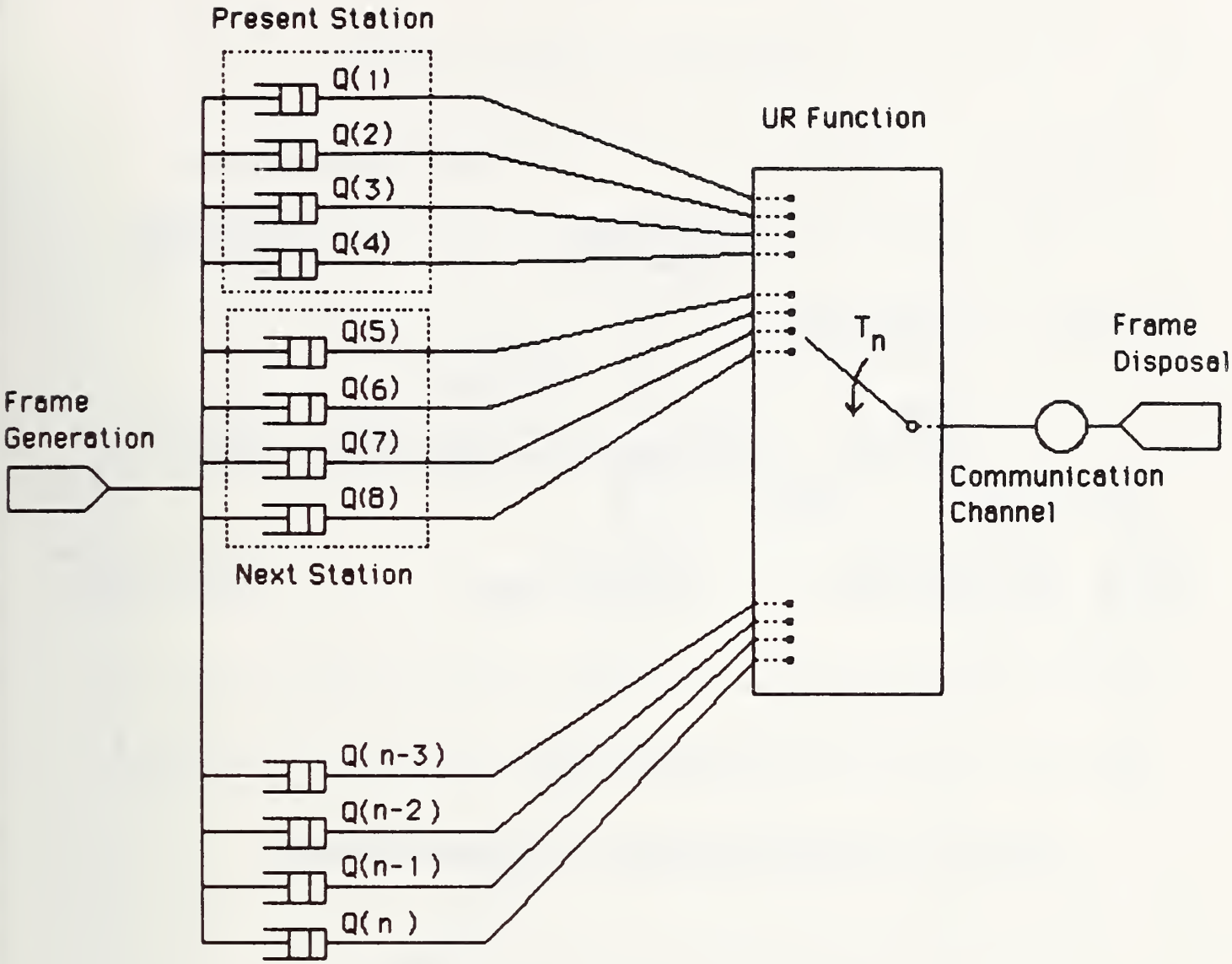


Figure 2. Queueing model for the physical and data link layers of the token bus protocol.

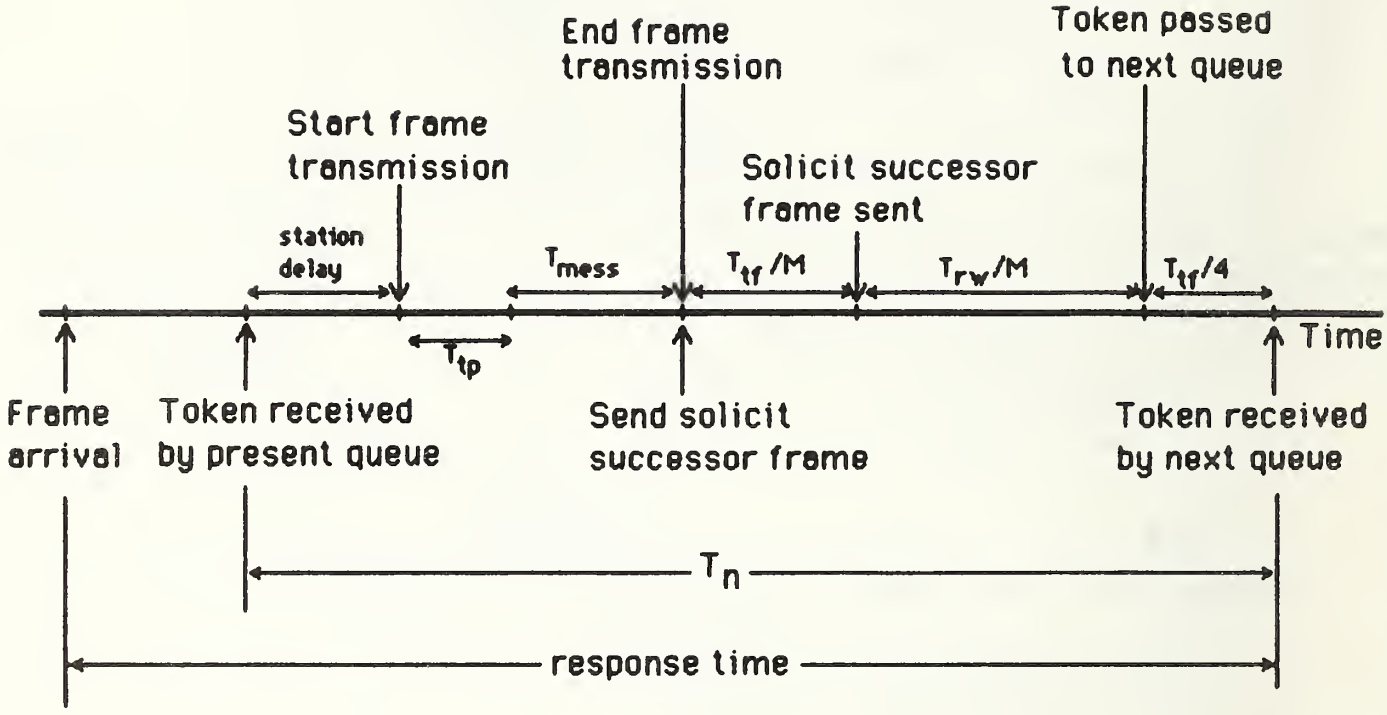


Figure 3. Timing relationships for the queueing model.

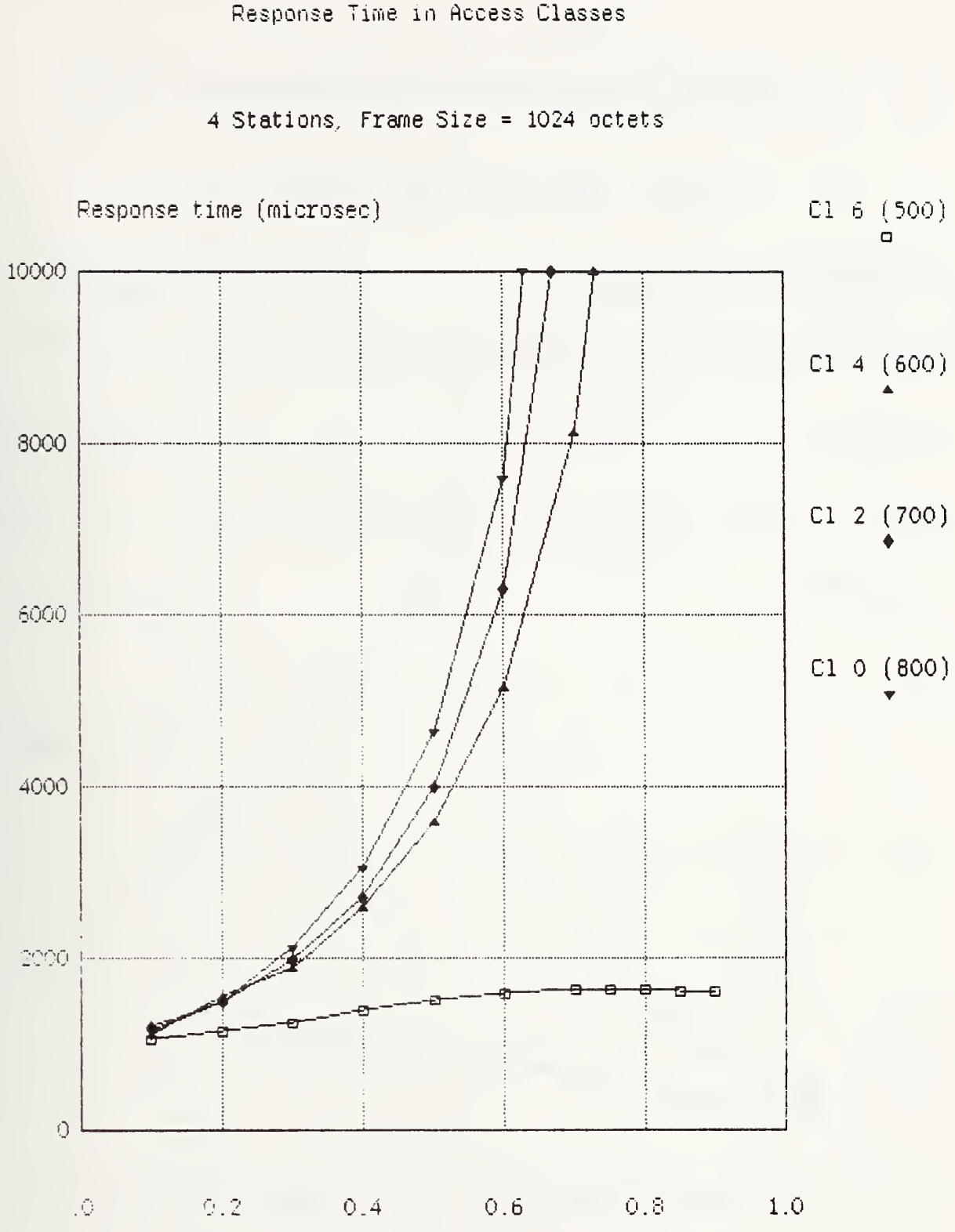


Fig. 4 Normalized Traffic Intensity

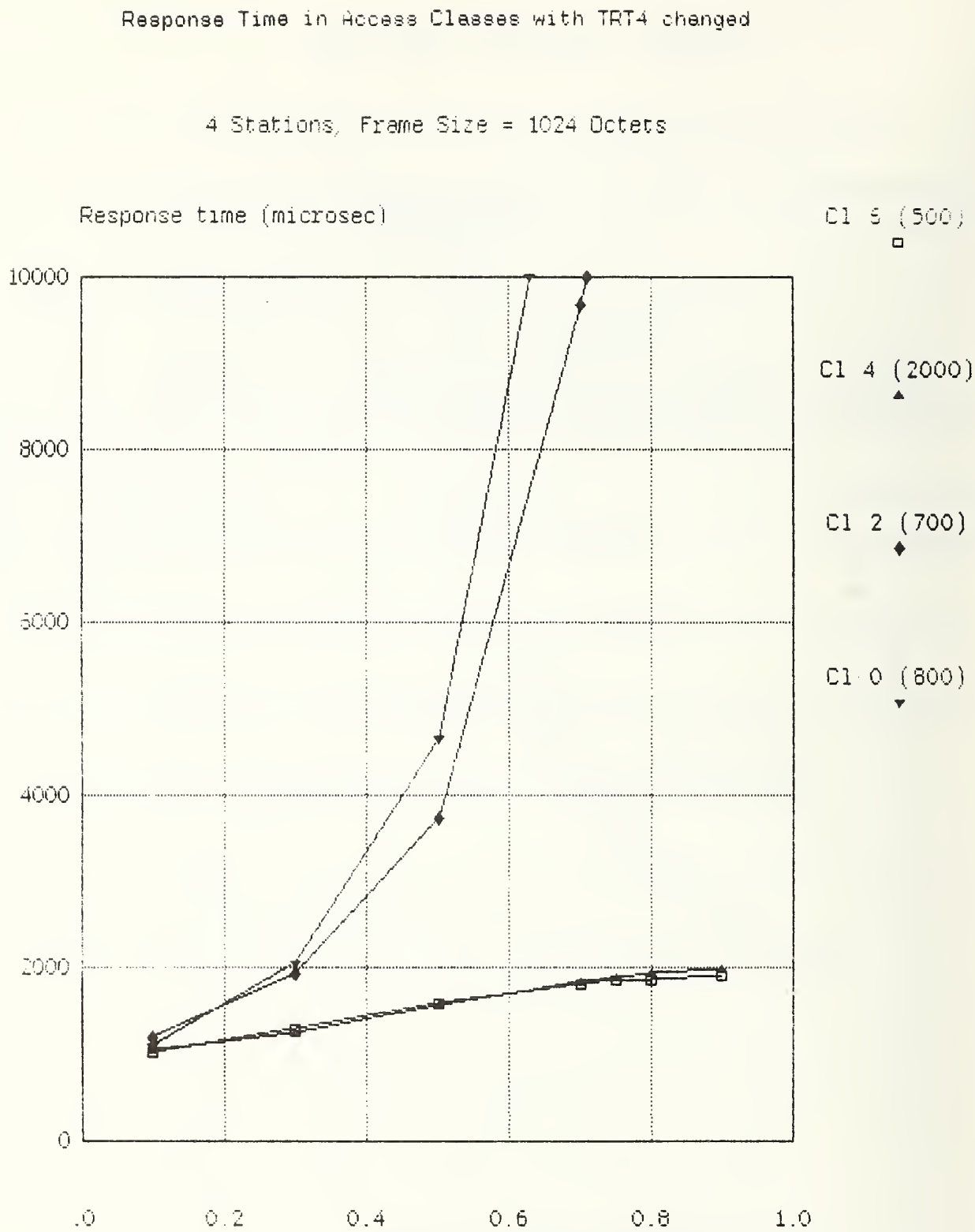


Fig. 5 Normalized Traffic Intensity

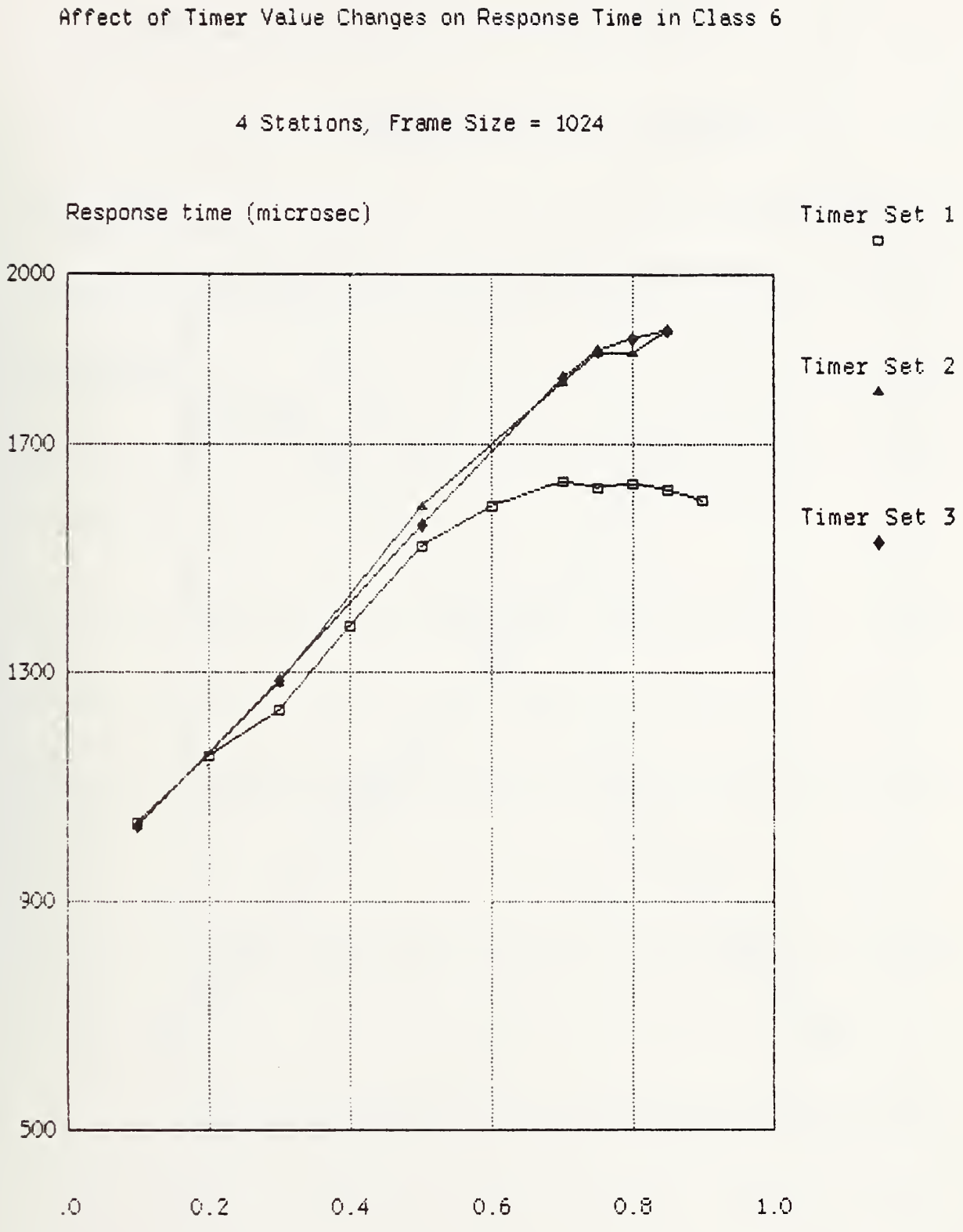


Fig. 6 Normalized Traffic Intensity

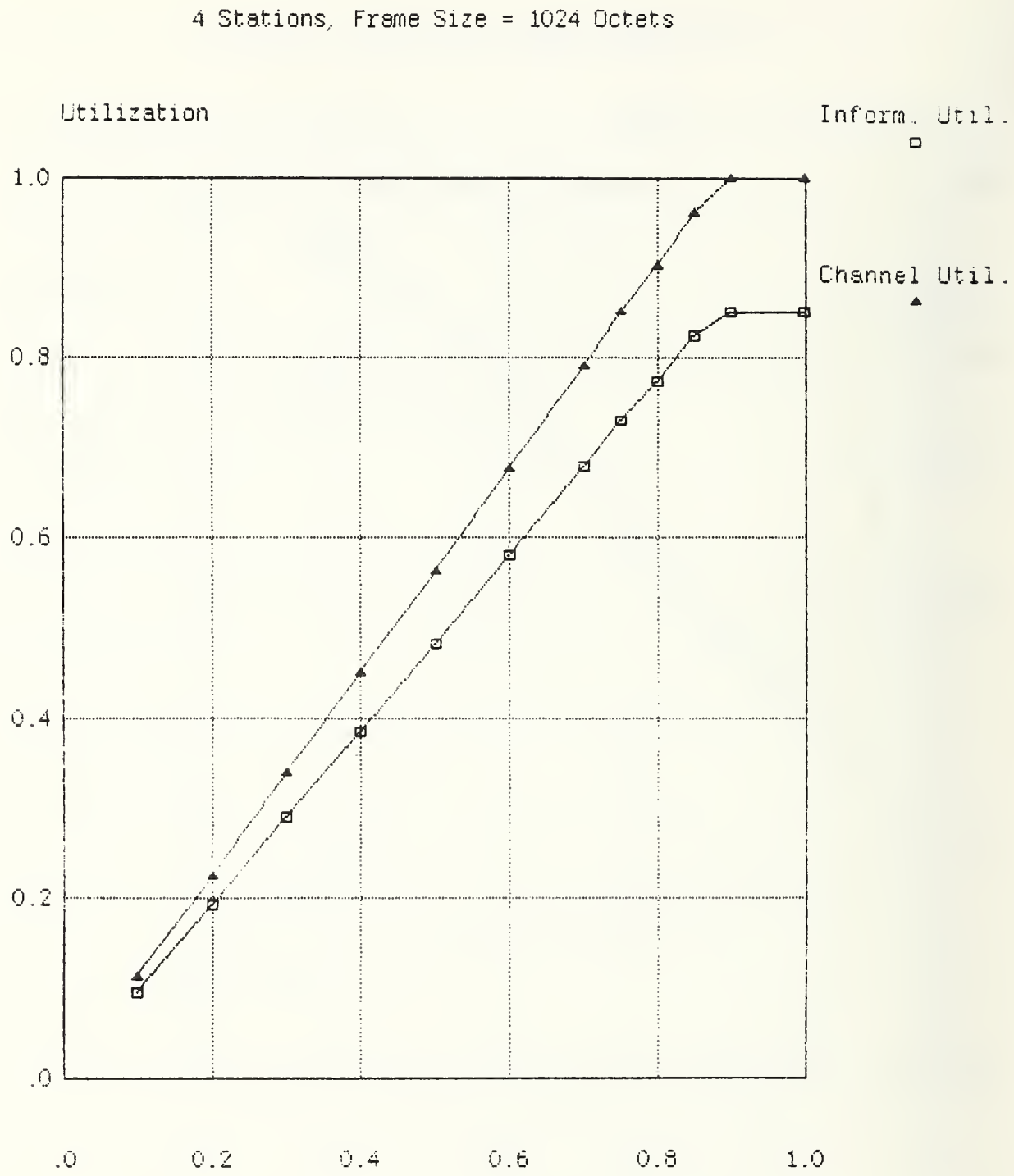


Fig.7 Normalized Traffic Intensity

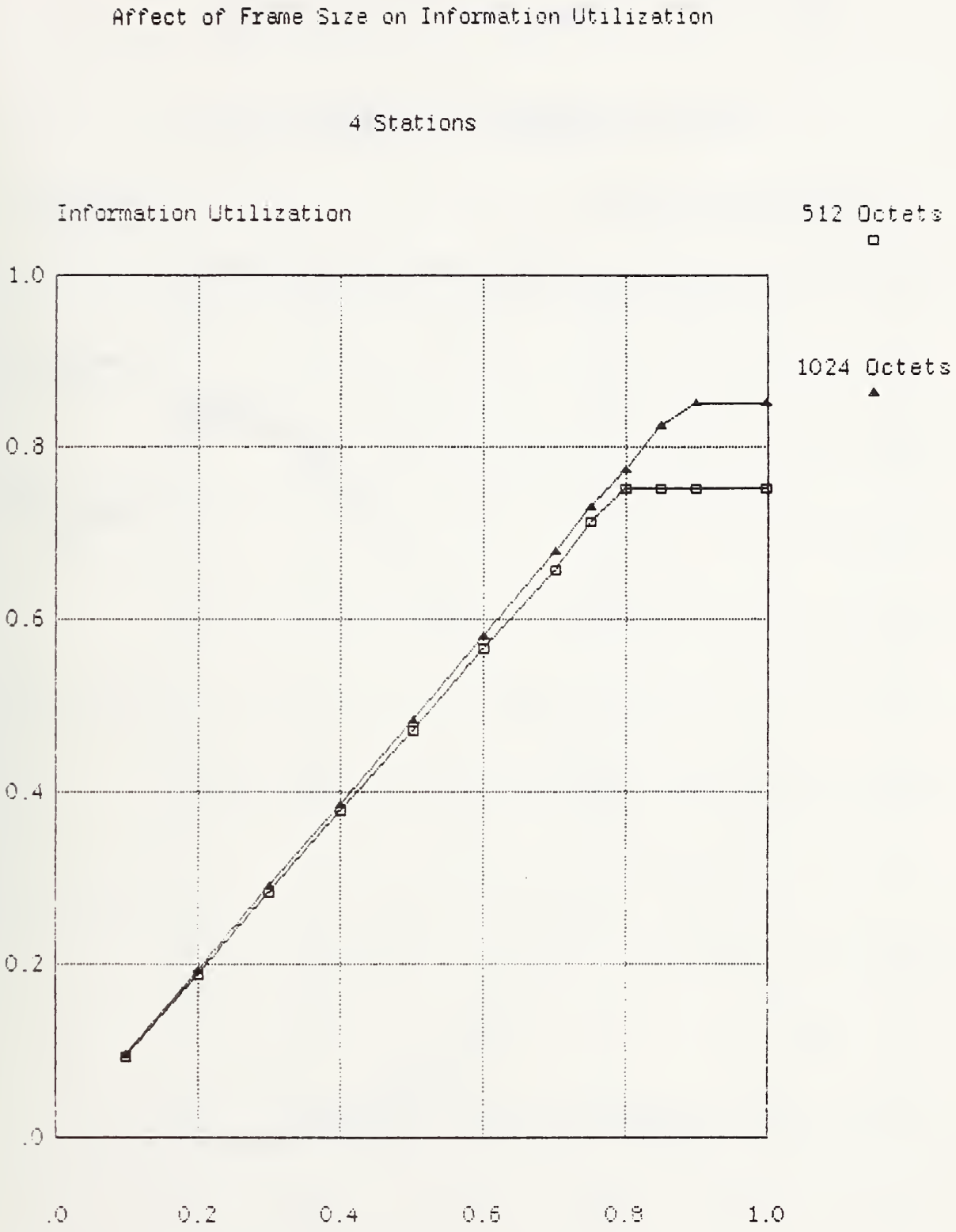


Fig. 8 Normalized Traffic Intensity

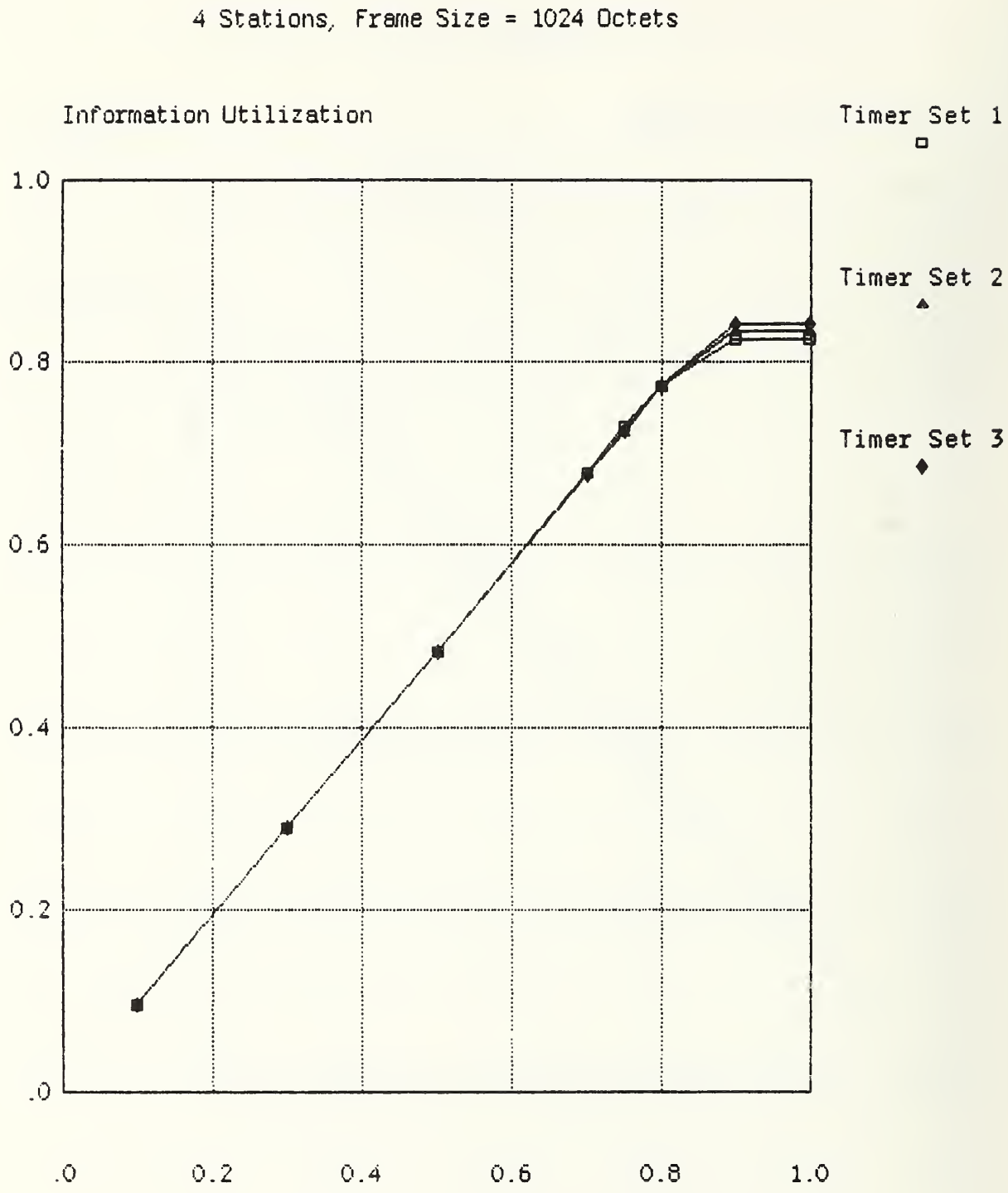


Fig.9 Normalized Traffic Intensity

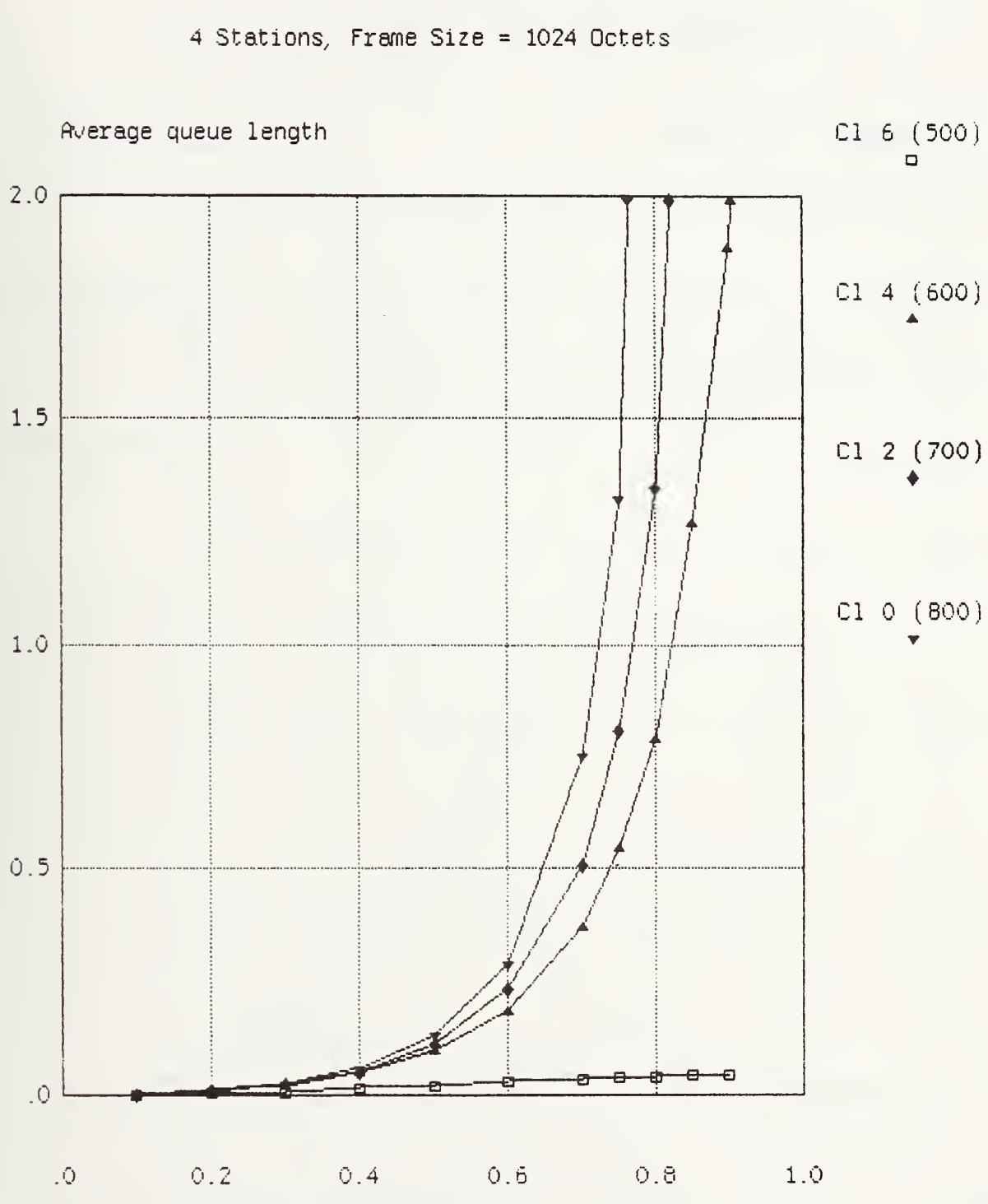


Fig. 10 Normalized Traffic Intensity

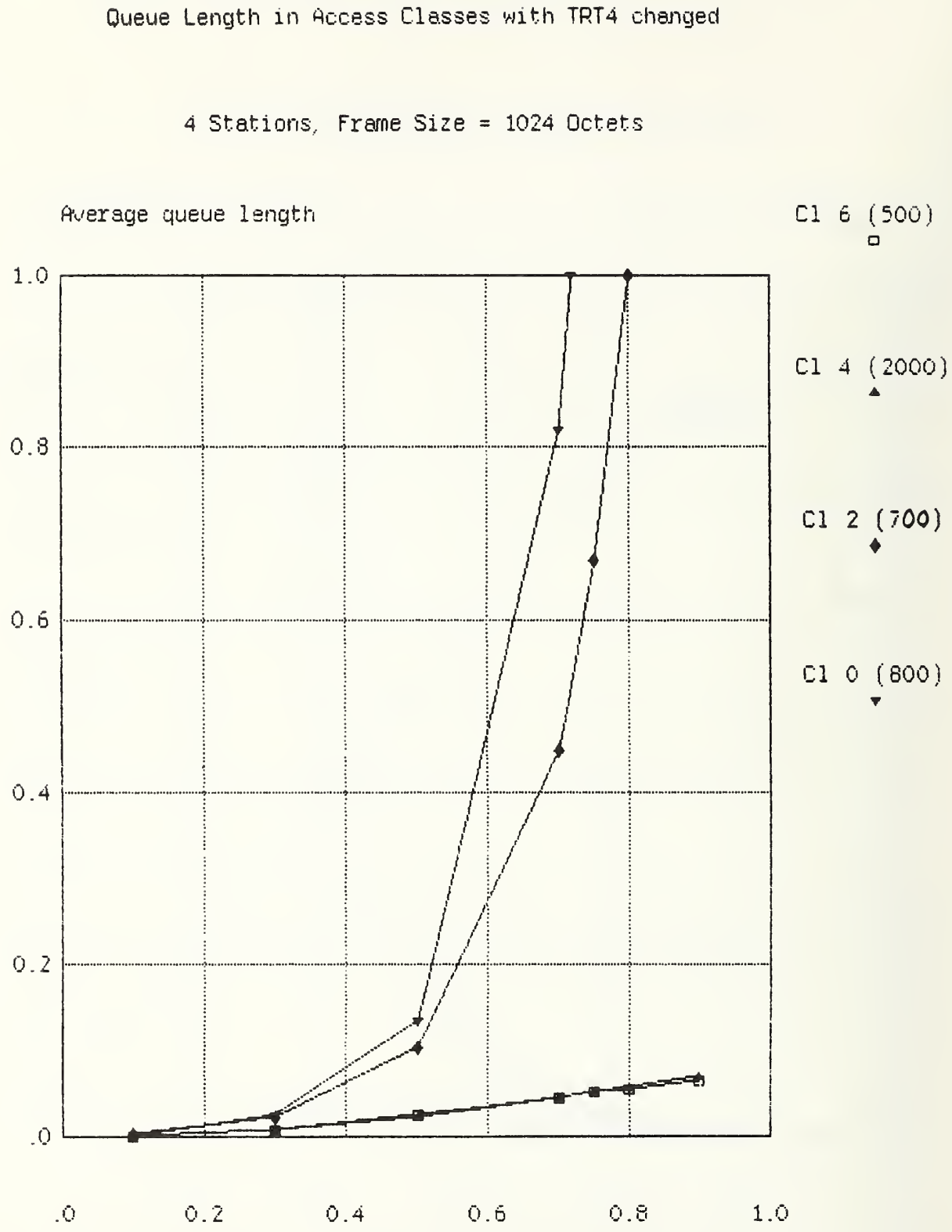


Fig.11 Normalized Traffic Intensity

Affect of Timer Values on Queue Length in Class 6

4 Stations, Frame Size = 1024 Octets

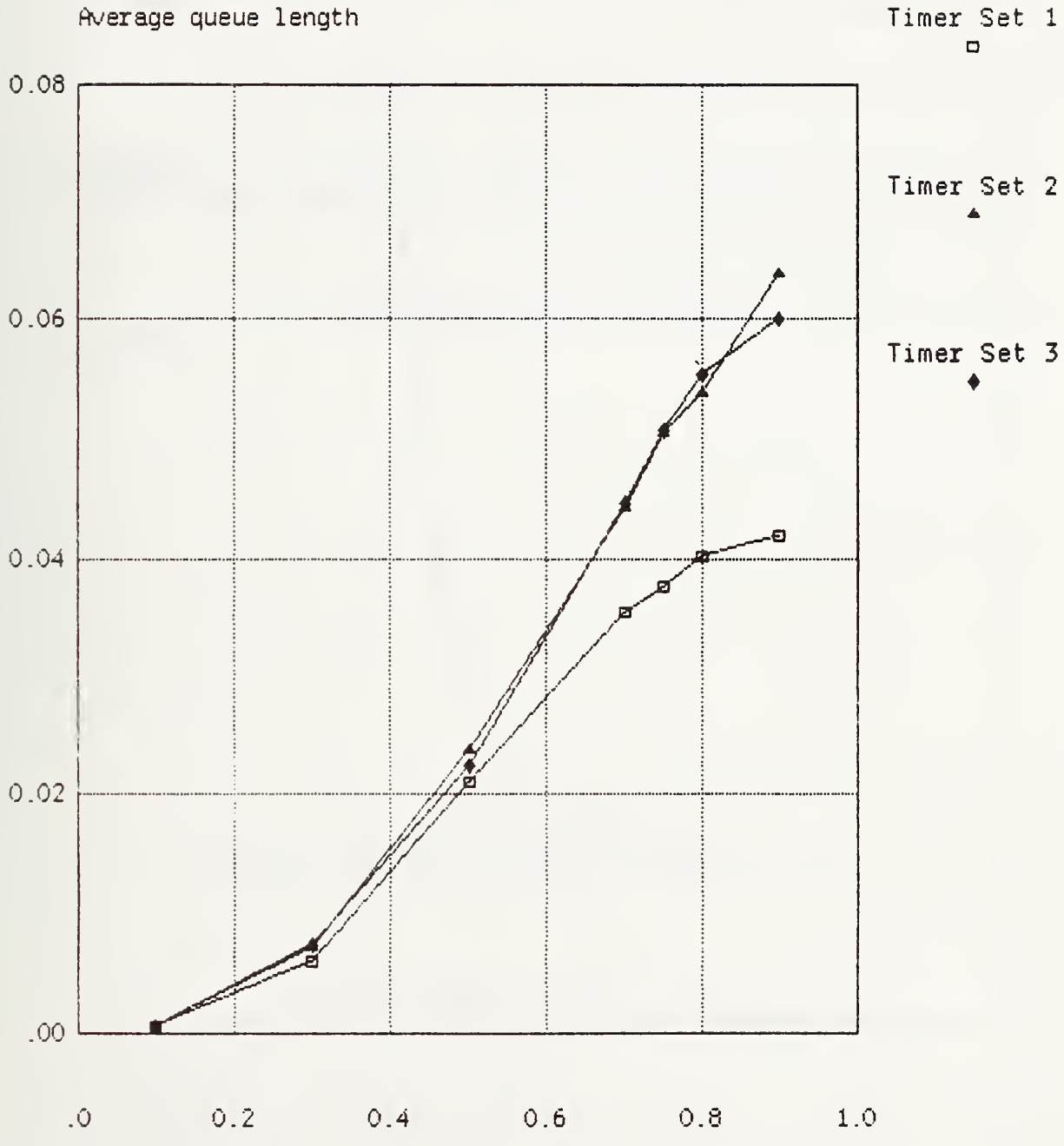


Fig.12 Normalized Traffic Intensity

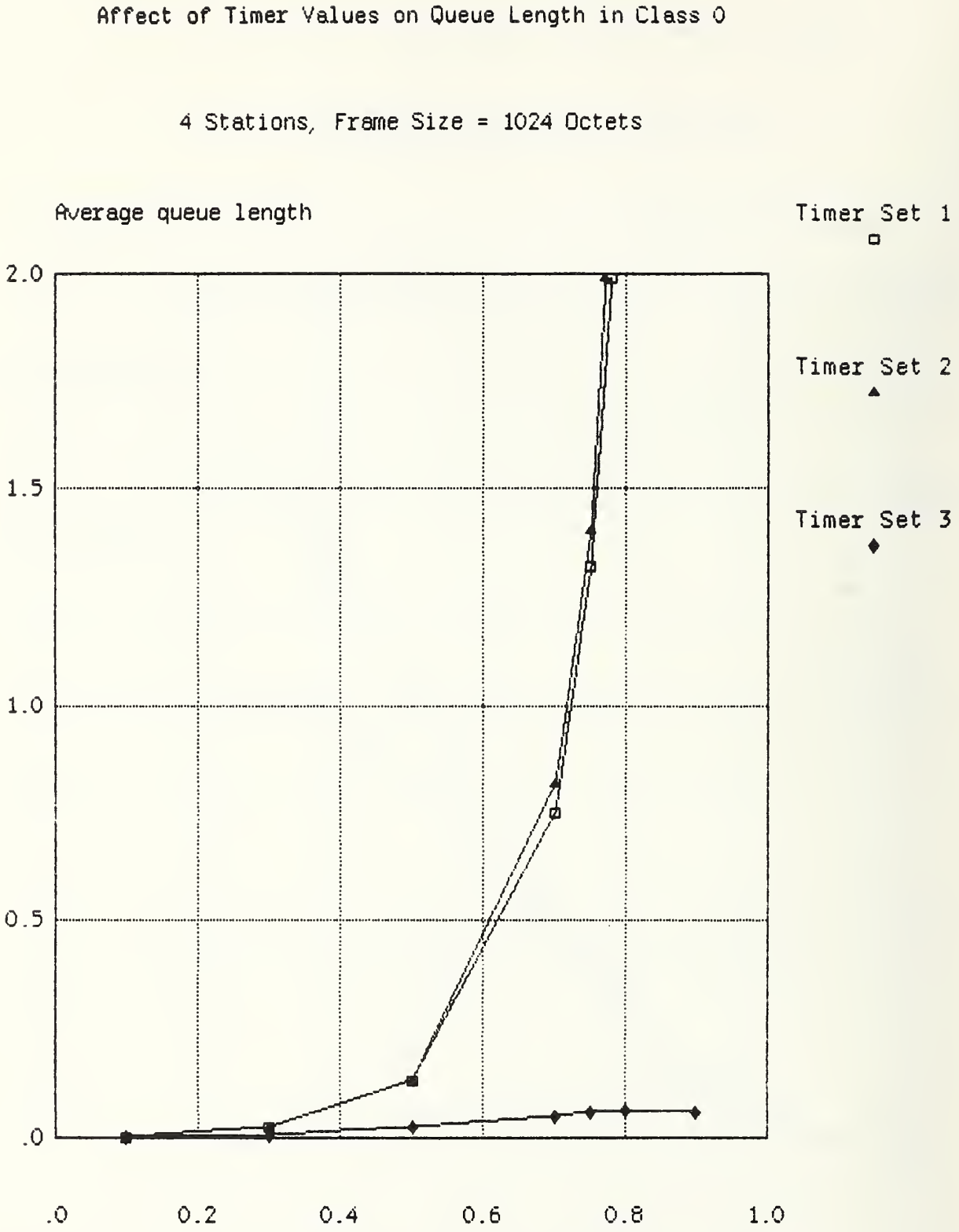


Fig. 13 Normalized Traffic Intensity

DISCRETE EVENT SIMULATION OF THE IEEE 802.4 TOKEN BUS LAN PROTOCOL

A STRUCTURED ANALYSIS APPROACH

Edward R. Nugent
Boeing Computer Services
Engineering Technology Applications Division

April, 1985

Abstract

Boeing Computer Services is developing a discrete event simulation of the IEEE 802.4 Token Bus Media Access Control protocol. NBS will use the simulation model as part of their token bus research project. This paper describes the BCS simulation approach. Topics include project background, objectives and the simulation methodology used.

Background

The Boeing Company is dedicated to the integration of multi-vendor equipment to solve its manufacturing and office automation requirements. The standardization of communication protocols for these various components is an area of vital importance to our company, and Boeing strongly supports General Motors' Manufacturing Automation Protocol (MAP) as a very positive step in this standardization process. In fact, to complement the MAP activity, Boeing has initiated the development of Technical/Office Protocols (TOP) project, which defines the suite of standard protocols necessary to support the office automation and Engineering/Scientific computing environment.

The National Bureau of Standards (NBS) is managing a cooperative research project aimed at understanding and evaluating the behavior of local area networks that conform to the IEEE 802.4 standard. This is the Media Access Control (MAC) method called for by the MAP specification (GM84). The NBS Institute for Computer Sciences and Technology is leading this research project by establishing a Research Associate Program, which provides a mechanism for industry to jointly participate with government in research projects. The goals of the NBS effort are to enhance the understanding and competence in token bus technology, to make the cooperative research results available to the voluntary standards arena, and to issue Federal Information Processing Standards and Guidelines for token bus local area networks. The vehicle NBS will employ to accomplish its objectives is the establishment of a performance testbed laboratory.

Boeing's role in this research effort is to provide the predictive analysis tools (discrete event simulation) needed by the research group in order to effectively use the NBS performance testbed. This laboratory is currently under construction at the Gaithersburg, Maryland campus.

The NBS Local Area Network group presently has a preliminary discrete event model (NBS84) and some excellent analytical models (NAK85a) (NAK85b) which were developed in house. Boeing's primary objective is to build on this extensive modeling base to develop additional model(s) that are tailored specifically to the research of token bus performance on the NBS testbed facility.

The approach taken has been to design and implement a "virtual" testbed (e.g. a simulation tool to model hypothetical local area network

environments). This will allow economical testing of general performance issues, in order to direct the final experimentation on the "real world" testbed. The virtual testbed will also perform experiments outside the scope of the NBS lab (i.e. very large networks). This parallel testbed concept will increase the researchers confidence in the statistical inferences made from analysis of real data, as well as add some measure of validity to the model's ability to predict IEEE 802.4 protocol performance.

The virtual testbed will be a complete simulation system, which will include a DBMS, graphical display capability and the simulation kernel.

Finding the Proper World View for MAC Simulation

The following section describes a methodology for building MAC protocol research models. Although the discussion will be limited to IEEE 802.4, we believe this methodology has widespread applicability to model building in general, especially when the environment includes specialists working as a project team.

Classical software development methodologies do not work well when the objective is the simulation of a system. This is because model development naturally tends toward the scientific method.

Scientific methodology can be described in the following 6 steps (GRAY80), which are compared to their model building counterparts in Figure 1.

STEP	SCIENTIFIC METHOD	MODELING REQUIREMENT
1	Observe System	Same
2	Formulate Hypothesis	Develop Model
3	Make Predictions based on Hypothesis	Use Model to Predict System Behavior
4	Compare Predicted Results to Observed Results	Validate Model (compare to real system if possible)
5	Change Hypothesis if Differences Exist	Modify Model if Necessary
6	Repeat Step 3 and on	Same

Figure 1

As seen by the loop at step 6 this is an iterative methodology, where the desired system is fixed (the system to be modeled) and the software is developed by finding the best representation of that system. This model then determines the needed inputs to the software and the types of outputs available from it.

This approach is diametrically opposed to classical software development which prescribes the definition of system outputs as one of the first steps to be taken. Why is this not the case in simulation software development? The answer lies in the nature of modeling itself. It is often possible to build models which are good predictors of some system attributes and poor predictors of others. For example, a model may be a good predictor of token rotation time and a poor predictor of maximum delay encountered by a frame to be transmitted from access class 6.

Along these same lines, there is no guarantee that if outputs were possible to define before model development, that those outputs would be a complete and adequate description of that system. Certainly this does not preclude the simulation analyst from having a knowledge of what information is desired from the simulated system. It does mean, however, that the exact form of that data may not be known until the design process is complete.

This leads to difficulty in managing simulation projects where teams of researchers are working together to integrate database and data management with simulation in a complete simulation system environment. What is required is a formal but flexible methodology, that promotes communication among the research team. A structured methodology based on data flow diagrams is proposed as the tool to solve this requirement. Not only does a structured technique offer the change mechanism needed in model iterations, but it also provides the formal documentation required by classical software methodologies.

A Stuctured System Approach

The approach taken has been tailored from Dickenson's methodology using structured techniques (DICK81). The content of the methodology is partitioned into the following components:

1. Data Flow Diagrams (DFD), which describe the system and the model in a mostly graphical form and shows the relationships of processes and data movement between processes without regard to time sequencing.

2. Data Dictionary, which defines all data used in the methodology, including aliases for data used in multiple processes.

3. Process Descriptions, which define each process in detail. At the lowest level this becomes the simulation pseudocode.

The methodology calls for a specific structure to the ordering of the

DFDs. This is a hierarchical structure, with each descending level offering a greater level of detail. The general form of this ordering is shown in figure 2.

The methodology calls for "explicit iterations", which means that no level is fixed in content until the end of the project. This is a major advantage to modeling projects that require cycling to achieve the desired result.

Application to Token Bus Simulation

The following section describes the application of this methodology to the development of an IEEE 802.4 Token Bus Simulation. The diagrams that are used to illustrate this section are excerpts from the full DFDs, data dictionary and process descriptions. The sections chosen show how this technique can be used to start from a high level view of the problem and work down to a specific simulation process.

The methodology begins with the development of the context diagram. Figure 3 shows the context diagram for a Token Bus Simulation system. It provides a very general view of the expected outside interaction with the system. The context diagram in conjunction with the level 0 diagram (figure 4) and their associated data dictionary and process descriptions define the requirements for the IEEE 802.4 simulation.

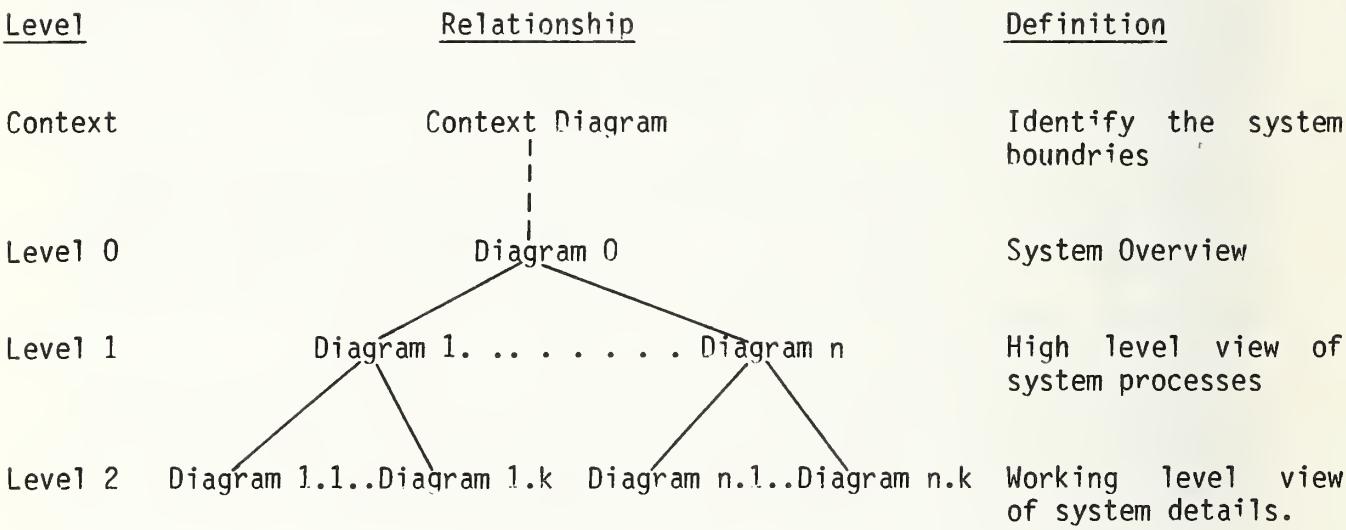
These diagrams were developed after considerable time was spent learning the needs of the NBS token bus researchers and understanding the IEEE 802.4 specification. Although the context diagram is not expected to change, the level 0 diagram is a candidate for change based on definition of additional requirements to the basic system.

The level 1 diagrams are now developed in an activity analogous to preliminary design. Figure 5 shows the high level view of the simulation process, which was identified as process number 5 on Diagram 0. At this level the basic simulation components are identified and described. The level 1 diagrams are more susceptible to change than the level 0 diagram during the iterative process, although major changes are not anticipated here.

This process sets the stage for the detailed design of the system. The detail design is developed using the level 2 diagrams as a working tool. At this level the full power of the methodology becomes apparent.

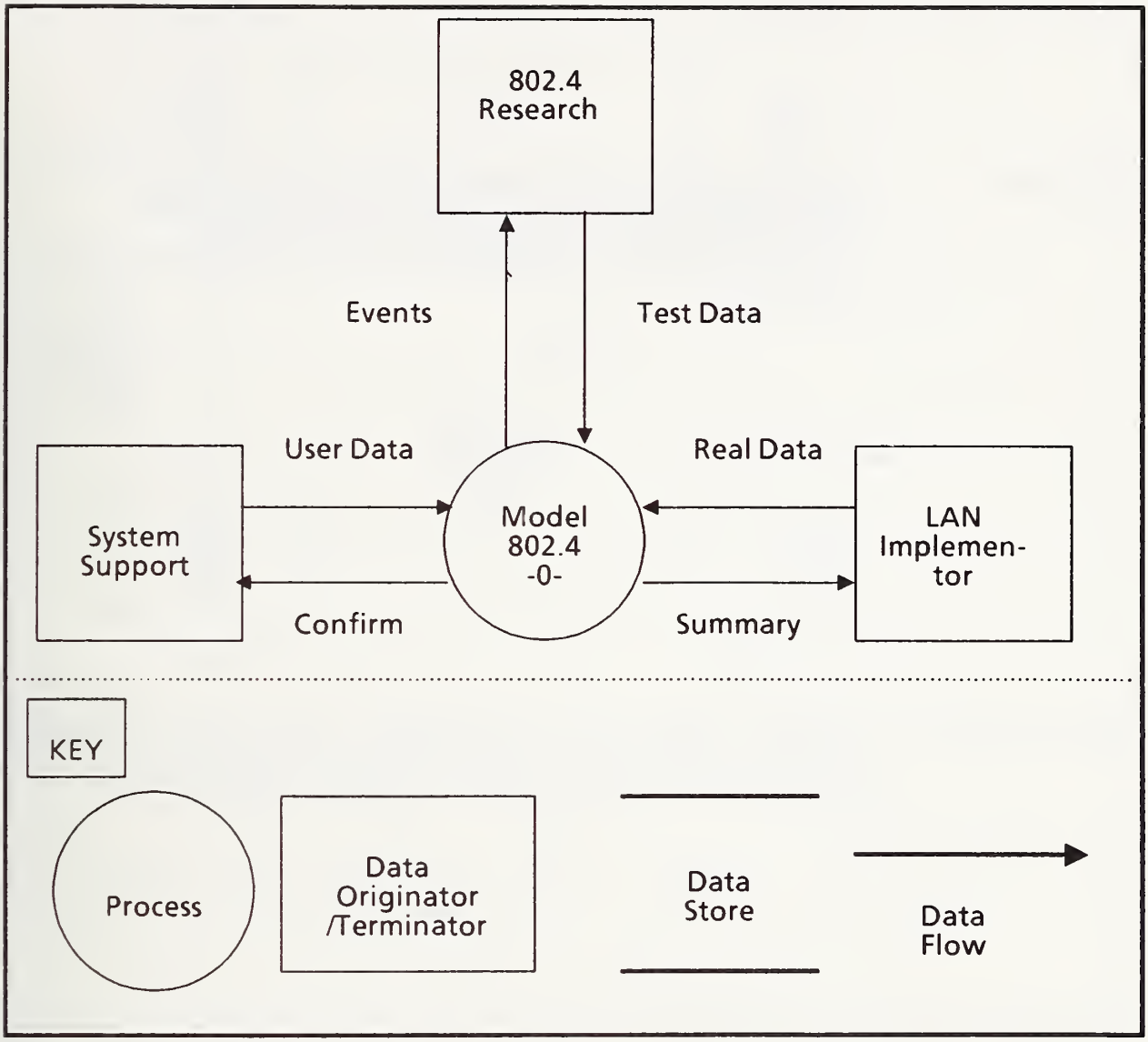
The higher layer diagrams (Context through Level 1) document the "what are we modeling?" aspect of the simulation process. In other words, they record our observations of the system. Level 2 diagrams, on the other hand, assist in and document model formulation. This will answer the question "how are we going to model it?"(STREI81).

Figure 6 shows the level 2 diagram for the process of transmitting a message on the physical media. This process was identified in Diagram 5



DFD Hierarchy

Figure 2



(a)

Context Level

Figure 3

(a) Context Diagram

Test Data - all data required to set up the simulation program to run experiments on 802.4 protocol performance. Includes simulation control data, station management data, configuration data, experimental test parameters and display control data.

(b)

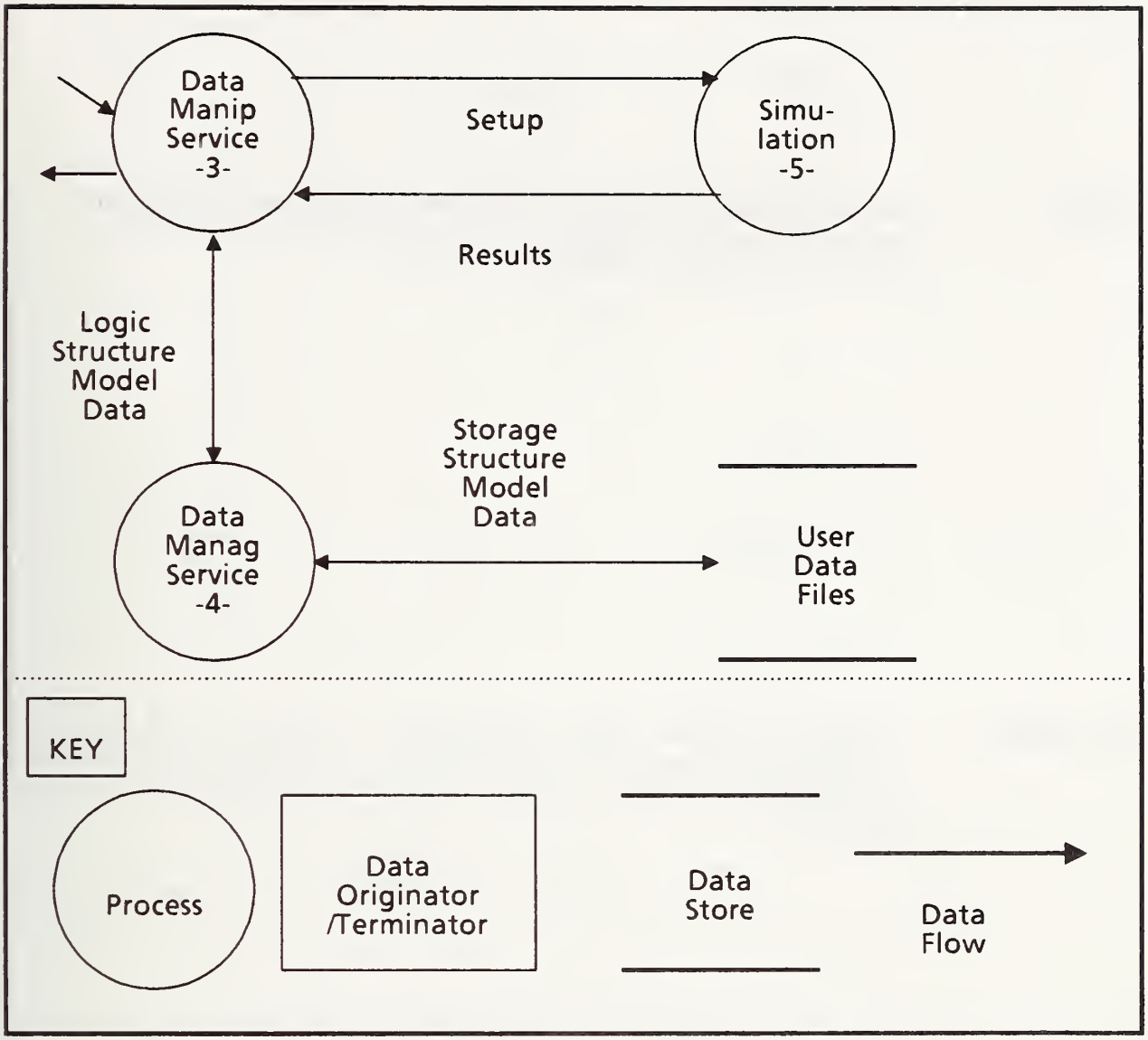
Model Protocol - provide a discrete event simulation system that will predict performance of an IEEE 802.4 network. Include capability to manage, store and display data for multiple experiments.

(c)

Context Level

Figure 3

(b) Sample Data Definition
(c) Process Description



(a)

Level 0

Figure 4

(a) Partial Diagram 0

Setup- All data required to initialize the model to perform a single 802.4 simulation. See level 1 and 2 diagrams for complete list of data elements included in this data flow.

(b)

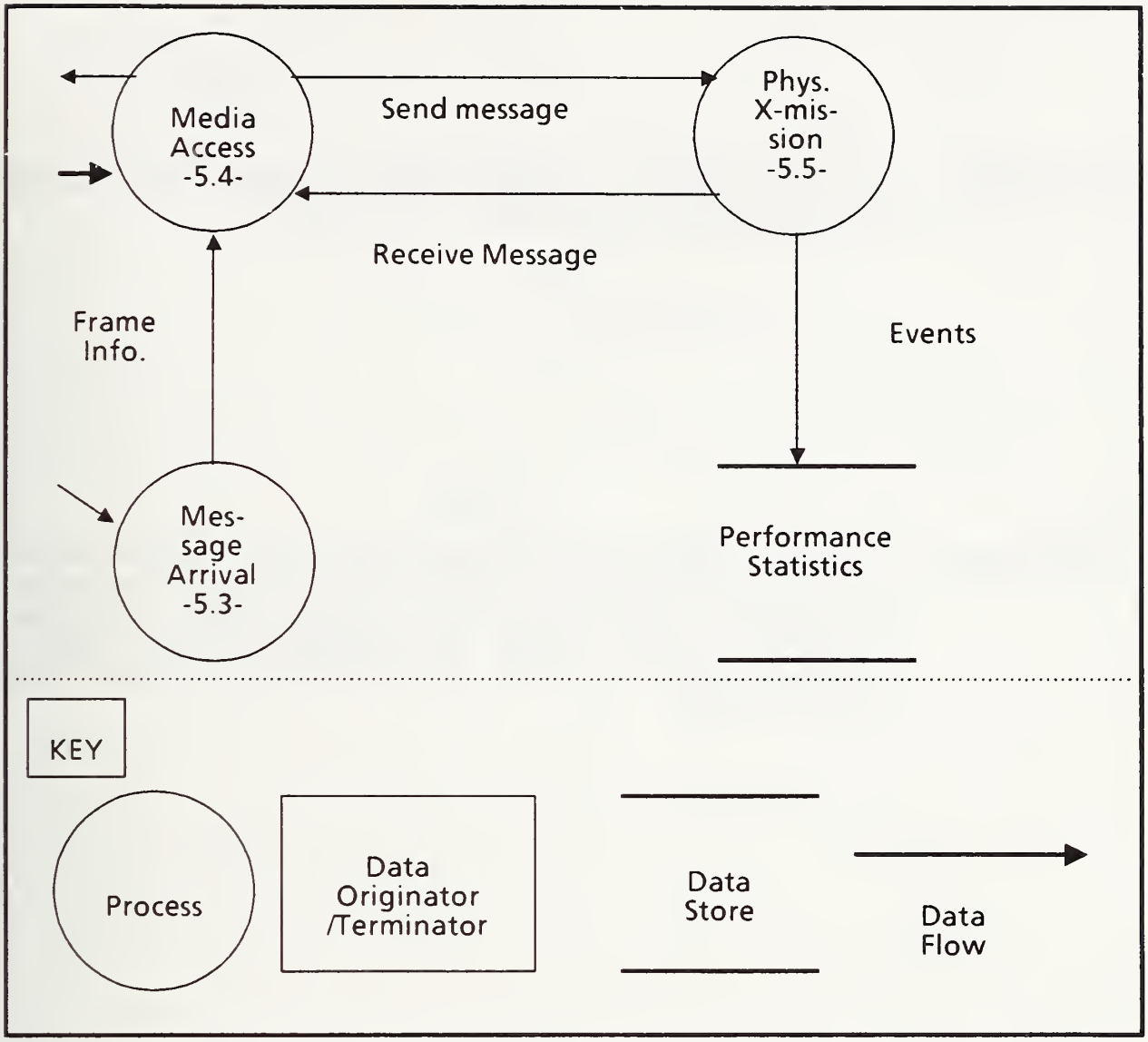
Simulation- Perform discrete event simulation of IEEE 802.4 protocol performance based on setup data. Provide results to data manipulation service in form of event listing or summary statistics.

(c)

Level 0

Figure 4

(b) Sample Data Definition
(c) Sample Process Description



(a)

Level 1

Figure 5

(a) Partial Diagram 5

Send Message- All data required to transmit a frame on the media. Includes preamble length, frame length, SA, DA, and FC.

(b)

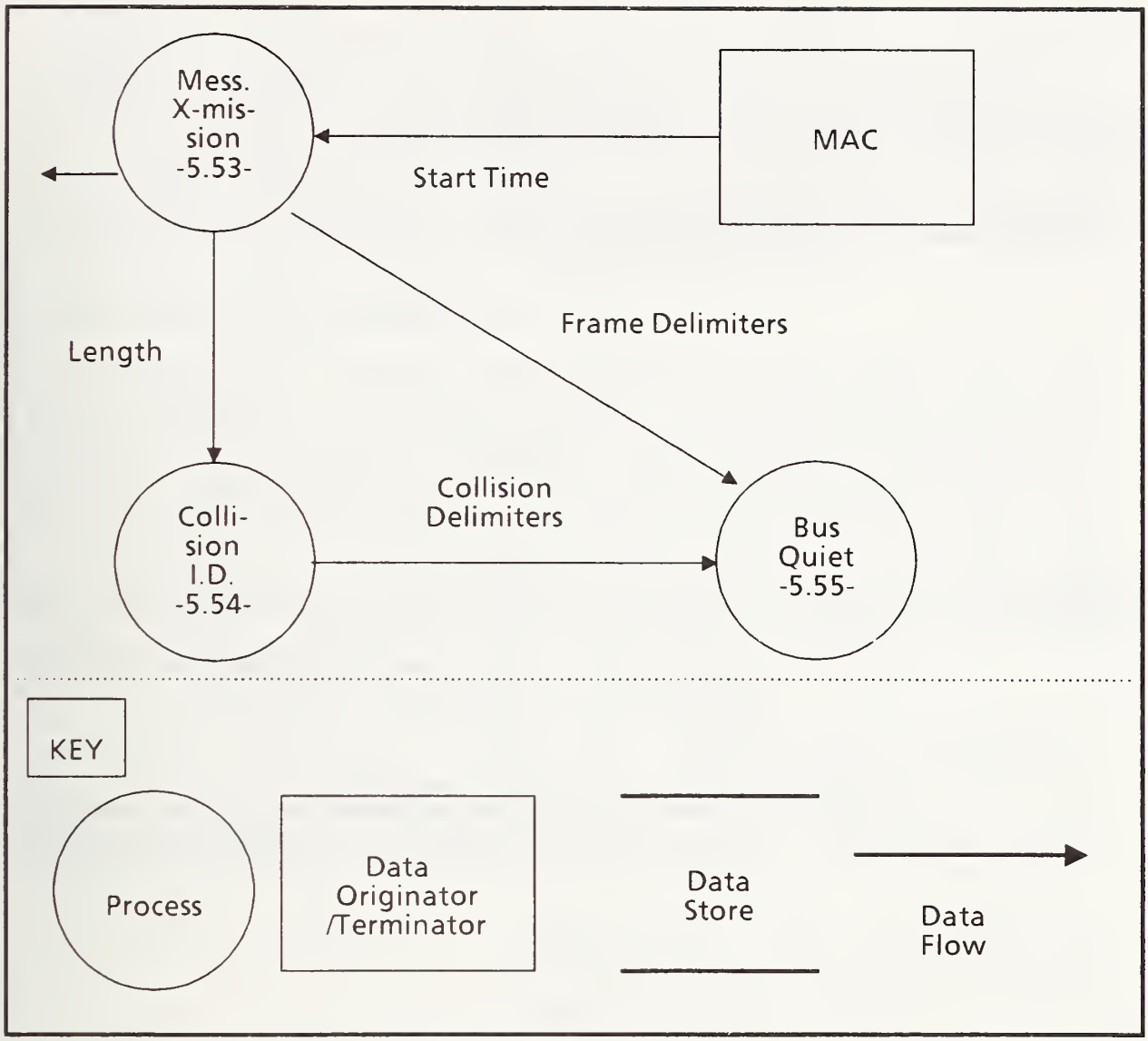
Physical Transmission- Send a message from the station transmitting to the head end and return to all stations on the forward channel. This process will identify those frames that collide during contention processing and will handle the operation of the following boolean variables; bus_quiet, Rx_Protocol_Frame, Rx_Data_Frame and Noise_Burst.

(c)

Level 1

Figure 5

(b) Sample Data Definition
(c) Sample Process Description



(a)

Level 2

Figure 6

(a) Partial Diagram 5.5

Frame Delimiters- SD, ED, and first preamble bit.

Collision Delimiters- SD and abort sequence

(b)

Collision Identification-

Event 1: More than 1 frame or noise burst arrives at the head end.
Action 1: Send abort to all stations and intercept incoming data . Replace this with pseudo-silence until a valid SD is heard.

Event 2: SD is recieved at the head end.
Action 2: If no other signals at current time in the head end, send the SD to stations. This will set the bus_quiet indicator to false.

(c)

Level 2

Figure 6

(b) Sample Data Definition
(c) Sample Process Description

(figure 5). At level 2 we break the process down to the simulation components themselves. At this point the type of simulation (i.e. discrete, continuous) need not be identified. The process descriptions at this level identify the events, processes and functions of the real world system judged to be critical to the system by the token bus researchers. These process descriptions and associated data flows and data dictionary become the documented detail design and provide the pseudocode for the construction of the simulation program.

Special Simulation Considerations

Validation of the simulation model is greatly eased by the use of the level 2 diagrams. Validation of a model is a check to ensure that the model formulated is a true representation of the system being simulated. The level 2 work provides a complete, clear, concise and non-redundant method of communication between the researcher validating the model and the simulation analyst building the model. The validation consists of the determination that all possible critical aspects of the system have been identified. In this project additional validation/verification will be accomplished through comparison to the testbed facility and other discrete event models. Paper validation by the above process will precede run validation.

In classical software development programs, choice of the computing environment is made early in the process. Boeing Computer Services has a wide range of simulation software available for use. The choice depends on the identification of the best "world view" for the simulation project. Different systems are best viewed from one of several "world views". The general classifications of these views are:

1. Event Orientation: the system is viewed as a series of discrete events that trigger changes in the state of the system and schedule subsequent events.

2. Process Orientation: the system is viewed as a set of discrete events that trigger some standard process execution that will change the state of the system.

3. Continuous Orientation: the system is viewed as a series of continuous functions that cause smooth changes to the state of the system.

4. Hybrid: some combination of the above orientations.

Classification of the system can be made by examining the level 2 process descriptions. This will point us in the direction of the simulation language best suited for MAC modeling. The ability of the language chosen to interact with other software directs us toward the

support software required to complete the total simulation system.

Coding, Verification, Testing and Documentation

Final simulation coding, verification and testing of the simulation software also proceed from the level 2 process descriptions. Verification in this sense is assurance that the simulation code embodies all the critical items detailed in the process flows. The test plan is developed so that all events shown will occur during one of the simulation test runs.

It has been mentioned throughout this paper that this methodology will provide documentation required of classical software development. Two documents, the Systems manual and the Users manual do not come directly from the structured techniques described. The production of these documents is, however, made much easier with the documentation provided by the methodology.

Finally, it should be noted that the development of the system does not require the completion of a given level before beginning the next. Actually, it would be surprising if this were to happen. As we have seen, the methodology allows detail design to influence preliminary design, due to its ability to iterate through the design process. This provides the flexibility required during simulation development. The documentation provides the communication tool for the research team to stay on top of these changes and work in harmony.

Summary

A methodology for developing a Token Bus Simulation of IEEE 802.4 has been described. This structured analysis technique is more appropriate than classical software development methodologies because it is:

1. A complete, clear, concise and non-redundant method of describing an IEEE 802.4 LAN simulation system.

2. A dialogue for communication between the simulation analyst and other token bus researchers.

3. A working tool.

4. Current throughout the course of the project.

5. A method that produces required documentation as a natural product of analysis.

Acknowledgement

I would like to thank Dan Streiffert for his valuable contribution as the originator of this simulation methodology at The Boeing Company. I would also like to thank Bob Slate, Paul McElwain, Dave Wolf and Mark Jones for their thoughtful assistance in applying this metholdology to the simulation of IEEE 802.4 protocol performance.

References

(DICK81) Dickenson,B.,"Developing Structured Systems",New York: Yourdon Inc.,1981.

(GM84) General Motors "Manufacturing Automation Protocol", Warren MI: 1984.

(IEEE84) Institute for Electrical and Electronic Engineers. IEEE Standard 802.4, Draft F, July, 1984.

(GRAY80) Graybeal,W.J.,Pooch,U.W.,"Simulation: Principles and Methods", Cambridge: Winthrop Publishers Inc., 1980.

(NAK85a) Nakassis,A."On the Stability of a Token Passing Network", Proceedings of the Workshop on Analytical and Simulation Modeling of IEEE 802.4 Token Bus, to be published by NBS:1985

(NAK85b) Nakassis,B."Token Passing Networks and Starvation Issues", see above.

(NBS84) National Bureau of Standards, Archambault,J.L., "An IEEE 802.4 Token Bus Network Simulation", NBSIR 84-2966, October, 1984.

(STREI81) Streiffert,D.L.,"Structured Analysis Techniques Applied to Discrete Simulation and Modeling", Internal report. Boeing Computer Services:1983.

SIMULATION OF THE IEEE 802.4 TOKEN PASSING BUS PROTOCOL USING SIMSCRIPT

A.R.K. Sastry and M.W. Atkinson

Information Sciences Group
Rockwell International Science Center
1049 Camino Dos Rios
Thousand Oaks, California 91360

Abstract—A simulation model has been developed for the performance evaluation of the IEEE 802.4 token passing bus local area network protocol using SIMSCRIPT. The model has identifiable 'processes' corresponding to the four 'machines' of the protocol, i.e., access control, receive, transmit, and interface machines. In addition, a 'frame process' is used to simulate the signal flow on the bus. An initialization 'routine' serves to input the network parameters and to initially activate the processes in the proper order, while a statistics extraction routine gathers output data during a simulation run. The entire model is developed in an incremental mode, gradually increasing the detail and complexity so that code can be validated by 'walking through' at every stage of the development. Queues with four different priorities, a message generation process at each queue, random selection of frame lengths, and token rotation timers have been incorporated. Results from a number of simulation runs suggest the need to develop methodology to relate the timer values with the desired priorities under given traffic conditions, which seems to be a very significant user–oriented issue.

1. INTRODUCTION

The IEEE 802.4 draft Standard token–passing bus scheme [1] is a medium access control procedure for local area networks implemented on a broadcast bus. It belongs to the medium access sub–layer in the data link layer of the Open System Interconnection (OSI) model [2] of the International Standrds Organization (ISO), as shown in Fig. 1. The scheme is one of the three considered for standardization by the IEEE 802 committee for local area networks, the other two being the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) and the Token Passing Ring. Unlike the CSMA/CD and token passing ring, which have been well studied in the literature [3],[4], the token passing bus as described in [1] is relatively new, and its performance and design issues have not been investigated yet in detail. The token passing bus scheme has a deterministic bound on delay and becomes attractive at low bus propagation delays and high message rates [5]. There have also been some studies on the correctness and completeness of the protocol [6],[7], when it was in its early stages of the standardization process. Further studies on its validation and performance aspects are needed.

This paper describes some preliminary work on development of a simulator for the scheme as specified in [1] and its use in performance evaluations. The simulation effort is divided into two incremental parts: (i) a performance part that includes all the data and token transmission mechanisms of the scheme, assuming that all stations are active all the time and that the network is functioning normally without noise bursts, missing tokens, etc.; and (ii) a network maintenance part that includes token claims, entry and exit of stations, contention resolution mechanisms and so on. The performance part has been completed and is found to be a valuable tool in assessing throughput–delay performance under normal network operating conditions. The network maintenance functions are currently being implemented, which, when completed will allow performance analyses in more realistic settings and investigations on the correctness and completeness of the protocol. The subject matter of this paper is on the performance part. The token passing bus scheme is described briefly in section 2. The key features of SIMSCRIPT, the language chosen for the simulation, are summarized in section 3. The simulation methodology, description of the model, validation procedures, and the mechanism of collecting statistics are described in section 4. Some performance results obtained from simulation runs are discussed in section 5. Section 6 gives conclusions.

2. THE TOKEN PASSING BUS SCHEME

In this scheme, stations connected to a bus form a logical ring based on the descending order of their addresses (unlike in a ring network in which stations would be physically connected serially). A station is allowed to transmit data only when it receives a 'token' (a control frame), which represents a right to access the bus. After transmitting data, a station hands over the token to the next active station in the logical ring. The communication on the bus thus has two phases; i.e., a token–passing phase and a data transfer phase. Detailed procedures have been specified in the draft standard document [1] on the data transfer mechanism and network control.

Each station's protocol is made up of five components: the Access Control Machine (ACM); the Receive Machine (RXM); the Transmit Machine (TXM); the Interface Machine (IFM); and a station management unit. At each station, there is provision for an optional four–level priority mechanism for traffic handling, with each level having a separate queue. The transmission of data frames from each queue is conditioned by the residual value of a target token rotation timer for that queue at the time of receiving the token. If the timer expires, transmission of data from that queue is not allowed. The timer is reset with an initial value before beginning transmission of data frames or before passing the token if no data frames are to be transmitted, as the case may be. Timers at different queues may be assigned different initial values depending on the priorities desired. Thus, if a large number of data frames are transmitted from one queue (using up a long time) in a given token rotation, lower priority queues in that station and queues at other stations get relatively

less time for transmissions in that rotation, depending on the initial values assigned for the timers of those queues. This interaction among the timers is incorporated in the scheme to enforce fairness in bus access allocations. While considerable flexibility exists in tailoring the priority scheme to the needs of specific applications, implementation of a given priority scheme through selection of appropriate initial values for the timers is somewhat complex due to the interdependence among the timers. The complexity grows with increase in the number of stations. Other timing interrelationships determined by such parameters as bus transmission rate, frame sizes, propagation delay, and interface processing delay also become important in determining the achievable throughput [8]. Simulation aids in understanding some of these issues that are rather difficult to analyze.

3. SIMSCRIPT: LANGUAGE STRUCTRE Vs NETWORK ELEMENTS

Communications networks belong to a class of physical systems that can be studied effectively using discrete event computer simulation models. Those networks of interest usually involve a complexity that extends beyond the restrictive assumptions required for analytical methods of study. With simulation, network performance can be examined over a wide variety of network topologies, traffic loads and operating conditions to derive results where analysis is unwieldly or not feasible. Simulation can be used to supplement other analytical results, and higher order statistical measures of performance can also be obtained.

The computer modeling activity is more efficient if a problem–oriented language is used rather than a general purpose algorithmic language like Fortran. We have successfully used SIMSCRIPT for a number of telecommunications network simulation projects [9]. SIMSCRIPT is a language comparable in power to Fortran, but with added list processing (queue handling) and discrete event simulation capabilities. It is English–like in appearance and contains semantics whose world–view assists the person modeling; the programming is done in a language closer to the concepts involved. SIMSCRIPT contains semantics (statement types) that closely fit the concepts in communications network simulation. The dynamics of these networks usually involve several mutually related components (for example, nodes and links) working in a concurrent asynchronous fashion. The ability to model this functioning is provided by the software mechanisms in SIMSCRIPT. It makes the state changing events happen at the proper time. It provides context switching from "process to process" without programmer intervention. Creating the optimum connection between the language element and an element in the model is the real art in modeling in SIMSCRIPT.

Various network components (their activities) are modelled as SIMSCRIPT "processes" which control the movement of the messages (packets) around the network. Usually, several copies (instances) of these processes are necessary, and the language gives an individual the ability to replicate these while specifying only one prototype. As the number of these problem elements is usually not known *a priori*, dynamic storage allocation/ deallocation is provided. Elements of the traffic (calls, messages, packets) can be created and destroyed at will. The modeling of random phenomena is assisted by the existence of the most commonly used random distributions. SIMSCRIPT provides software mechanisms for the collection of statistics as the simulation proceeds over time. This would

include performance measures such as throughput and delay. Node and Link utilizations are easily collected using these language features. A complete description of the language is contained in [10].

SIMSCRIPT is available at Rockwell on the VAX 11/780, IBM/S370 and the IBM PC/XT. In addition, a graphics package has been developed specifically to display the simulation results on Tektronix terminals. SIMSCRIPT is the commercial product of CACI, Inc. and is widely used and accepted in government and industry. The language is the same for each host computer; therefore it is highly portable. The English-like appearance, world-view, a set of dedicated statements and associated software for discrete-event simulation result in a dramatic increase in model building efficiency over other algorithmic languages. Our experience at Rockwell shows that SIMSCRIPT requires much less time for model building compared with Fortran.

4. SIMULATION OF THE TOKEN PASSING BUS

Each machine is modeled as a SIMSCRIPT "process" that operates asynchronously with each other. A message generation mechanism, located at each station, fills 4 levels of priority queues which act as sources for the traffic. Exponential distributions are used for inter-frame intervals and for selecting the length of each frame. A number of parameters such as number of stations, initial values for the target token rotation timers for each queue, mean inter-frame interval of the message generation distribution of each queue, mean length of the frame for each queue, maximum propagation delay, and bus transmission rate are all user–selectable. These parameters can be incorporated by changing a data set that otherwise functions as a set of default values. Fig. 2 summarizes the various functions incorporated in the simulator. A frame process is used to simulate the signal flow on the bus. Each frame transmitted on the bus 'wakes up' all the receivers on the bus. Each frame carries a number of attributes such as time of generation, time of leaving the queue, length of the frame, type of frame (data or control), time of reception at the destination receiver, source address and destination address. The model is developed in an incremental mode, in a gradual manner such that adherence to the protocol specification can be assured by 'walking through' the code at every stage of the development. For example, alternate transmission of a single data frame and a token frame is realized initially for a single queue, followed by step–by–step introduction of various features such as multiple queues and message generation processes, variable frame lengths, target rotation and token hold timers, and so on. An initialization 'routine' provides input on the network parameters and initially activates different processes in proper order. A statistics extraction routine gathers data throughout the simulation run and provides the output at the end of the run. The output data contain an array of statistics on number frames generated, number of frames transmitted, throughput efficiency, and mean queueing delay; for each queue, station and for the network as a whole. Results obtained from some typical simulation runs are discussed in the following section.

5. PERFORMANCE RESULTS FROM SIMULATION RUNS

In the investigations conducted using the simulator to understand the behavior of the priority scheme, throughput and average delays are measured for different initial values for the timers and for different mean inter-frame intervals for the frame generation distributions that feed the queues. Table I shows the typical output for a given station that includes statistics within a particular rotation and the cumulative statistics up to that rotation starting from the first rotation. The results are shown in Figs. 3 and 4. In Fig.3, which gives results for a case in which frame generation at each queue is at a fast pace such that there is no starvation of traffic, it can be seen that the timers should be given a minimum value before a significant improvement in throughput efficiency can be realized. This is due to the presence of some fixed number of high priority frames that should be transmitted by each station. Fig. 4 shows the performance with different mean inter–arrival times between frames for the exponential frame generation distribution (a seperate run is made for each mean value). As the mean value increases, the offered traffic at each queue decreases and the throughput efficiency, ratio of transmitted data octets to the generated (offered) data octets, increases. However, the average delay per frame does not fall monotonically. One of the queues at each station is required to transmit on high priority in each rotation, four frames if available. As the traffic generation slows down, a point is reached at which the required four frames are not available at this queue; thus the token is passed more frequently to lower priority queues in which frames with large queueing delays are waiting. This caused the early dip and the subsequent increase in the delay curve. As the traffic generation slows down further due to the higher mean interarrival times, the queueing delays fall substantially leading to the eventual downward trend in the average delay curve.

CONCLUSIONS

The results indicate that the relationships between desired priority levels and the assignment of initial values to the timers of various queues are not easy to determine, since any change in any given timer value affects the throughput and delays of all the queues. This relationship becomes more difficult to determine if the frame generation processes are not identical. Some constraint values on throughput and delay must be specified as guidelines to aid in the selection of initial values for the timers.

ACKNOWLEDGEMENTS

The authors wish to thank their colleagues W.F. Hall, A.P. Andrews, and K.W. Fertig, Jr., of the Rockwell International Science Center and J.F. Mesmer of the Rockwell International Space Station Systems Division, for their keen interest and support throughout the work.

REFERENCES

[1]. IEEE Project 802, Local Area Network Standards, "Token-passing Bus Access Method and Physical Layer Specifications," Draft IEEE Standard 802.4, Draft E, July 1983.

[2]. International Standards Organization (ISO), "Information Processing Systems – Open Systems Interconnection – Basic Reference Model," Draft International Standard ISO/DIS 7498, 1982.

[3]. W. Bux, "Local Area Sub–networks: A Performance Comparison," IEEE Trans. Commun., pp. 1465–1473, October 1981.

[4]. W. Stallings, "Local Network Performance," IEEE Communications Magazine, Vol. 22, pp. 27–36, February 1984.

[5]. B. Stuck, "Calculating the Maximum Mean Data Rate in Local Area Networks," IEEE Computer, pp. 72–76, May 1983.

[6]. T.L. Phinney and G.D. Jelatis, "Error Handling in the IEEE 802.4 Token–Passing Bus LAN," IEEE Sel. Areas in Commun., Vol. SAC–1, pp.784–789, November 1983.

[7]. S.K. Rahimi and G.D. Jelatis, "LAN Protocol Validation and Evaluation," IEEE Sel. Areas in Commun., Vol. SAC–1, pp. 790–802, November 1983.

[8]. A.R.K. Sastry, "Maximum Mean Data Rate in a Local Area Network with a Specified Maximum Source Message Load," Conf. Rec., IEEE INFOCOM, pp. 216–221, March 1985.

[9]. M.W. Atkinson, "Network Simulation Using SIMSCRIPT," Conf. Rec., IEEE GLOBECOM, Atlanta, Nov. 1984.

[10]. *Simscript II.5 Programming Language,* CACI, Inc., 1983.

Table I. Typical output for a given station and for the network at the end of a simulation run (number of stations: 5)

QUEUE EXRACTION REPORT FOR STATION NO 4

	QUE TOKEN HOLD TIME	NO. DATA FRA TRAN'D	REMAINING QUE SIZE	ACTUAL TOKEN ROTATION TIME
QUE NO. 4	2.00	1	2	30835.00
QUE NO. 3	7311.00	2	0	30837.00
QUE NO. 2	11072.00	2	1	38148.00
QUE NO. 1	8204.00	1	2	49220.00

CUMMULATIVE STATISTICS OF THROUGHPUT AND DELAY AT TIME = 89248.00 OCTETS
FOR STATION AND EACH QUEUE WITHIN STATION

STATION 4 STATISTICS

TOTAL DELAY	254079.17	TOTAL QUEUE DELAY	226497.17	TOTAL THRUPUT	0.52
SUM FRAMES GENERATED	11	SUM OCTETS GENERATED	51228		
SUM FRAMES X'MITED	6	SUM OCTETS X'MITED	26982		
SUM.FRAMES REC'VD	6				
AVG DELAY	42346.53	AVG QUEUE DELAY	37749.53		

QUEUE STATISTICS FOR STATION 4

	QUE # 4	QUE # 3	QUE # 2	QUE # 1
SUM TOTAL DELAY	35402.00	74844.97	85913.20	57919.00
AVG TOTAL DELAY	35402.00	37422.48	42956.60	57919.00
SUM QUE DELAY	30837.00	64331.97	73904.20	57424.00
AVG QUE DELAY	30837.00	32165.98	36952.10	57424.00

CUMULATIVE STATISTICS

SUM FRAMES GENR'TD	3	2	3	3
SUM OCTETS GENR'TD	15256	10229	14138	11605
SUM FRAMES TRAM'TD	1	2	2	1
SUM OCTETS TRAM'TD	4465	10313	11809	395
THRUPUT EFFICIENCY	0.290	1.000	0.829	0.030

ROTATION STATISTICS

SUM FRAMES GENERATED	3	2	3	3

COMPOSITE NETWORK STATISTICS: ALL STATIONS, ALL QUEUES

TOTAL DELAY	782321.38
TOTAL QDELAY	701921.38
TOTAL THRUPUT	0.35
TOTAL FRAMES GENERATED	60
TOTAL OCTETS GENERATED	221887
TOTAL FRAMES TRANSMITTED	21
TOTAL FRAMES RECEIVED	21
TOTAL OCTETS TRANSMITTED	78300
AVERAGE TOTAL DELAY	37253.40
AVERAGE QUEUE DELAY	33424.83
BUS UTILIZATION	99.14 PERCENT

FINAL CUMULATIVE STATISTICS
SIMULATION ENDED NORMALLY AT TIME = 89248.00
INITIAL TOKEN HOLDER WAS STATION 5 WHICH PASSED TOKEN 1 TIMES
RESULTING IN an AVG. ROTATION TIME = 89248.00

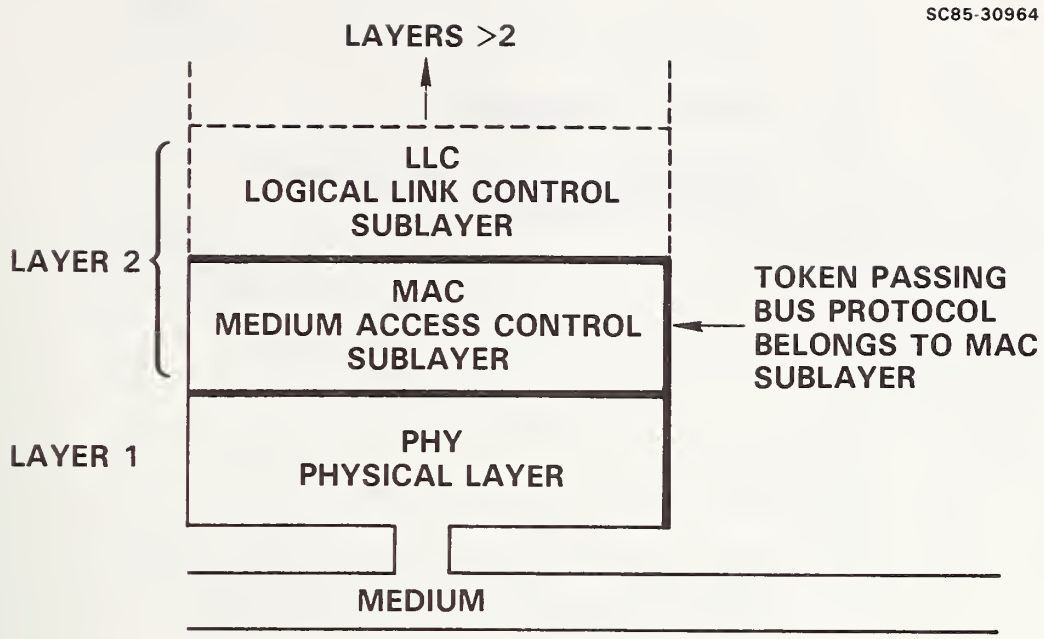


Fig. 1. Relationship of the token-passing bus protocol to the layers in the OSI model.

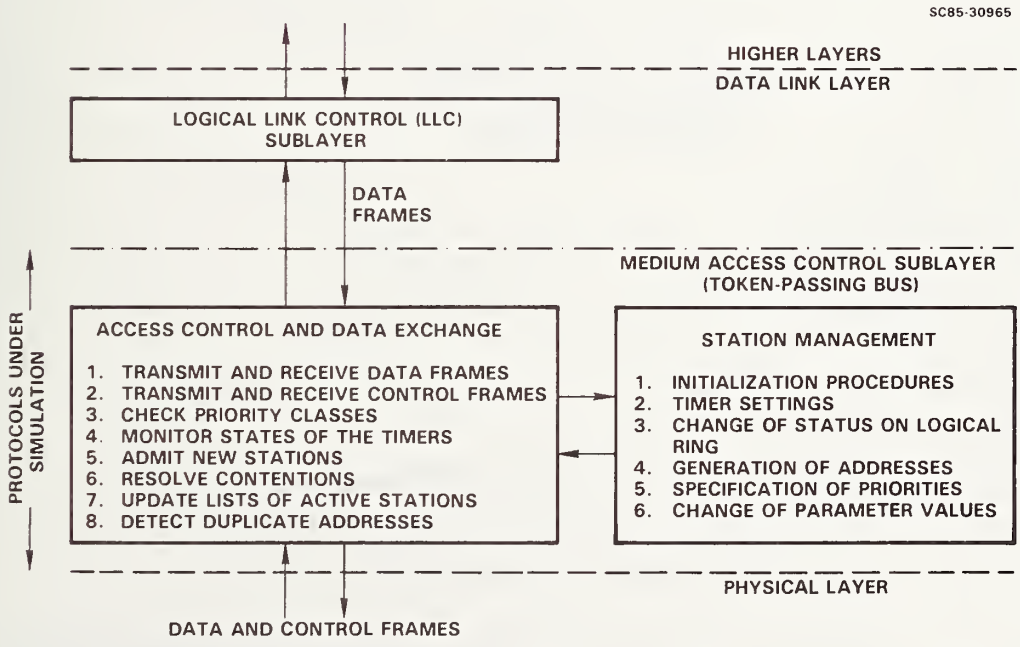


Fig. 2. Functions of the token-passing bus medium access protocol.

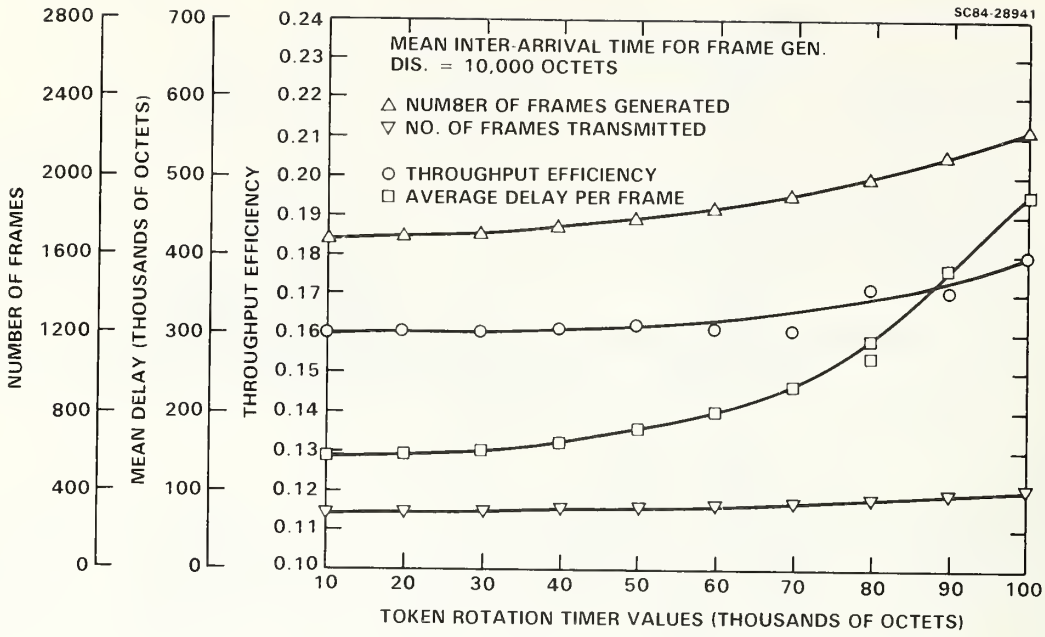


Fig. 3. Performance of the token passing bus as a function of the initial values of the target token rotation timers (timers of all the queues are assigned the same values in this example).

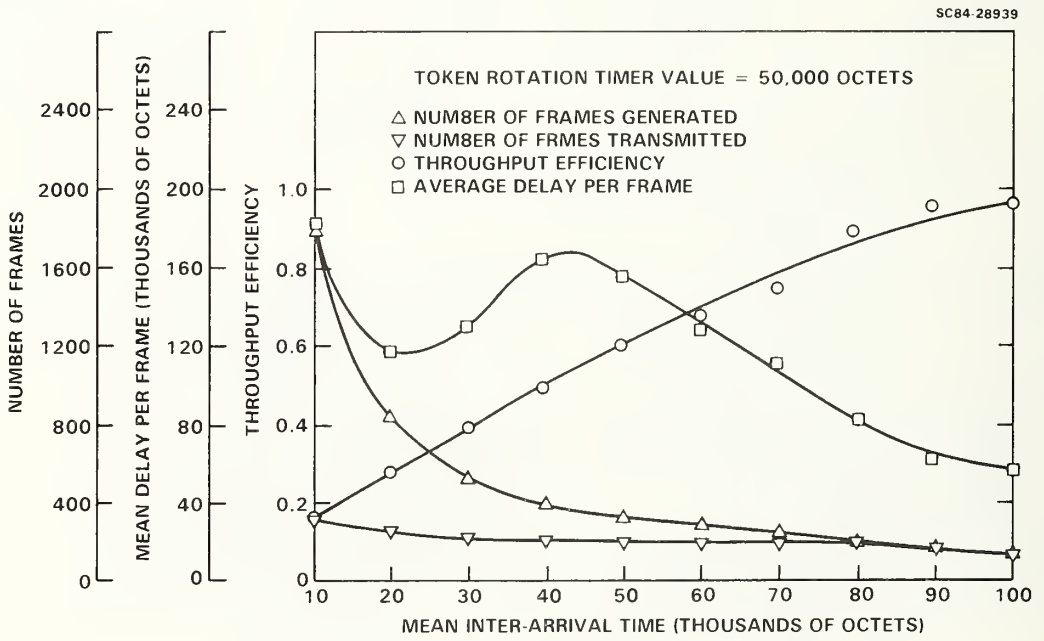


Fig. 4. Performance of the token passing bus as a function of mean inter-arrival frame times of the frame generation distribution.

Session II:

Verification and Compliance Testing

Chairperson: K. H. Muralidhar
Industrial Technology Institute

Token Bus (IEEE Std. 802.4) Network Simulator

Orly Kremien
Motorola Semiconductor Israel Ltd.
147 Bialik Street
Ramat-Gan 52523, Israel

ABSTRACT

The Token Bus Network Simulator (TBNS), developed by Motorola Semiconductor Israel (MSIL), is a software tool which aids in Token-Bus (IEEE 802.4) protocol development, verification and performance evaluation. It is a discrete event-driven simulator that is coded in PASCAL and provides predictions of delay, throughput and many other performance measures as a function of offered load. The simulator implements the IEEE 802.4 (Rev. A, 1984) Token-Passing Bus Medium Access Control (MAC) Specification of protocols for local area networks. It models token-bus network behaviour in batch mode and under interactive user control. The simulator can trace the progress through the network of each message/event to facilitate model validation and analysis. Use of the simulator at MSIL has resulted in the discovery of several protocol errors (including one deadlock situation) which were reported back to the IEEE 802.4 committee.

OBJECTIVES FOR TBNS DEVELOPMENT

Several objectives led to the development of TBNS :

- Token-Bus protocol verification - after the definition phase of a Token-Bus Controller (TBC) VLSI chip and a thorough study of the IEEE 802.4 protocol, we felt that some of the mechanisms suggested might not work. We wanted to find deadlocks and inefficiencies of the protocol, if such existed.

- Token-Bus Network performance evaluation - to gain better understanding and deeper insight into token-bus networks in general, and TBC based networks in particular. From this study we were able to identify sensitivities of a Token-Bus network to configuration and protocol parameters.

- Token-Bus chip development - verification of micro-code and logic implementation by integration of a VLSI simulator with TBNS.

- Networking software development to support Token-Bus networks - verification of software functions (e.g. network management) in a network environment.

- Simulation aided configuration and tuning of token-bus networks.

TBNS IMPLEMENTATION

The modular design of TBNS provides ease of ennancement and ease of adaptation to future standards revisions. The main modules of the simulator are :

- user interface

- network configuration algorithms

- node (chip) specification description

- simulator management routines

USER INTERFACE

An enhanced, menu-driven user interface (with an elaborate debugger) allows for interactive network configuration, simulation control (define stopping criterion), store/restore system image, halt simulation/resume run, display/set station variables, display/set management variables, presentation of system state, presentation of statistics, trace simulation (scheduler, control-program, ACM actions, TxM, RxM), and also has an option to force "nasty" events (introduce artifical errors).

For a specific simulation run, the user describes the network topology and node types. After initialization the user defines a stopping criterion (reach steady-state, transmit N frames, etc.). When the run is completed the user will have all performance measures available (queuing delay, throughput, protocol statistics etc.). For performance evaluation the simulator is first run until steady-state is reached, statistics are then reset and following the reset the simulator runs until the stopping criterion is met.

NETWORK CONFIGURATION ALGORITHMS

Network topology is described by physical characteristics, such as cable length, cable type (baseband, broadband),data-rate, propagation-time and node distribution along the cable. Node types define node characteristics in terms of offered load (by a station of this type), frame length distribution, node options (e.g. PROWAY) and parameter setting, and other node attributes. Each node type may describe one or more nodes. Address assignment and node allocation along the cable are either random or user specified.

NODE (CHIP) SPECIFICATION DESCRIPTION

The simulation model of the MAC sublayer implements the interface machine (IFM), the access control machine (ACM), and receive and transmit machines (RxM, TxM). Propagation delay, frame transmission time and frame collisions are taken into account. Station delay which results from system bus latency is included in the model (some physical characteristics such as noise are excluded from the model).

ACCESS CONTROL MACHINE (ACM)

The ACM, which is the heart of the MAC sublayer, is the most complex and important machine. IEEE 802.4 (part 7) specifies the ACM using a formal extended state transition model. This formal model was directly transformed into an extended state machine coded in PASCAL. The ACM is described by the ACM state variable (OFFLINE, IDLE, etc.) for each station, state routines and events for scheduling state routines. Conditions are checked by state routines and the appropriate action routine is entered. Action routines usually trigger scheduling of other events.

INTERFACE MACHINE (IFM)

The IFM is represented by four input frame queues for each node, and procedures to access (add/get a frame) these queues. The LLC sublayer and all higher layers are represented by the traffic generator which is part of the IFM.

RECEIVE AND TRANSMIT MACHINES

The receive machine is responsible for frame reception and detection of changes in the bus state (as perceived by this station). The receive machine is modeled by a start_of_reception event, and an end_of_reception event. It updates the relevant state variables (bus_quiet, noise_burst, Rx_protocol_frame, Rx_data_frame) accordingly. It should be noted that one station can detect bus_quiet at the same time as another station detects bus_busy. The model also accounts for frame collision (superposition of signals). Noise burst might also result from frame arrival to a station while this station is in transmit mode. Notice that if only the preamble part of a frame collides with another frame, the first frame will result in a valid frame reception and the latter in noise burst.

The transmit machine is modeled by a start_of_transmission event, end_of_transmission event and transmit_mode state variable. Frame transmission time as well as propagation delay between the transmitting station and the receiving station are both taken into account.

SIMULATOR MANAGEMENT

The network simulator is a discrete, event-driven simulator. All events generated during a simulation run reside in a linked list (non-linear). Emphasis is placed on simulator efficiency to allow for fast simulation of medium and large networks at various loads. Each event has several attributes; the main two attributes are event (future) schedule time and event-type. Sophisticated algorithms (of log N complexity scheduling time, with N events present in the list) are used for event-list handling. Events in the Event List are sorted by their scheduled time. Events are either unconditional, in which case their scheduled time will never change, or conditional, in which case they might be rescheduled as a result of a change in the state of the network. All events, conditional and unconditional, reside in one list. The main simulator management routines are :

- simulator control and event handling routines

  The simulation control program always selects the first event to be executed and the simulation clock is advanced accordingly. The event type is checked and the appropriate routine is executed. Following the execution of that routine, usually one or more events are scheduled.

- traffic generation modeling (Workload modeling)

A traffic generator (which is part of the IFM) generates data frames of varying size which arrive at various rates (frame length and frame rate distributions are specified for each node type during simulation initialization).

- statistics collection and analysis

  Statistical information is gathered during a simulation run and analyzed upon user request.

TBNS USAGE

USING SIMULATION FOR TOKEN-BUS PROTOCOL VERIFICATION

One of the token-bus network simulator main objectives is to aid in protocol verification. Two methods are used for protocol verification :

run test cases - with this method, the expected protocol behavior under both normal and abnormal conditions is tested. In order to prove the correctness of the token-bus model, we formulate assertions which reflect the desired correctness properties. Some of these assertions are taken from the literature and others from a thorough study of the protocol. For each of these assertions we have to show that the protocol for each entity satisfy the high-level assertion.

analyze statistical measures - statistical information might reveal unexpected behavior of the protocol (e.g. blocked stations). Different traffic loads, operating conditions, parameter settings and configurations are observed, and statistical measurements implemented in the simulator help with the protocol verification task. Using this method, we discovered a deadlock situation - a combination (which is not rare) of protocol parameter settings and load might lead into a situation where stations are not allowed into the logical ring.

TOKEN-BUS PERFORMANCE

To characterize token-bus network performance, a series of measurements obtained from token-bus simulator runs are presented and interpreted. From these measurements we are able to characterize the effects of various protocol parameter settings and hence recommend settings to avoid

certain possible anomalies. Dependence on topology, number of stations, cable-length and other parameters is also characterized.

We have completed our first set of experiments aimed at performance evaluation. One of the results we have from these experiments is that the token-bus network is not sensitive to node address allocation. These results are described in detail in a separate article.

TOKEN-BUS CHIP DEVELOPMENT

An interface to a high-level chip-simulator was added for network-simulator/chip-simulator integration. The TBC chip designers use this tool to debug the micro-code in a network environment. They are able to run a large amount of test cases and to get chip performance statistics, point out performance bottlenecks, and use this information to optimize TBC design.

NETWORKING SOFTWARE DEVELOPMENT TO SUPPORT TOKEN-BUS NETWORKS

Using simulation run results, considerable insight into token-bus network management problems is acquired. Network management functions (distributed and centralized) which are unique to token-bus are identified, and models of these functions are developed. Simulation is then used to check the effect on the system of these functions, and those which give a positive effect will be selected for implementation in a token-bus network.

CONCLUDING REMARKS

The Token-Bus Network Simulator proved to be a valuable tool for protocol development. The simulator provides us with an accurate model of a real network which is impossible to model analytically. The modular design enables us to modify our model easily whenever there is a revision of the token-bus protocol or a new feature of its VLSI implementation. We recommend this approach for protocol development projects.

ACKNOWLEDGMENTS

I would like to thank D. Kolton for his valuable assistance with the implementation, debugging and running of TBNS.

BIBLIOGRAPHY

1. IEEE Standard 802.4, Rev. A, 1984.

2. Rahimi, S. K. Jelatis, G. D.
LAN Protocol Validation and Evaluation
IEEE Journal on Selected Areas in Communications, November 1983

3. Phinney, T. L. Jelatis, G. D.
Error Handling in the IEEE 802 Token-Passing Bus LAN
IEEE Journal on Selected Areas in Communications, November 1983

4. McCormack, W.M. Sargent, R. G.
Analysis of Future Event Set Algorithms for Discrete Event Simulation
Communications of the ACM, December 1981

PERFORMABILITY MODELING TOOLS

J. F. Meyer

Communications and Network Laboratory
Industrial Technology Institute
Ann Arbor, MI

Abstract

Methods/tools for modeling performability (unified performance-reliability) are described with application to the evaluation of real-time local area networks. Emphasis is placed on the use of stochastic activity networks (SANs), where the presentation includes precise definitions of a SAN and associated concepts. Construction of SAN-based performability models is then discussed and the use of this procedure is illustrated in the modeling of a local area network with timing constraints.

1. INTRODUCTION

Complex systems require correspondingly powerful modeling tools to effectively determine needed information concerning a system's ability to perform (performability) in the presence of faults. Local area networks and, particularly, IEEE 802.4 Token Bus networks are no exceptions in this regard.

A new class of probabilistic network models, called *stochastic activity networks* (SANs) [1, 2] , has been developed for this purpose via work at both the Industrial Technology Institute (Communications and Network Laboratory) and The University of Michigan (Computing Research Laboratory). SANs, which are generalized versions of stochastic Petri nets, permit the representation of concurrency, timeliness, fault tolerance, and degradable performance in a single model. Given a SAN model of the system

in question, it is possible to determine, either by analysis or simulation, the probabilistic nature of the SAN's state-activity behavior. This information, in turn, can serve as the basis for evaluating system performability.

The key to successful application of this methodology is the development of software tools which can aid the process of model construction and can fully automate performability solutions. As might be expected, no single solution technique can provide the support necessary for an arbitrary system. Both analytic and simulation techniques, along with appropriate model decomposition methods are necessary and, accordingly, tools must be developed to meet each of these needs. Simulation can be done directly at the network level, while analytic solutions require derivation of higher level state-transition models called *stochastic activity systems* (SASs) [2] . In this case, model construction includes determination of the SAS that corresponds to an underlying SAN, followed by an appropriate characterization of the SAS's state-activity behavior. This phase of model construction must likewise be automated since the SAS corresponding to a realistic SAN can have thousands of states. However, one of the principal features of SANs is the fact that higher level representations of their behavior can indeed be derived automatically. Moreover, unlike queueing networks for example, all the information required to do this is provided directly by the syntax of a SAN model.

Following a brief discussion of the basic concepts that underly performability evaluation, this paper reviews the definition of SANs and describes the procedure by which SANs and their corresponding SASs are used to model performability. Although the focus here is on analytic methods/tools, we are also in the process of developing a simulation package to augment existing analytic capability. In the concluding section, we illustrate an application to LANs and, specifically, a LAN that is subject to real-time

constraints on the timeliness of message passing.

2. PERFORMABILITY EVALUATION

Performability is a generally defined concept which unifies usual notions of performance and reliability and includes each as special cases. As argued in [3, 4] , the need for this unified view arises when system performance is degradable in the presence of structural change due to faults. Although prior familiarity with performability modeling is useful in what follows, a brief review of basic concepts and terminology should suffice for anyone familiar with probabilistic models.

Given a system S (interpreted as including not only a system, per se, but also relevant aspects of its environment), the *performance* of S over a specified *utilization period* T is a random variable Y taking values in a set A. Elements of A are the *accomplishment levels* (performance outcomes) to be distinguished in the evaluation process. The *performability* of S is the probability measure $Perf$ (denoted p_S in [3, 4]) induced by Y where, for any measurable set B of accomplishment levels ($B \subseteq A$), $Perf\,(B)$ is the probability that S performs at a level in B. Solution of performability is based on an underlying stochastic process X, called a *base model* of S, which represents the dynamics of the system's structure, internal state, and environment during utilization. A base model X together with a performance variable Y is a *performability model* of S. (We are omitting some details here regarding how Y must relate to X, resulting in a slight departure from the definition given in [4]). Performability model *construction* is the process of identifying a performance variable Y and determining a base model stochastic process X that permits a solution of performability. Performability model *solution* is the process of obtaining performability values $Perf\,(B)$ for

accomplishment sets B that are of interest to the user. Generally, knowledge of the probability distribution function (PDF) of Y suffices to determine such values.

As reviewed in [5] , most of the work to date on performability evaluation has focused on accommodating properties of fault tolerance and degradable performance. However, in the case of distributed real-time systems, and particularly real-time LANs, properties of concurrency and timeliness can likewise influence a system's ability to perform. Accordingly, when such evaluations are model-based (on either analytic models or simulation models), all four properties need to be dealt with effectively in the modeling process.

During the past 20 years, considerable effort has been devoted to the modeling of concurrent systems for the purpose of behavior analysis and, to a lesser extent, performance evaluation. Much of this activity has been directed toward behavior analysis in a nonprobabilistic setting through the use of Petri nets, derivatives thereof, and other model types which are behaviorally equivalent to Petri nets (see [6] for a comprehensive discussion of such models). Efforts have also been made to extend Petri-type nets, via the introduction of timing, to obtain models that are better suited to performance evaluation [7, 8, 9, 10, 11, 12] . Regarding probabilistic models that deal with concurrency, one might legitimately include queueing networks (as surveyed, for example, in [13] but, typically, such models treat concurrency at much higher (less detailed) levels than do Petri-type models. Of greater relevance are probabilistic network models that capture concurrency at lower levels. The latter include GERT Networks [14] and General Activity Networks [15] that date back to 1964, along with more recent models which are direct probabilistic extensions of Petri nets [16, 17, 18] .

The background and needs indicated above (see [19] for a more detailed discussion of these considerations), motivated the development of stochastic activity networks, the class of network models referred to briefly in our introductory remarks. These models incorporate features of both queueing networks and stochastic Petri nets and, via an appropriately defined set of network primitives, permit complex "instantaneous activities" to be represented quite simply. Moreover, they indeed appear to be well suited to the representation of concurrency and timeliness as well as fault tolerance and degradable performance.

3. STOCHASTIC ACTIVITY NETWORKS

As defined in [1] , stochastic activity networks (SANs) are probabilistic extensions of activity networks (ANs), where the nature of this extension is similar to the definition of stochastic Petri nets in terms of (ordinary) Petri nets. For a more thorough understanding of what follows, the reader should become familiar with the definitions of both ANs and SANs, as presented in [1] . Briefly, ANs are generalized Petri nets with two types of *activities* ("transitions" in Petri net terminology): *timed activities* (e.g., activities T_1-T_5 of Fig. 2) and *instantaneous activities* (e.g., activities I_1-I_5 of Fig. 2). (It should be noted that the need for such a distinction was independently recognized by Marsan, et al. [20] in their definition of "generalized stochastic Petri nets".) Further generalization is achieved via the association of *cases* with activities (see activities T_4 and T_5 of Fig. 2, for example) and the use of *gates* (e.g., G_1-G_5 of Fig. 2) which permit greater flexibility in describing how an activity is enabled and how the completion of an activity affects the next marking of the network.

Given an AN which is *well-behaved* (see [1]) in some specified initial marking, a SAN is formed by adjoining functions C, F, and G, where C specifies the probability distribution of case selections, F represents the probability distribution functions of activity times, and G describes the sets of "reactivation markings." Relative to the definition presented in [1] , the introduction of reactivation markings (via G) is new. Moreover, since the concept of a SAN is central to the discussion that follows, it is appropriate to define them in greater detail.

3.1. Model Definition

Definition 2.1: A *stochastic activity network* is a structure (M, μ_0, C, F, G) where:

i) M is an activity network with a set of places P consisting of n places.

ii) μ_0 is the *initial marking,* a stable marking in which M is well-behaved.

iii) C is the *case distribution assignment,* an assignment of functions to activities such that for any activity a, $C_a : N^n \times X_a \rightarrow [0,1]$, where N is the set of natural numbers and X_a is the set of cases of activity a. Furthermore, for any marking $\mu \epsilon N^n$ and activity a which is enabled in μ, $C_a(\mu, \cdot)$ is a probability distribution called the *case distribution* of activity a in marking μ.

iv) F is the *activity time distribution function assignment,* an assignment of functions to timed activities such that for any timed activity a, $F_a : N^n \times R \rightarrow [0,1]$, where N is the set of natural numbers and R is the set of real numbers. Furthermore, for any stable marking $\mu \epsilon N^n$ and timed activity a which is enabled in μ, $F_a(\mu, \cdot)$ is a probability distribution function called the *activity time distribution function* of activity a in marking μ ; $F_a(\mu, \tau) = 0$ if $\tau \leq 0$.

v) G is the *reactivation function assignment,* an assignment of functions to timed activities such that for any timed activity a, $G_a : N^n \rightarrow P(N^n)$, where N is the set of natural numbers and $P(N^n)$ denotes the power set of N^n. Furthermore, for any stable marking $\mu \epsilon N^n$ and timed activity a which is enabled in μ, $G_a(\mu, \cdot)$ is a set of markings called the *reactivation markings* of a in μ.

A stochastic activity network is represented graphically by associating expressions inside brackets and braces with the graphical representation of an activity network. Expressions inside brackets "[ ]" represent case distributions and are assigned to the cases of the graph. Expressions enclosed by brackets "$< >$" represent activity time distribution functions and are associated with timed activities. Expressions inside braces represent reactivation functions and are assigned to timed activities. When an activity time distribution functions is exponential it can uniquely be described by a rate (the parameter of the exponential distribution function). As a result, in graphical representation of SANs, an exponential activity time distribution is indicated by its characterizing rate (which may be a function of the markings of the network) inside parentheses. Fig. 2 shows the graphical representation of a stochastic activity network using the above conventions.

3.2. Execution of Stochastic Activity Networks

Stochastic activity networks are extensions of activity networks [1] where both temporal uncertainty and spatial uncertainty are specified probabilistically.

Regarding temporal uncertainty, execution of the network proceeds as follows. When instantaneous activities are enabled, they complete instantaneously. Enabled timed activities, on the other hand, will (with probability 1) require a nonzero time to

complete. An enabled timed activity is *activated* at instants of time called "activation times" (defined below). After a timed activity is activated it may complete at an instant of time called a *completion time.* If the activity becomes disabled before it completes it is said to be *aborted.* The difference between a completion time of a timed activity and its last activation time (before that completion time) is an *activity time.* Activation times of timed activities are formally defined as follows.

Definition 2.2: An *activation time* of an enabled timed activity a is any of the following instants of time:

i) a time when a becomes enabled, provided a is previously disabled.

ii) a time when a completes, provided a remains enabled in the next stable marking.

iii) a time when a stable reactivation marking μ' of a in stable marking μ is reached, provided a is enabled in μ' and is previously activated in μ.

The following are assumptions about the temporal uncertainty of stochastic activity networks:

a) Activity times are mutually independent random variables.

b) The distribution function of the activity time of a timed activity a, given that a is activated in stable marking μ, is $F_a(\mu,\cdot)$, where $F_a(\mu,\cdot)$ is the activity time distribution function of a in μ (see Definition 2.1, part iv)).

Regarding spatial uncertainty, the execution proceeds as follows. When an activity a completes in a marking μ, it can only affect its input places and output places. The next marking is determined in two steps as follows.

1) For each input gate of activity a with input function f, the marking of the corresponding input places, represented as a vector μ_1, is changed to a vector μ_1' such that $\mu_1' = f(\mu_1)$. For input places directly connected to the activity this means losing of one token for each of those places.

2) Case c of activity a is chosen with probability $C_a(\mu, c)$, where $C_a(\mu, \cdot)$ is the case distribution of activity a in marking μ (see Definition 2.1, part iii)). Then, for each output gate of activity a for case c which has output function g, the marking of the corresponding output places, represented as a vector μ_2, is changed to a vector μ_2' such that $\mu_2' = g(\mu_2)$. For output places directly connected to the activity this means gaining of one token for each of those places.

4. SAN-BASED PERFORMABILITY EVALUATION

In the case of analytic modeling, SANs are used to determine a stochastic process X that

can serve as the base model of a performability model. In the construction of X from some specified SAN, it is convenient to first derive a corresponding higher level model, called a stochastic activity system (SAS) [2] , which suffices to describe the SAN's state-activity behavior. Model construction is also influenced by the type of solution method employed. In the discussion that follows, we assume that the solution method is of the type introduced in [21] and subsequently refined in a number of recent studies (see [22, 23, 24] , for example). This approach presumes a performability model wherein a fixed "performance rate" is associated with each "structure state". Accordingly, in the process of model construction it is natural to attempt a structure-performance decomposition as early as possible, preferably in the initial SAN model of the system. The

procedure specified and illustrated below is based on this philosophy.

4.1. Model Decomposition

To describe the nature of this decomposition, we introduce the notions of a *performance submodel,* a *structure submodel,* and a set of *common places.* These concepts will enable us to specify necessary and sufficient conditions that a SAN must satisfy if its stochastic behavior is to be described in a manner conforming to "reward model" solution techniques [21, 22, 23, 24] . Central to the decomposition is the notion of a set of structure-related activities and a set of performance-related activities. Given a stochastic activity network, the set of *structure-related activities* is the set A_s containing all activities which represent variations in the system due to a change in system structure. The set of *performance-related activities* is the set A_p containing all activities which represent variations in the internal state and environment of the system, excluding structure-related activities. In terms of A_s and A_p , two submodels are distinguished as follows:

Definition 4.1: Given a stochastic activity network (M, μ_0, C, F, G), where M is an activity network with a set of activities A , a set of places P , a set of input gates I, and a set of output gates O, the *structure submodel* is a stochastic activity network (M', μ_0', C', F', G') where:

1) M' is an activity network consisting of
   a. the set of structure-related activities A_s .
   b. the set P_s of places which are connected to a structure-related activity through a single gate.
   c. the set I_s of input gates which are connected to some structure-related activity.
   d. the set O_s of output gates which are connected to some structure-related activity.

e. interconnections between P_s, A_s, I_s, and O_s as in M.

2) μ_0' is the function μ_0 restricted to P_s.

3) C' is the function C restricted to A_s and structure markings.

4) F' is the function F restricted to A_s and structure markings.

5) G' is the function G restricted to A_s and structure markings.

and the term *structure marking* denotes the marking of M restricted to P_s.

Definition 4.2: Given a stochastic activity network (M,μ_0,C,F,G), where M is an activity network with a set of activities A, a set of places P, a set of input gates I, and a set of output gates O, the *performance submodel* is a stochastic activity network (M',μ_0',C',F',G') where:

1) M' is an activity network consisting of

   a. the set of performance-related activities A_p.

   b. the set P_p of places which are connected to a performance-related activity through a single gate.

   c. the set I_p of input gates which are connected to some performance-related activity.

   d. the set O_p of output gates which are connected to some performance-related activity.

   e. interconnections between P_p, A_p, I_p, and O_p as in M.

2) μ_0' is the function μ_0 restricted to P_p.

3) C' is the function C restricted to A_p and performance markings.

4) F' is the function F restricted to A_p and performance markings.

5) G' is the function G restricted to A_p and performance markings.

and the term *performance marking* denotes the marking of M restricted to P_p.

Definition 4.3: Given an activity network, the set of *common places* P_c is the set $P_p \bigcap P_s$.

4.2. Base Model Construction

In order to construct a base model by the following procedure, the SAN in question must satisfy the following conditions.

1. The SAS realized by the structure submodel and SAS(s) realized by the performance submodel must have a finite set of states.

2. The marking of a common place may not change upon completion of any activity in the performance submodel.

3. No activity $a \in A_s$ may have an activity time distribution function, case distribution, or reactivation function which is dependent on a marking in the set P_p.

4. The set of performance-related timed activities ($A_p \bigcap A_T$) occurs at a sufficiently high rate, as compared to the structure-related timed activities ($A_s \bigcap A_T$), so that, to a good approximation, the performance submodel will reach steady-state conditions between the occurrence of structure-related activities.

When a SAN meets the conditions stated above, base model construction can proceed in the following manner.

Procedure:

1. Construct a stochastic activity network which meets the previously stated conditions.

2. Define the reward rate as a function of the (steady-state) marking of the performance submodel or the steady-state probability distribution of the stochastic state behavior of the performance submodel.

3. Construct the SAS realized by the structure submodel of the SAN in question.

4. For each state of the SAS which corresponds to a distinct marking of the common places in the reachability set of the structure submodel:

a. Construct the SAS realized by the performance submodel of the SAN in question, taking the initial marking of the common places to be their marking in the current state of the structure SAS.

b. Derive the stochastic state behavior of the SAS constructed in a.

c. Compute the reward rate in the current state of the structure SAS from the steady-state stochastic state behavior of the performance submodel.

The state behavior of the structure submodel, together with reward rates determined above constitute a base model of the system in question. Moreover, such base models are suited to performability solution methods of the type described in [21, 22, 23, 24] .

Steps 3 and 4 of this procedure can be automated and software tools for this purpose are currently implemented as part of a larger performability evaluation package known as METAPHOR (Michigan EvaluaTion Aid for PerpHORmability) [22, 25] .

5. LAN EXAMPLE

To illustrate the use of these modeling techniques for performability modeling of real-time local area networks, let us consider the local behavior of a network in a single station which employs a scheduling mechanism referred to as *promptness control* [1] . In short, promptness control is a monitor that rejects a message before it is transmitted if, at the time of a promptness check, the message cannot be received before its deadline. Without promptness control, we assume that the local behavior of the station can be modeled by a queueing system (see [26, 27, 28] for related work using this approach). Standard solution techniques for queueing networks (see [29] , for example) may then be used to solve the model. With promptness control, however, the above local behavior cannot be modeled directly by a queueing network, requiring instead the type of

modeling capability provided by SAN's.

5.1. System Description

The system we consider is a total system $S=(N,E)$ where, informally, the local area network N and environment E can be described as follows. N is a degradable system using M identical channels ($M \geq 1$) for transmitting messages when it is fault-free. Station B whose local behavior is to be modeled has a finite capacity L ($L \geq 0$), that is, B is capable of storing at most L messages. Moreover, there is a promptness control point (pc point) located at the head of the queue in station B (see Fig. 1). The environment E consists of the arrival of messages and their associated real-time requirements. More specific assumptions about N and E are as follows.

5.1.1. Environment Model

Messages arrive at each station of the network randomly. Each incoming message (to a station) is associated with a specified deadline before which it is required to be received at its destination (in order to be on time). Various incoming messages to a station are classified according to their allowed delays (difference between their deadlines and their respective arrival times) such that each class represents a set of messages with the same fixed allowed delay. We assume that the number of such classes is finite and that the arrival of messages within each class is a Poisson process. We also assume that the arrivals of messages of different classes form a set of mutually independent (random) processes.

5.1.2. LAN Model

As depicted in Fig. 1, the fault-free structure of the network uses M identical channels. When a channel fails the system is able to recover (with a specified coverage c) to a degraded configuration with one less channel. However, when there is only one channel left and that channel fails, total system failure will occur. The stations and promptness controls are assumed to be fault-free. Failure of a channel is assumed to occur as a Poisson process with rate of λ. The arrival of messages at station B is also assumed to be a Poisson process with rate of α. Promptness control PC of station B checks any message at the head of the queue of B for promptness whenever the transmission of a message of B is completed. If the message is late it will immediately be rejected from the system; otherwise, the message will get access to a channel and will be transmitted. The channel access time is assumed to be exponentially distributed with mean of $\frac{1}{\mu}$. When a message arrives at its destination it will be checked for promptness. If it is on time it will be received at the station; otherwise, it will be rejected from the system. More specifically, in a structural configuration with i fault-free channels ($0 < i \leq M$), we assume that the local behavior of station B can be modeled (with good approximation) by an M/M/i/i+L queueing system (with FCFS scheduling discipline and) which employs promptness control. Accordingly, an incoming message may be lost either because the station is full (at its arrival) or because it is rejected by promptness control (since it is late). Obviously, when a message is transmitted via a channel it may still be late and miss its deadline during the transmission.

5.2. Performance Variable

We presume that, ideally, the user wants the LAN to successfully transmit all the messages that arrive at station B. However, due to finite capacity of station B, failure of the channels, and deadlines for message transmission, ideal behavior will generally not be attainable. We assume that after an incoming message misses its deadline it will be of no value to the user(s). Accordingly, performance variables such as "throughput" should reflect the rate at which messages are transmitted "on time," i.e., before or at their associated deadlines. To accomplish this in more formal terms, let $T=[0,\ t]$ denote the period during which the system is utilized and consider the following (random) variables.

M_t = number of messages that arrive during T.

SM_t = number of messages that are received on time during T.

Then real-time throughput measures can be defined as follows:

$RTh_t = \dfrac{SM_t}{t}$ is the *real-time throughput* of the system during T.

$NRTh_t = \dfrac{SM_t}{M_t}$ is the *normalized real-time throughput* of the system during T.

In the evaluation that follows, we take $Y_t = NRTh_t$ to be the performance variable Y of the performability model (see Section 2).

5.3. Base Model Construction

Base model construction is a process of determining a stochastic process X which can serve as a base model for performability modeling of the system. Such process, how-

ever, is influenced by the type of solution method employed. In the discussion that follows, we assume that the solution method is of the type introduced in [21] and subsequently refined in a number of recent studies (see [22, 23, 24] , for example). This approach presumes a performability model wherein a fixed "performance rate" is associated with each "structure state". Accordingly, in the process of model construction it is natural to attempt a structure-performance decomposition as early as possible. For the system in question this is accomplished as follows.

The structure configurations of the network N is represented by the set of states $Q_R = \{0,1, \cdots M\}$, where state i represents the number of fault-free channels in the network. (Note that $i=0$ corresponds to system crash.) Using the earlier assumptions about the occurrence of faults in the system, we can model the structure behavior of the network as a Markov process X_R with the set of states Q_R. For each structure state i, the performance aspects of the network in station B can then be modeled (with good approximation) by a M/M/i/i+L queueing system with promptness control. Consequently, we can identify a Markov process $X_{I,i}$ with a set of states $Q_{I,i}$ such that $X_{I,i}$ models the local behavior of station B when the network is in structure state i. Composing the structure related X_R with the performance related $X_{I,i}$, the behavior of the network N in station B is finally modeled as a single Markov process X with state set $Q = \{(i,j) \mid I \epsilon Q_R, j \epsilon Q_{I,i}\}$, where i is a structure state of N and j is the performance related state of N in structure state i. X then serves as a base model for the performability modeling of the system in question.

The above procedure for determining a base model X, however, may be complicated when the capacity of station B increases, resulting in a large state space. For example, for the case of a single fault-free channel and a station with capacity $L=9$,

there will be 2^{10}=1024 states in the state set of X. One way to alleviate this difficulty is to use some models which can represent the system in a natural way and whose state behaviors can serve as a base model X. Provided that there is a procedure for deriving the base model X from the model used, and this procedure is automated, the above process of base model construction is greatly facilitated.

Employing this approach, we have used stochastic activity networks (SANs) (see Section 3) to model the system in question where M=2 and L=4, i.e., for a case where network N has 2 channels and the capacity of station B is 4. It is also assumed that the environment consists of only one class of incoming messages (i.e.,, all incoming messages have a fixed allowed delay). A stochastic activity network corresponding to this case is given in Fig. 2. As shown in Fig. 2, the model consists of two major parts; a *structure submodel* and a *performance submodel,* representing the structure related and performance related parts of the system, respectively. This decomposition facilitates base model construction and model solution (using techniques described in [21, 22, 23, 24]).

In the performance submodel of Fig. 2, the completion of timed activities T_1 and T_2 represent the completion of the transmission of messages of station B by the two channels. Completion of timed activity T_3 represents the arrival of an incoming message at B. The three blocks consisting of instantaneous activities I_3, I_4, and I_5 function similarly; each of these blocks represents the shifting of a waiting message in the station to the next empty buffer stage. Instantaneous activity I_2 represents the rejection of late messages in B (i.e., messages which have missed their deadlines) at pc point PC. This rejection occurs whenever a message of station B completes its transmission and a late message is still waiting at the head of the queue in B. Instantaneous activity I_1 acts like a scheduler assigning messages passed PC to the channels for transmission. Places

P_3, P_4, P_5, and P_6 represent the first, second, third, and forth buffer stages in station B, respectively. Having tokens in any of these places represents a message at the corresponding buffer stage. When there is only one token in one of these places, it means that the corresponding message will be transmitted by a channel (i.e., it will pass PC). On the other hand, when there are two tokens in one of these places, it means that the corresponding message will not be transmitted by any channel and hence will be rejected at PC. Places P_1 and P_2 represent the status of the two channels, which are either available or unavailable to station B. A channel is unavailable to B if it is transmitting a message of B or if it is faulty. Place P_7 represents the number of messages arrived in station B which will be transmitted (i.e., will not be rejected at PC). Place P_8 represents the total number of messages in the system which arrived at B (and are either waiting or being transmitted).

When activity T_3 completes, meaning that a message has arrived at station B, one of two cases may occur when it will be checked for promptness at PC; either it will be on time or it will not. These two possibilities are modeled by two cases of activity T_3, case 1 and case 2, respectively, as shown in Fig. 2. If case 1 occurs, one token is put in place P_6, meaning that a message which will pass PC has just arrived at station B. If case 2 occurs, two tokens are put in place P_6, meaning that a message which will be rejected at PC has just arrived at B. The (previously mentioned) blocks will then shift these tokens accordingly if there is an empty stage in the buffer. This is done more specifically as follows. When a place in the buffer has one token (i.e., representing a message which will pass PC) while the next place is empty (i.e., the next stage is empty), the instantaneous activity of the block between these two places completes immediately and removes the token from the first place and puts it in the next one (i.e.,

the message is immediately shifted to the next empty stage in the buffer). Similarly, when a place in the buffer has two tokens (i.e., representing a message which will be rejected at PC) while the next place is empty (i.e., the next stage is empty), the instantaneous activity between these two places completes immediately and removes the tokens from the first place and puts them in the next one (i.e., the message is immediately shifted to the next empty stage in the buffer).

As for the structure submodel of Fig. 2, completions of timed activity T_4 represents the failure of a channel. Instantaneous activity I_7 represents the occurrence of system failure when both channels have failed. Place P_9 represents the number of fault-free channels. Place P_{10} represents the status of the overall system, either operational or total failure. When a channel fails (via completion of T_4) one of two cases may occur; either the system can successfully reconfigure to an operational state with a single fault-free channel (case 1) or it cannot successfully reconfigure and the system will fail (case 2). When case 1 occurs, the resulting degraded performance is modeled as follows. First, place P_{11} receives one token, denoting that a channel has failed. This change in structure immediately affects the performance submodel via completion of instantaneous activity I_6 and the disabling of timed activity T_2 (representing the failure of a channel). Completion of instantaneous activity I_6 puts one token in place P_2, denoting that a channel has become unavailable to B.

The above describes an activity network model of the system. A stochastic activity network model of the system can be formed by assigning the activity rates (timed activities are assumed to have exponential activity time distributions) to all timed activities and the case distributions to activities T_3 and T_4. The stochastic activity network model of the system in question is shown in Fig. 2, where the activity rate of a timed

activity is given by an expression inside parentheses and the case distribution of an activity is represented as expressions inside brackets associated with the cases of that activity. Note that case probabilities $a(P_7)$ and $c(P_9)$ are functions of the number of tokens in places P_7 and P_9, respectively, and are defined as follows:

$$a(P_7)(k) = Erl(k,t) = 1 - e^{-\mu t} \sum_{i=0}^{k-1} \frac{(\mu t)^i}{i!}$$

$$c(P_9)(k) = 1 - \frac{1}{k}$$

Given this SAN, a base model X is obtained by determining the network's stochastic state behavior. This base model along with the performance variable $Y = Y_t$ defined earlier constitutes a performability model (X, Y) of the system. The solution of this model is discussed in the subsection that follows.

5.4. Model Solution

The solution method used is of the type introduced in [21] and which subsequently refined in a number of recent studies [22, 23, 24] . This approach takes advantage of an important property exhibited by most degradable fault-tolerant systems, namely, that performance related activities occur at much higher rates than do occurrences of faults. Accordingly, in each structure state of the system, the state behavior of the system can be viewed as the long run (steady-state) behavior of the performance related activities. With respect to a performance variable, this implies that there is a fixed (steady-state) performance rate associated with each structure state. Considering such rates as reward rates, the base model X can then be described by a reward model [30] . This reward model is then solved to obtain the probability distribution function (PDF) of the performance variable of the system.

In the SAN model of Fig. 2, the structure submodel is relatively simple and consists of three stable markings representing the three structure states; two fault-free channels, a single fault-free channel, and no fault-free channels (system failure). The solution then proceeds by solving the performance submodel for each of these structure states (see [2] regarding a procedure for this technique). The solution for the case of a system failure is trivial. All performance variables will simply have rate of 0 in this structure state. For the other two structure states, the state behavior of performance submodel will be Markov processes (see [2] for conditions ensuring that the state behavior of a SAN is a Markov process). Standard Markovian solution techniques may then be used to solve these processes. Figs. 3 and 4 show the state-transition-rate diagrams of such Markov processes for the cases of two fault-free channels and a single fault-free channel, respectively. The states of these diagrams are tuples whose elements, from right to left, represent the number of tokens, if any, of the places $P_1, P_2, \ldots, P_6$, respectively. Obtaining the steady-state probabilities of these states, the performance submodels for the above two structure states can analytically be solved. Using such solution method, Fig. 5 displays the performance rate of the network for the case of two fault-free channels, with promptness control and without promptness control, as a function of traffic intensity $\rho = \frac{\alpha}{\mu}$. It is assumed, in this and subsequent figures, that the allowed delay is the mean value of the channel access time, i.e., $D = \frac{1}{\mu}$. Fig. 6 shows the same information but for the case of a single fault-free channel. As is shown, promptness control always improves the performance of the system. Finally, for the case of $\rho = 2$ and the above allowed delays, we use METAPHOR [22, 25] to derive the PDFs of the performance variable Y of the network, with promptness control and without promptness control, for a given set of system parameters as depicted in Fig. 7. As is the case for (strict)

performance in a fixed structure state, we see that promptness control likewise improves the overall performability of the system.

[1] A. Movaghar and J. F. Meyer, "Performability modeling with stochastic activity networks," in *Proc. 1984 Real-Time Systems Symp.*, Austin, TX, Dec. 1984.

[2] J. F. Meyer, A. Movaghar, and W. H. Sanders, "Stochastic activity networks: structure, behavior, and application," ITI Technical Report 84-7, Industrial Technology Institute, Ann Arbor, MI, Dec. 1984.

[3] J. F. Meyer, "On evaluating the performability of degradable computing systems," in *Proc. 1978 Int. Symp. on Fault-Tolerant Computing*, Toulouse, France, June 1978, pp. 44-49.

[4] J. F. Meyer, "On evaluating the performability of degradable computing systems," *IEEE Trans. Comput.*, vol. C-22, pp. 720-731, Aug. 1980.

[5] J. F. Meyer, "Unified performance-reliability evaluation," in *Proc. of the American Control conference*, San Diego, California, June 6-8, 1984.

[6] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[7] G. Nutt, "The formulation and application of evaluation nets," Ph.D Thesis, University of Washington, July 1972.

[8] C. Ramchandani, "Analysis of asynchronous concurrent systems by Petri nets," Ph.D Thesis, Dept of Electrical Engineering, MIT, July 1973.

[9] J. L. Baer and J. Jensen, "Simulation of large parallel systems: Modeling of tasks," in *Measuring, Modeling, and Evaluating Computer Systems*, H. Bellner and E. Gelenbe, Ed. Amsterdam: North-Holland, 1977, pp. 53-73.

[10] Y.W. Han, "Performance evaluation of a digital system using a Petri net-like approach ," in *Proc. of the National Electronic Conf.*, vol. 32, 1978, pp. 155-160.

[11] J. D. Noe, "Nets in modeling and simulation," in *Net Theory and Application, Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 1980.

[12] J. Sifakis, "Performance evaluation of systems using nets," in *Net Theory and Application, Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 1980.

[13] *Computing Surveys (Special Issue on queuing network models of computer systems performance)* , vol. 10, no. 3, Sept. 1978.

[14] A. A. B. Pritsker and W. W. Happ, "GERT: Graphical evaluation and review technique - Part I. fundamentals," *Journal of Industrial Engineering*, vol. 17, no. 5, pp. 267-274, May 1966.

[15] S. E. Elmaghraby, "An algebra for the analysis of generalized activity networks," *Management Science*, vol. 10, pp. 621-631, 1964.

[16] B. Beyaert, G. Florin, P. Lonc, and S. Natkin, "Evaluation of computer system dependability using stochastic Petri nets," in *Proc. 1981 11th Int. Symp. on Fault-Tolerant Computing*, Portland, ME, June 1981, pp. 66-71.

[17] S. Shapiro, "A stochastic Petri net with application to modeling occupancy times for concurrent task systems ," *Networks*, vol. 9, pp. 375-379, 1979.

[18] M. K. Molloy, "Performance analysis using stochastic Petri nets," *IEEE Trans. Comput.*, vol. C-31, pp. 913-917, Sept. 1982.

[19] J. F. Meyer, "Performability modeling of distributed real-time systems," in *Mathematical Computer Performance and Reliability*, G. Iazeolla, P. J. Courtois and A. Hordijk , Ed Amsterdam: North-Holland, 1984.

[20] M. A. Marsan, G. Balbo, and G. Conte, "A class of generalized stochastic Petri nets for performance evaluation of multiprocessor systems," *ACM Trans. on Computer Systems*, vol. 2, no. 2, pp. 93-122, May 1984.

[21] J. F. Meyer, "Closed-form solutions of performability," *IEEE Trans. Comput.*, vol. C-31 , pp. 648-657, July 1982.

[22] D. G. Furchtgott and J. F. Meyer, "A performability solution method for degradable, nonrepairable systems," *IEEE Trans. Comput.*, vol. C-33, June 1984.

[23] L. Donatiello and B. R. Iyer, "Analysis of a composite performance reliability measure for fault tolerant systems," IBM Res. Report RC10325, January 1984.

[24] A. Goyal and A. N. Tantawi, "Evaluation of performability in acyclic Markov chains," IBM Res. Report RC10529, May 1984.

[25] D. G. Furchtgott, "Performability models and solutions," Tech. Report CRL-TR-8-84, Univ. of Michigan, Ann Arbor, MI, Jan. 1984.

[26] A. S. Sethi and T. Saydam, "Performance analysis of token ring local area networks," in *9th Conference on Local Computer Networks*, Minneapolis, Minnesota, October 1984.

[27] M. E. Ulug, "Calculation of waiting times for a real-time token passing bus," GE Research Report, Schenectady, New York, 1984.

[28] M. Marathe and S. Kumar, "Analytical models for an ethernet-like local area network link," in *ACM/Sigmetrics Conference on Measurement and Modeling of Computer Systems*, Las Vegas, NV, Sept. 1981.

[29] L. Kleinrock, *Queueing Systems, Volume I: Theory*. New York, NY: John Wiley, 1975.

[30] R. A. Howard, *Dynamic Probabilistic Systems, Vol II: Semi-Markov and Decision Processes*. New York, NY: Wiley, 1971.

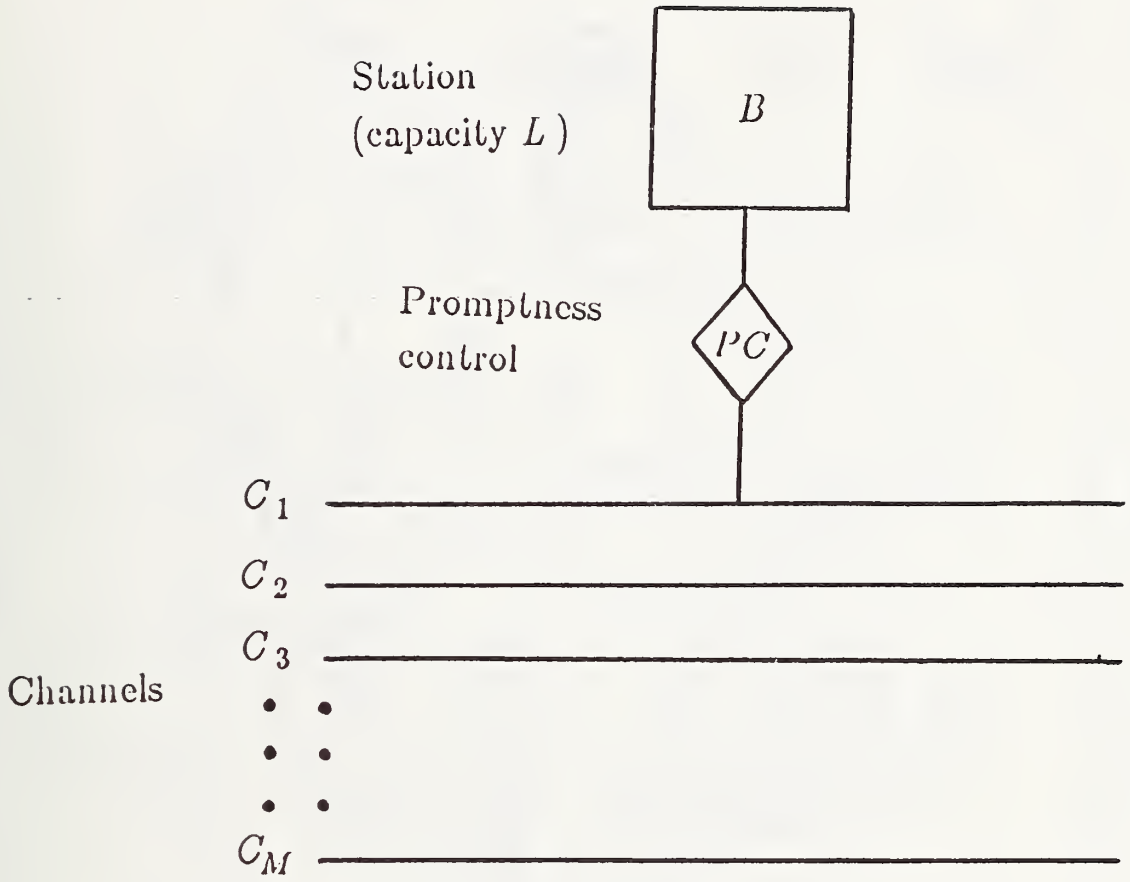
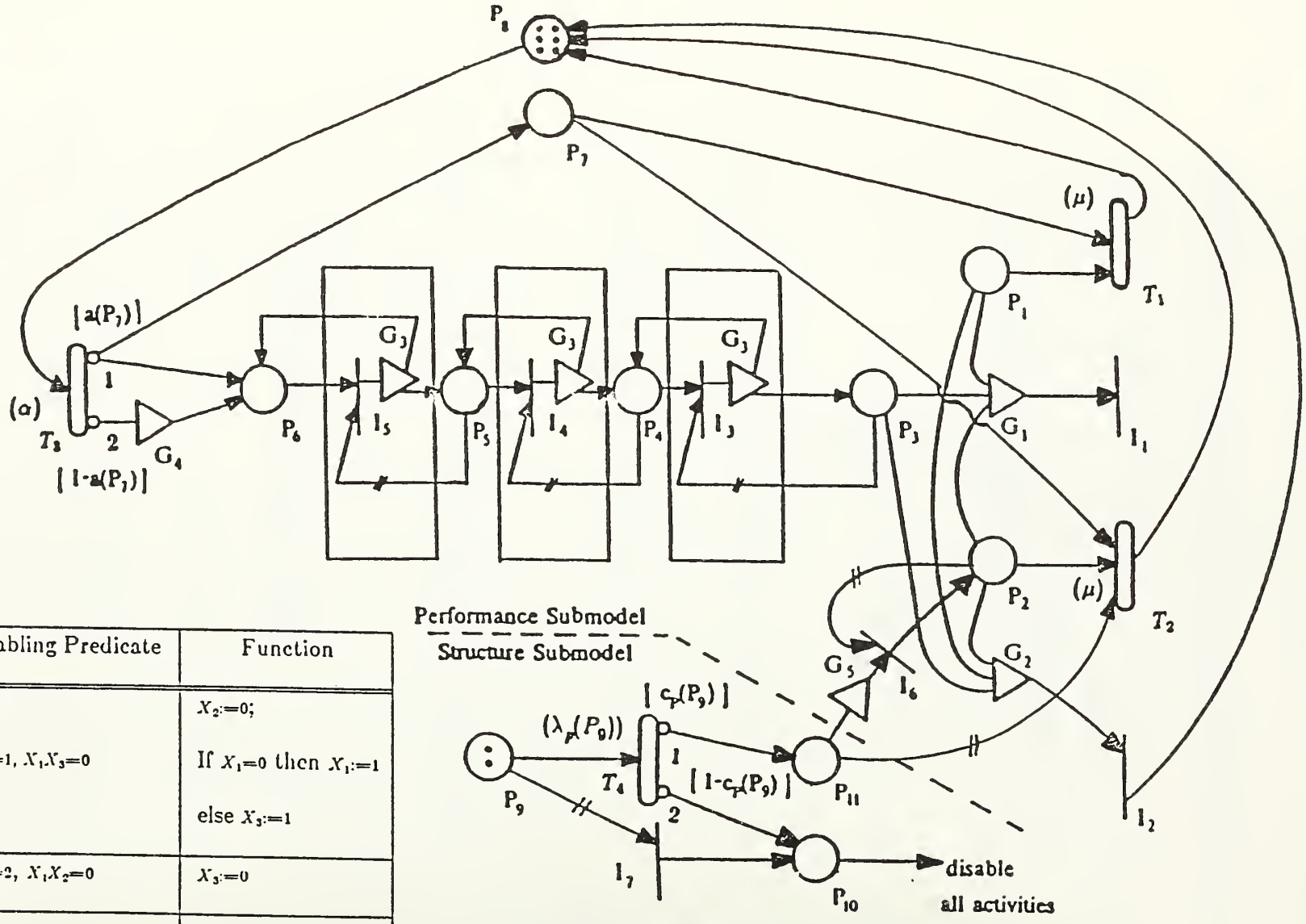


Fig. 1 Block diagram of N.



Gate	Enabling Predicate	Function
G_1	$X_2=1, X_1X_3=0$	$X_2:=0$; If $X_1=0$ then $X_1:=1$ else $X_3:=1$
G_2	$X_3=2, X_1X_2=0$	$X_3:=0$
G_3		$X_2:=X_1+1; X_1:=0$
G_4		$X:=2$
G_5	$X=1$	

Fig. 2 Stochastic activity network model of S.

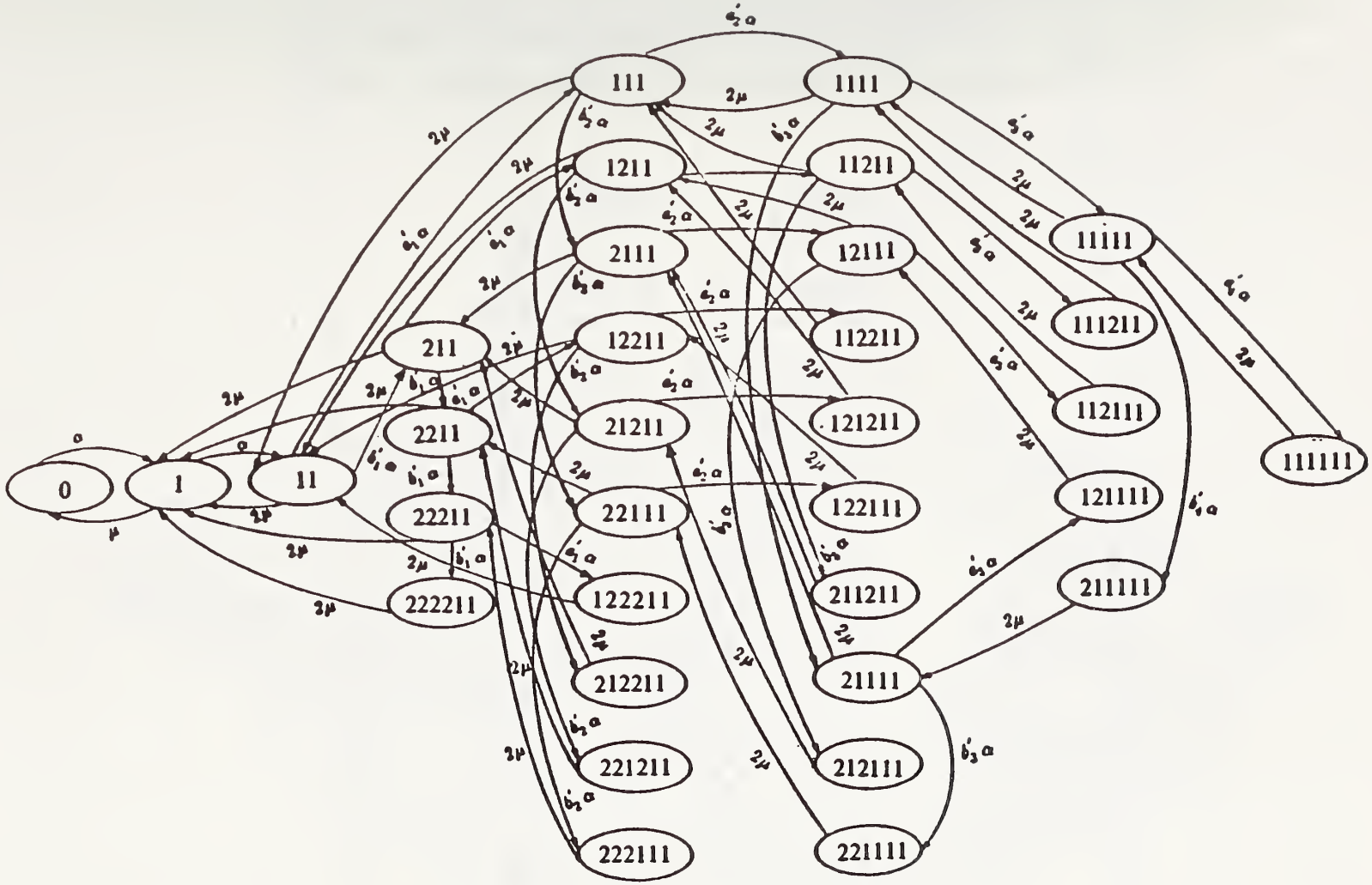


Fig. 3 State-transition-rate diagram of the behavior of S for the case of two fault-free channels.

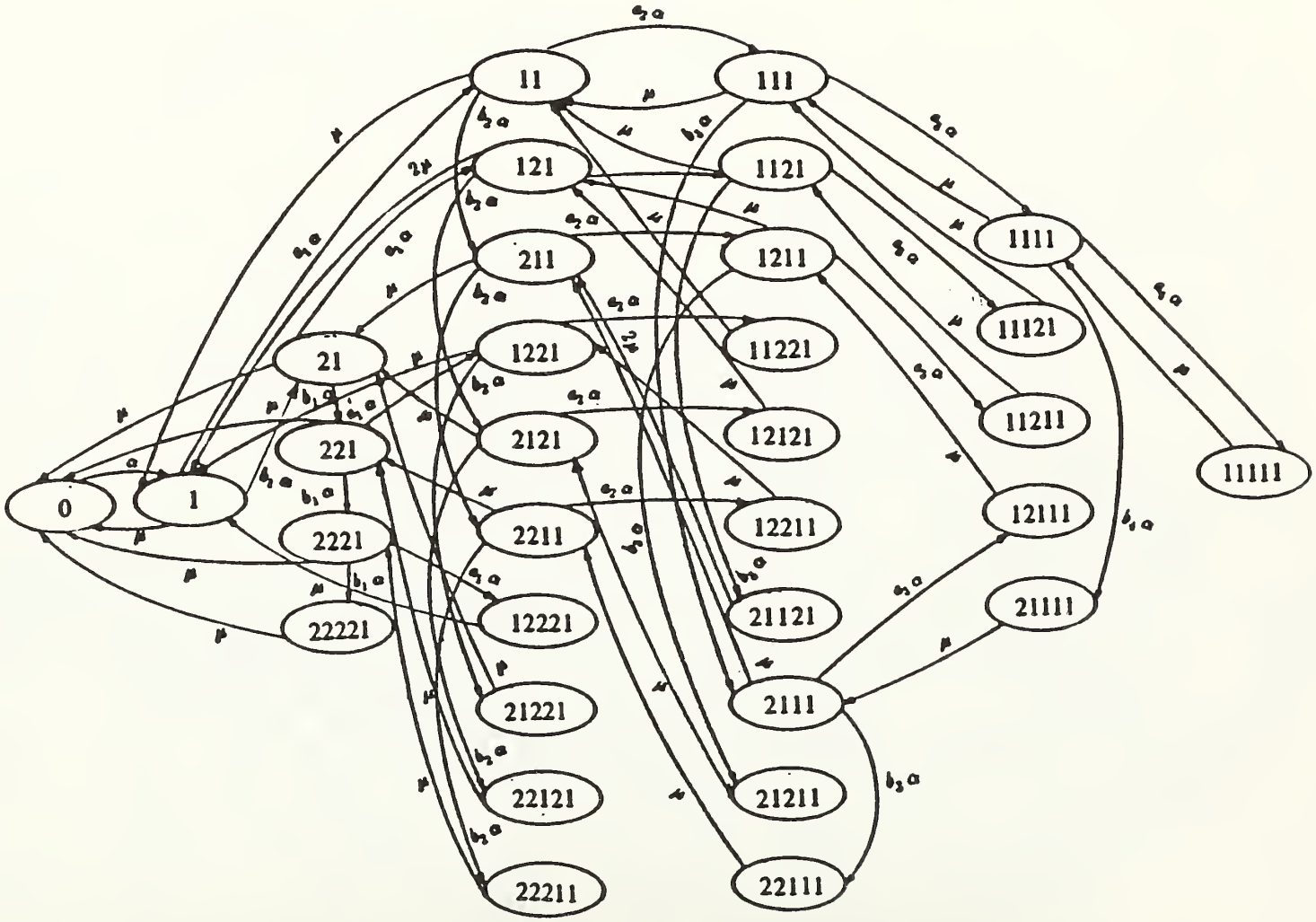


Fig. 4 State-transition-rate diagram of the behavior of S for the case of a single fault-free channel.

Fig. 5 Performance rate of S for the case of two fault-free channels.

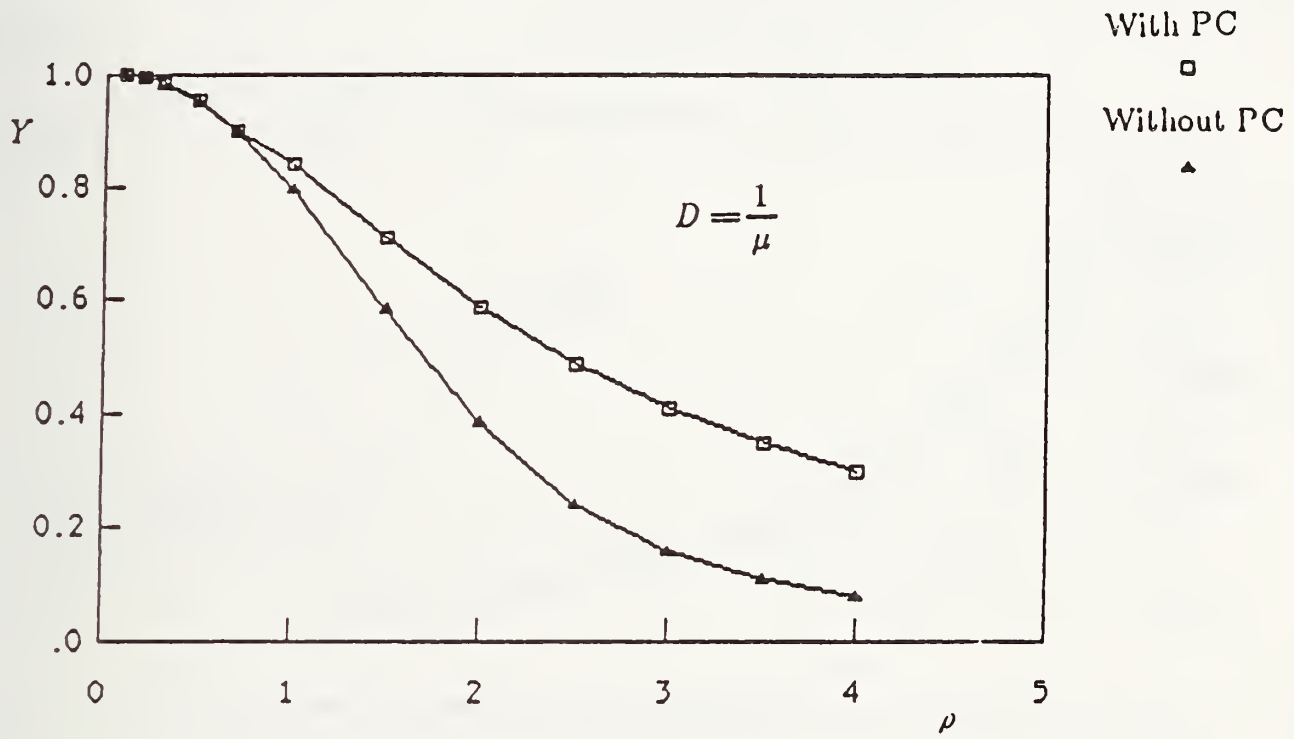


Fig. 6 Performance rate of S for the case of a single fault-free channel.

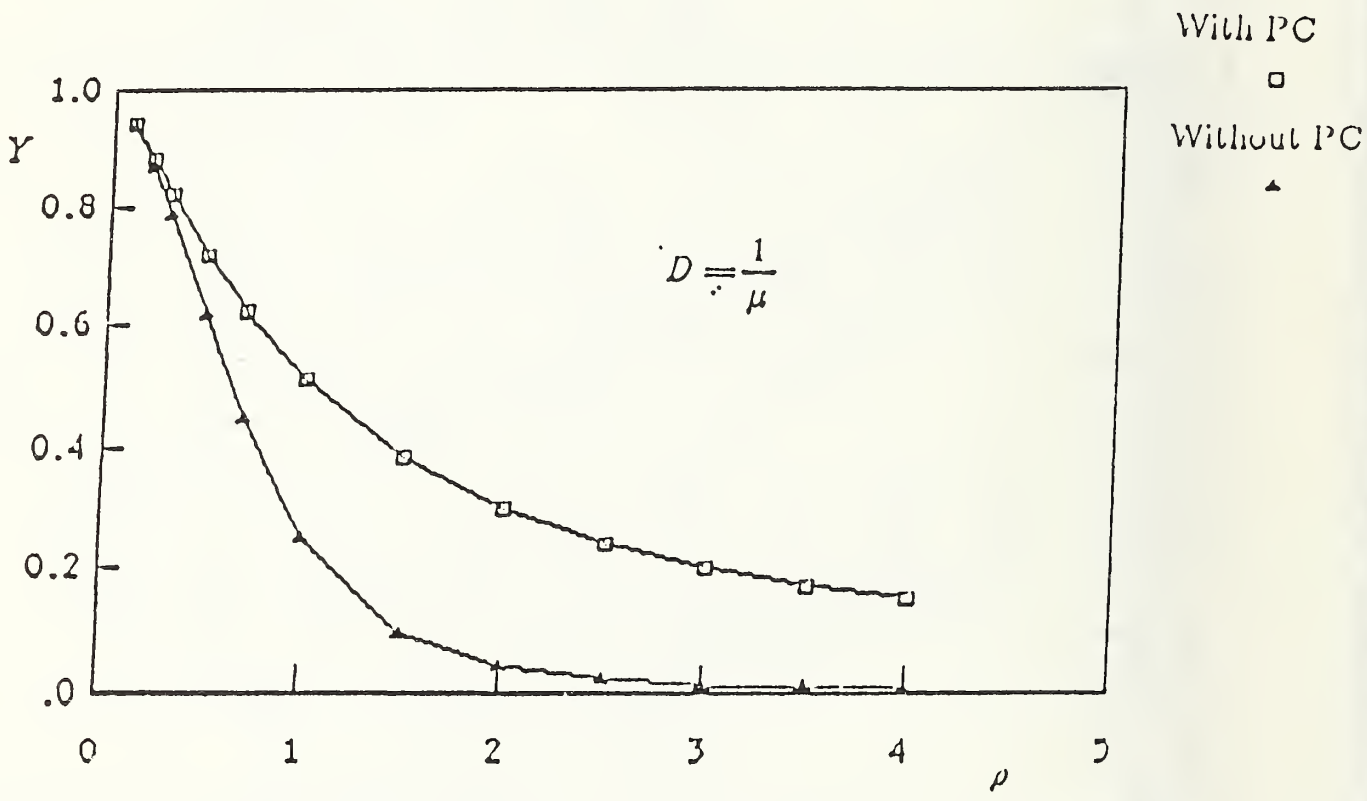
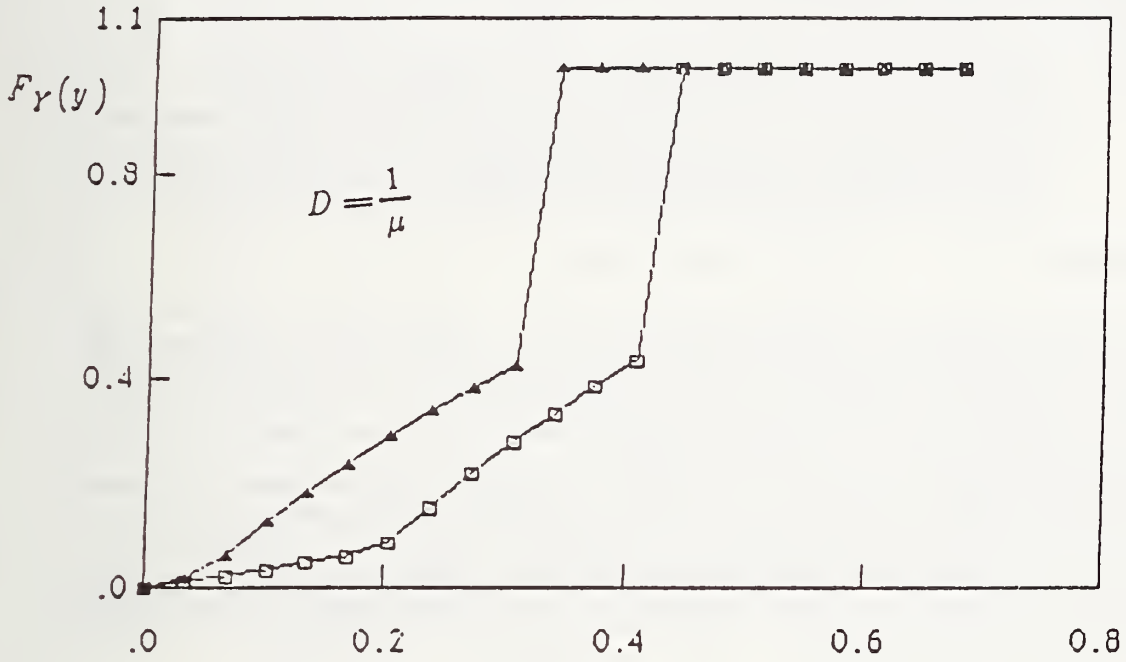


Fig. 7 Plot of $F_Y(y)$ as a function of y for the indicated choices of t and base model parameters.



Token Passing Networks and Starvation Issues.

by Anastase Nakassis
Institute for Computer Science and Technology
National Bureau of Standards, Dept of Commerce.
Building 225, room B221.
Wash. D.C. 20234

Abstract: In what follows we will advance a necessary and sufficient condition for a low priority queue to eventually get and use the token. Then we will use some of the machinery we will develop in the proof of the above mentioned condition in order to explore issues of Target Rotation Time (TRT) allocation and fairness.

INTRODUCTION.

Assume that there are k queues named 1,2,3,...,k. It is assumed that the token is passed from queue 1 to queue 2, to queue 3,...,to queue k, to queue 1, ... and so on. Clearly what matters is the cyclical order, so that one might have named the same queues i,i+1,...,k,1,2,...,i-1 ($1<i<=k$). In particular, the last queue (k) may be any particular queue we wish to denote as "last queue".

By "token passing" we mean something more general than what the term usually denotes (passing the token from one station to the next). In this text, token passing means passing the "right of transmission" and takes place from queue to queue, not from station to station. For reasons that will be explained later, token passing will be thought of as being an instantaneous event. In what follows we assume that we have been given two sequences - h[i] and TRT[i] , i=1,2,3,4, ..., k - such that:

1. If queue i is of high priority, then $h[i]>0$ and $TRT[i]=0$.
2. If queue i is not of high priority, then $h[i]=0$ and $TRT[i]>0$.

Evidently, h stands for token holding time and TRT for Target Rotation Time.

Notice that we do not make any assumption concerning the distribution of high priority queues, although the IEEE 802.4 standard (July 1984, draft F) on which this work is loosely based does imply some distribution.

Let $f(x)=\max\{x,0\}$, that is $f(x)=x$ whenever $x>0$ and $f(x)=0$ otherwise (another way to define f is x plus absolute value of x, divided by two). Then a necessary and sufficient condition for low priority queue i not to starve under any conceivable circumstances is that,

$$TRT[i] > \sum_j h[j] + \sum_j f(TRT[j]-TRT[i]) \qquad (1)$$

In particular, if i is chosen in such a way that TRT[i] is minimal among all positive TRT's, then the above condition becomes a necessary and sufficient condition so that no queue ever starves. Indeed, in this case the left side hits its minimum while the right side hits its maximum as a cursory inspection will readily show.

By starvation, of course, we mean that one can conjure up patterns of token usage which are compatible with the protocol specifications and in which queue i has messages to send but is unable to do so for all times beyond some arbitrary time t.

Finally, let us address the problem of "instantaneous" token transition. If the time it takes to pass the token from one station to the next is constant, then we can reduce each positive TRT entry by D, the total time we spend in token passing during a complete revolution, and act as if the time we spend in token passing is zero. Clearly, if there were low priority queues in the original network for which $0<TRT[i]<=D$, then these queues will starve regardless of the token usage by the other queues. We can therefore assume that $\min \{ TRT[i] \mid i=1,2,3,\ldots,k \}$ exceeds D so that we can reduce each TRT by D and use expression (1).
Alternatively, if the token passing times are constant, then the lemmas we will develop will show that a sufficient and necessary condition is given by (2) below:

$$TRT[i] > \sum_j h[j] + \sum_j f(TRT[j]-TRT[i]) + D \qquad (2)$$

In the more general case, the time needed to pass the token is not constant and one way to use the results obtained under zero token passing time is to insert between any two queues an imaginary high priority queue which holds the token long enough to simulate the token passing time of the original network. Thus, we can always embed the original network into a network with more high priority stations and no token passing time. We will see in an appendix how this construct can be used in order to yield both probabilistic statements and deterministic statements in the general case.
But, for the time being, let us assume that the token passing time is constant and equal to zero.

PROOF OF THE STATEMENT

Definition: A sequence $\{s[i,j]\}$, $i=0,1,2,3,\ldots$ and $j=1,2,3,\ldots,k$, is called admissible iff (if and only if):

1. for all i and j $s[i,j]$ is not negative, and
2. for all j $s[0,j]=0$, and
3. whenever $h[j]>0$, $h[j]>=s[i,j]$, and
4. whenever $h[j]=0$ and $s[i,j]>0$, then
   $TRT[j]>= s[i-1,j]+s[i-1,j+1]+\ldots+s[i-1,k]$
   $+s[i,1]+s[i,2]+\ldots+s[i,j]$.

Admissibility then means that one can create a scenario such that at the i-th revolution the j-th queue has held the token for $s[i,j]$ time units. Evidently, queue j can starve iff there is an admissible sequence such that

$$s[i-1,j]+s[i-1,j+1]+\ldots+s[i-1,k]+s[i,1]+s[i,2]+\ldots s[i,j-1]>=TRT[j] \qquad (3)$$

for all i that exceed some nonnegative i[0].
In this case, even if queue j has messages to send, it is prevented from using the token by the protocol.

In what follows we will assume that j=k. Indeed, from what we remarked in the introduction, there is no loss of generality in assuming that a particular queue is the last queue.

We plan to prove by contradiction that if (1) holds for queue k, then queue k cannot starve. To do so we need some lemmas which, given an admissible sequence that satisfies (3) for all i, i›=i(0), and j=k, will allow us to construct more tractable admissible sequences that also satisfy (3). The following two lemmas show us how to "frontload" an admissible sequence.

Lemma 1. Assume that {s[i,j]} is an admissible sequence. Let {t[i,j]} be another sequence which agrees with s in all terms but two, t[i',j']=s[i',j']+p and t[i',j'+1]=s[i',j'+1]-p (p being a positive constant). The sequence t[i,j] is admissible iff

a. s[i',j'+1]›=p, and either
b. h[j']›=s[i',j']+p›0, or
c. TRT[j']›=s[i'-1,j']+s[i'-1,j'+1]+...
+s[i'-1,k]+s[i',1]+s[i',2]+ ..... +s[i',j']+p.

Proof: Trivial.
Condition a. ensures that all terms of t are nonnegative.
Condition b. ensures that if queue j' is of high priority, then its token usage does not exceed h[j'].
Finally, the forward transfer of credit ensures that from all sums of the type
s[i-1,j]+s[i-1,j+1]+...s[i-1,k]+s[i,1]+s[i,2]+...s[i,j]
the only one to increase is the one obtained when i=i' and j=j'.
Condition c. ensures that even after the increase, this sum does not exceed TRT[j'].

LEMMA 2. Let {s[i,j]} be an admissible sequence such that
s[i',j'+1]=s[i',j'+2]= ... =s[i',m]=0‹p‹s[i',m+1].
If either h[j']›=s[i',j']+p or if
TRT[j']›=s[i'-1,j']+s[i'-1,j'+1]+
...+s[i'-1,k]+s[i',1]+s[i',2]+...+s[i',j']+p,
then we can construct a new admissible sequence by transfering p time units from s[i',m+1]
(s[i',m+1]=s[i',m+1]-p) to s[i',j'] (s[i',j']=s[i',j']+p).

Proof: It suffices to remark that all terms remain nonnegative, that the only entry that increases is s[i',j'] (which remains less than h[j'] if queue j' is of high priority), and that the only (i,j) for which s[i,j]›0 and the sum
s[i-1,j]+s[i-1,j+1]+...+s[i-1,k]+s[i,1]+s[i,2]+...s[i,j]
increases is (i',j'). By hypothesis, if queue j' is of low priority, then TRT[j'] will continue to exceed this sum even after the forward transfer of token usage.

THEOREM 1. There is no admissible sequence {s[i,j]} such that

s[i-1,k]+s[i,1]+s[i,2]+...+s[i,k-1]›=TRT[k] (4)

for m+1 successive values i provided that the number of of low priority queues is m and that condition (1) holds for queue k.

(this means that queue k is of low priority and that
TRT[k]› SUM h[j] + SUM f(TRT[j]-TRT[k])).
j j

Proof: We will argue by contradiction. Assume that m+1 such successive values can be found starting with i=i[0]. Let then i[n] stand for i[0]+n, the n-th successor of i[0]. We observe that (4) implies that s[i,k]=0 for i=i[0],i[1],i[2],...,i[m]. Furthermore, we may assume equality in all of the m+1 occurrences of relation (4). Indeed, whenever the inequality is strict, we can lower some, or all, of the s[i,j] so that the sequence remains admissible and relation (4) becomes an equality for i=i[0],i[1],...,i[m].
Since s[i,k]=0 for i=i[0],i[1],...,i[m], the row sums
s[i,1]+s[i,2]+...+s[i,k-1]
are all equal to TRT[k] for i=i[1],i[2],i[3],...,i[m].
Based on this remark we will prove that if j[1] is the first low priority queue, then the admissible sequence s can be modified in such a way that all of the above properties will hold and in addition:

s[i,j]=h[j] for i=i[1],i[2],...,i[m] and j‹j[1], and
s[i,j[1]]=f(TRT[j[1]]-TRT[k]) for i=i[2],i[3],...,i[m].

Indeed, if there is a high priority queue j, if s[i,j]‹h[j], and if j is followed by queues that use the token, then lemmas 1 and 2 assure us that we can obtain a new admissible sequence by transferring to j token usage from the queues that follow queue j. This transfer will stop when either s[i,j] equals h[j] or all token usage after queue j becomes zero. Thus, if j[1] is the first low priority queue, then one can construct a new sequence s for which all of the above conditions hold and for which s[i,j]=h[j] for i=i[1],...,i[m] and j‹j[1]. Indeed, conditions (1) and (4) ensure that in rows i[1] through i[m] there is enough credit (TRT[k]) to fill queues 1 through j[1]-1.
Since entries 1,2,...,j[1]-1 are identical for rows i[1] through i[m], and since rows i[1] through i[m] sum to TRT[k], then s[i,j[1]]‹=f(TRT[j[1]]-TRT[k]) for i=i[2],...,i[m].
Furthermore, conditions (1) and (4) guarantee that there is enough credit to be transfered in j[1] so that s[i,j[1]]=f(TRT[j[1]]-TRT[k]) for i=i[2],i[3],...,i[m].
Obviously, the transfer of credit to s[i,j[1]] does not destroy the admissibility of s.

By repeating the above arguments, we can prove that if j[2] is the second low priority queue, then we can transfer credit within s in such a way that :

s[i,j]=h[j] for i=i[2],...,i[m] and j[1]‹j‹j[2], and
s[i,j[2]]=f(TRT[j[2]]-TRT[k]) for i=i[3],i[4],...,i[m].

Finally, one can see by induction that if j[m-1] is the [m-1]-th low priority queue, then one can transform s so that:

s[i,j]=h[j] for i=i[m-1],i[m] and j[m-2]‹j‹j[m-1], while
s[i,j[m-1]]=f(TRT[j(m-1)]-TRT[k]) for i=i[m].

But then, all queues between j[m-1] and k are high priority

queues whose token usage time is bounded by the corresponding h value. Therefore, if i'=i[m], then

$$s[i',k]=TRT[k]-s[i',1]-s[i',2]-\ldots-s[i',k-1] \geq$$

$$TRT[k]-\sum_{j} h[j] - \sum_{j} F(TRT[j]-TRT[k]) > 0.$$

Therefore, we have just produced a contradiction. It ensues that when condition (1) holds for queue k and when queue k has messages to send, then it cannot be silenced for more than m turns, m being the number of low priority queues.

Corollary 1. A queue can be forced to remain silent for m turns.

Proof: The proof is implicit in what preceeded. Therefore, we will simply illustrate this corollary by an example. Assume that there are no high priority queues and that there are three low priority queues with TRT values 45, 40, and 30. The following sequence is then admissible:

```
 0  0   0
15 10   5
15 10   0
20 10   0
15 15   0
15 10   5                and so on.
```

Corollary 2. If

$$TRT[k] \leq \sum_{j} h[j] + \sum_{j} f(TRT[j]-TRT[k]) \qquad (5)$$

, then queue k can be made to starve.

Proof. We can always find nonnegative numbers c[j] such that
a. if h[j]>0, then h[j]>=c[j], and
b. if TRT[j], then f(TRT[j]-TRT[k])>=c[j], and
c. c[1]+c[2]+...+c[k]=TRT[k],

provided that (5) holds. Remark that c[k]=0 and that c[j]=0 for all low priority queues j for which TRT[j]<=TRT[k].

Let s[i,j]=c[j] for i=1,2,3,... and j=1,2,3,...,k.
Then we have :

a. All s[i,j] are non negative, and
b. whenever h[j]>0, then h[j]>=s[i,j]=c[j], and
c. whenever TRT[j]>0 and s[i,j]>0, then
   s[i-1,j]+s[i-1,j+1]+...+s[i-1,k]+s[i,1]+s[i,2]+...+s[i,j]<=
   c[j]+c[j+1]+...+c[k]+c[1]+c[2]+...+c[j]=
   TRT[k]+c[j] <=
   TRT[k]+f(TRT[j]-TRT[k]) = TRT[j].
    (notice that s[i,j] (c[j]) cannot be positive unless
      TRT[j]>TRT[k]).

Therefore, there is a sequence, the one we just constructed, which is admissible and which makes queue k starve. Indeed, the protocol will prevent queue k from using the token since:

$$s[i-1,k]+s[i,1]+s[i,2]+\ldots+s[i,k-1]=$$
$$0 \quad +c[1] \quad +c[2] \quad +\ldots+c[k-1] \quad =TRT[k].$$

(END OF PROOF)

It is quite obvious that in the above example not only queue k, but also every low priority queue j for which TRT[j]<=TRT[k] has been made to starve. Thus we have actually proven that there is an admissible sequence for which all starvable queues do starve at the same time. Indeed, we observe that while x is an increasing function of x, $\sum_j h[j] + \sum_j f(TRT[j]-x)$ is a decreasing function of x. Therefore, whenever there are starvable queues, there is among them a starvable queue k' for which TRT[k'] is maximal. But, as stated in the introduction, the names assigned to the queues and the choice of the "last queue" is arbitrary. Therefore, there is no lack of generality in assuming that k'=k (this is tantamount to saying that when there are starvable queues, then we will assign names in such a way that queue k is starvable and TRT[k] exceeds or equals TRT[j] for every other starvable queue j). Under these conditions,

a. Whenever 0<TRT[j]<=TRT[k], queue j fails condition (1) and is starvable, while
b. Whenever TRT[j]> TRT[k], queue j satisfies (1) and is not starvable.

By applying corollary two, we see that there is an admissible sequence that starves queue k and every other starvable queue j.

APPENDIX 1.

On the fairness issue.

One can, conceivably, argue that the starvation issue is not important because in any reasonable network the average input equals the average output. Thus, every queue will sooner or later get the token. If on the other hand the system is unstable, then regardless of starvation the system is not satisfactory.

This argument makes the implicit assumption that the average output is independent -or at least not sensitive to the choice- of the TRT's and of the h's. This assumption is not true for all networks (see ref[2]). But, since this objection does not raise starvation related issues, let us pass onto the next one:

Although a queue may eventually get the token, it is conceivable that it will do so in such an irregular way that the length of the queue's contents will vary greatly with time. Such a feature (feast or famine) is unwelcome in general, it is likely to result in higher standard deviation, and it may be totally unacceptable in situations in which the timely delivery of a message is important (e.g. when very long silences are interpreted as a sign that one's correspondent is no longer " alive and well").

In what follows we will use the insights obtained in the previous section in order to find sequences that will give us an idea of how uneven the token usage can become. Let us assume for instance that there are k low and k high priority queues in the following order:

TRT[1]h[1]TRT[2]h[2].....TRT[k]h[k]

(notice that we are using the same symbol to denote a queue and its token rotation time/token holding time). Let us also assume that :

a. TRT[k]=TRT=min{TRT[1],TRT[2],TRT[3],...,TRT[k]}, that
b. TRT>S= SUM h[j] +SUM f(TRT[j]-TRT), that
j j
c. F=TRT-S, and that,
d. U[i]=TRT[i]-TRT, i=1,2,...,k.

We can contruct then the following admissible sequence :

L0:	0	0	0	0	.....	0	0	0	0
L1:	U[1]	h[1]	U[2]	h[2]	.....	U[k-1]	h[k-1]	0	h[k]
L2:	U[1]+F	h[1]	U[2]	h[2]	.....	U[k-1]	h[k-1]	0	h[k]
L3:	U[1]	h[1]	U[2]+F	h[2]	.....	U[k-1]	h[k-1]	0	h[k]
Lx:	.....								
Lk:	U[1]	h[1]	U[2]	h[2]	.....	U[k-1]+F	h[k-1]	0	h[k]
Lk+1:	U[1]	h[1]	U[2]	h[2]	.....	U[k-1]	h[k-1]	F	h[k]
Lk+2:	U[1]	h[1]	U[2]	h[2]	.....	U[k-1]	h[k-1]	0	h[k]

In the above sequence row L1 contains admissible but arbitrary values. All other values have been found under the assumption that all queues are full and can use as much time as they are allowed by the protocol. Since rows L2 and Lk+2 are identical, the finite sequence given above will reproduce lines L1 through Lk+2 until such time that a queue will not be able to use all the time that the protocol allows it to. We see then that the token usage for the j-th low priority queue is proportional to (k+1)*(TRT[j]-TRT)+F. This sequence may or may not be a worst case sequence. It implies though that the token usage does not depend as much on the relative values of the TRT's as on the relative values of their differences and the quantity we called F. For instance, assume a network having no high priority queues but three low priority queues with T's equal to 70, 45, and 40 time units, respectively. Then F=40-30-5=5 and a possible sequence is :

L0:	0	0	0
L1:	30	5	0
L2:	35	5	0
L3:	30	10	0
L4:	30	5	5
L5:	30	5	0

and so on.

As usual, it is assumed that the second row values are arbitrary, but the rest of the values are obtained on the assumption that each queue can use the token for as much time as it is allowed to hold it by the protocol. In the example above, the relative token usage is 125/25/5 and the (token usage)/TRT ratios are 1.79/0.56/0.125

A "cure" for the above situation could be to force all the TRT[i]'s to be identical. This can also create problems. On the one hand it denies us the freedom to discriminate between low priority queues. It is also not much of a cure as the following example will show:

Assume one high priority queue with holding time h[1]=1 followed by k (k=2m-1) low priority queues with all TRT values TRT[1],TRT[2],...,TRT[k] equal to t+1. Then, if all low priority

queues are full, one admissible sequence is the following:

```
   L0:  0  0   0   0   0   0   ...   0   0   0   0
   L1:  0  t+1 0   0   0   0   ...   0   0   0   0
   L2:  1  0   t   0   0   0   ...   0   0   0   0
   L3:  0  1   0   t   0   0   ...   0   0   0   0
   L4:  1  0   0   0   t   0   ...   0   0   0   0
   L5:  0  1   0   0   0   t   ...   0   0   0   0

   Lx:  ........................................
 Lx+1:  ........................................
 Lx+2:  ........................................

 Lk-3:  1  0   0   0   0   0   ...   t   0   0   0
 Lk-2:  0  1   0   0   0   0   ...   0   t   0   0
 Lk-1:  1  0   0   0   0   0   ...   0   0   t   0
   Lk:  0  1   0   0   0   0   ...   0   0   0   t
 Lk+1:  1  0   0   0   0   0   ...   0   0   0   0    **
 Lk+2:  0  t+1 0   0   0   0   ...   0   0   0   0          and so on.
```

In this instance the token usage times are proportional to
t+m/t/t/.../t/t .
If t is small compared to m, then the first low priority queue enjoys a pronouncedly unfair advantage. If on the other hand t is big, then each queue will have to wait for a long period of time before getting the token. Finally, it must be noticed that every now and then we manage to shut off all queues but one (see the ** row). If the token passing time is constant and relatively big, then we have wasted a considerable amount of time doing little but token shuffling.

Appendix 2.

The case when the token passing time cannot be treated as a constant.

As we remarked earlier, one way to extend our results to the case in which the token passing times are not constant is to consider that between any two queues of the original network there is a phantom high priority queue that holds the token during the token passing phase between two adjoining real queues. Evidently, these phantom queues do not emit any "real messages" but they otherwise behave as high priority queues. If upper bounds can be found for their h values, then we could set their h values as equal to these upper bounds. If not, we can set them equal to plus infinity. A quite natural assumption is that the time needed for token passing between two adjoining stations i and j (j is i+1 except when i=k; in the latter case j=1) during the n-th rotation, X[j,n], is a random variable whose values are independent of the network's history and have a distribution function H(.;j). Assume then that there are k phantom queues and a sequence d[i], i=1,2,3,...k, such that

$$TRT[k] \geq \sum_j h[j] + \sum_j f(TRT[j]-TRT[k]) + \sum_j d[j] \qquad (6)$$

(Clearly d is zero for the real queues while h and TRT are zero for the phantom queues.
)

If (6) holds, and if q is defined by

$$q=\text{Prob}\{ X[j,n] < d[j] \text{ for } 1 \leq j \leq k \text{ and } m+1 \text{ consecutive values of } n\}, \qquad (7)$$

then the probability that queue k will be silent for m+1 consecutive revolutions is at most 1-q.
(Remark: We still assume that the last queue, k, is a real low priority queue. Thus the time it takes to go from queue k to the next real queue is thought of as being part of the next rotation.
)

The proof is straightforward: Given a sequence of observed values, the probability is q that (7) holds. For these sequences, queue k cannot have been silent for the m+1 token rotations that (7) covers. Indeed, if it were, then there would exist an admissible sequence for which phantom queue j holds the token d[j] time units or less for m+1 consecutive rotations, m being the number of the low priority queues; at the same time, the protocol is preventing queue k from using the token. If this were possible, then we could construct a new network in which the present network's "phantom" queues have been dubbed "high priority" queues with holding times equal to the corresponding d values. The sequence that makes queue k silent for m+1 turns in the present network would be admissible in the new network. Since the old and the new network have the same number of low priority queues with the same TRT's, and since relation (7) assures us that (1) will hold for the new network, we know that there can be no admissible sequences that can deny queue k the token for m+1 consecutive rotations.

It is quite obvious that the above result can be strengthened in several ways. For instance, we could have started as follows:

Let d[j]=max{ X[j,n] | n=n[0],n[0]+1,...n[0]+m} for every phantom queue j, and let q be the probability that (6) holds. Then, given any sequence of m+1 successive rotations, the probability that queue k will be denied the token in each and every rotation in this sequence is at most 1-q.
It is also quite obvious that we did not need to assume independence between the network's history and the times needed to pass the token, but in this case it would have been difficult to arrive at reasonable estimates of q. Since it is also clear that all of the above statements are vacuous when q is zero, from a practical point of view we need models which are both reasonable and tractable.
Clearly, if our network starts exhibiting pathological behavior, a more appropriate assumption is that the X values are not independent of each other and of the network's history. But, it would hardly be appropriate to focus on the problems of pathological behavior (in this instance the network's behavior when several nodes and/or the medium malfunction) before we more or less understand the normal, usual, and ordinary network functioning. From this point of view, it would appear that the independence assumption for the X values is appropriate.

Appendix 3.

The effects of exceeding the protocol's time allocation.

In what preceeds we assumed that whenever the protocol allows a queue to transmit for t time units, then the queue will hold the token for t time units or less. But, whenever a queue has not exhausted either its messages or the allowed time t, it will attempt to transmit one more message. Therefore, if the queue has sufficiently many messages, the last transmission may start at time t-d and end at time t+e so that the actual token usage (t+e) exceeds the allowed

token usage (t). When this effect is considered, it is clear that our previous results need be adjusted. Fortunately, it is easy to reformulate what preceeds in such a way that only minor changes will be needed.

While the protocol allocates token usage on the basis of the h and TRT sequences, the observed usage must reflect the time extensions due to the transmissions that extend the allocated time. Assume therefore that for each high priority queue i, i=1,2,3,...,k, he[i] exceeds h[i] by the length of the longest possible extension for queue i, and that TRTe[i]=0. Similarly, if j, j=1,2,3,...,k, is a low priority queue, TRTe[j] exceeds TRT[j] by the length of the longest possible extension for queue j and he[j]=0. Thus, a high priority queue i, i=1,2,...,k, can keep the token for up to he[i] time units while a low priority queue j, j=1,2,3,...,k, can keep the token for up to TRTe[j]-R units provided that the observed previous rotation time, R, is less than TRT[j]. Assume also that F is a real function defined as follows:

F(j,i) equals 0 unless TRT[j]›TRT[i]›0 in which case F(j,i) equals TRTe[j]-TRT[i].

Given these definitions, we find that all precceding results hold, provided that we substitute he[j] for h[j] and TRTe[j] for TRT[j] whenever TRT[j] exceeds TRT[i]. As an example, Theorem 1 holds if condition (1) is modified as follows:

$$TRT[i] \rightarrow \sum_{j} he[j] + \sum_{j} F(j,i) \qquad (8)$$

A cursory examination of our statements and conditions reveals that we can systematically modify and reformulate them in such a way that these statements will remain true while all possible token usage extensions will be correctly treated.

REFERENCES

[1] "An introduction to Probability Theory and Its Applications", volumes I and II, by William Feller. (John Wiley and sons).

[2] "Stability issues in token passing networks", by A.Nakassis. Internal ICST (NBS) memo.

Session III:

Performance Issues

Chairperson: Karen Hsing
National Bureau of Standards

Performance Analysis of the 802.4 Token Bus Media Access Control Protocol

Jade Y. Chien
Network Architect
Ungermann-Bass, Inc.
Santa Clara, California

Abstract

IEEE Standard 802.4-1984 defines a local area network protocol based on the concept of token-passing for controlling access to a broadcast medium. A performance analysis of such a network using simulation techniques has been conducted. Performance is characterized in terms of stability, fairness, throughput, and acquisition delay.

This paper is a report on some of those efforts. Our analysis shows that the network remains stable as the load increases. Fairness can be attained if enough time is allowed for the system to become saturated. The acquisition delay is sensitive and degrades greatly as load increases.

A comprehensive discussion of how the performance of the network is affected by system parameters like data length, network sizes, token hold time, and station delay is also included.

Keywords: local area network; 802.4; Media Access Control; token bus; simulation; stability; fairness; throughput; acquisition delay; performance characteristic.

Table of Contents

1. Introduction

Due to the recent interest in industrial automation, the IEEE 802.4 Token Bus Media Access Control (MAC) for Local Area Networks has received a great deal of attention. One of the most attractive architectures for such a network is a shared, ordered access broadband system with distributed control.

Nevertheless, the performance characteristics of the token bus protocol do not seem to be well understood. At Ungermann-Bass, Inc. we have recently conducted a performance analysis of this media access method based on a simulation model. The intent of the present study was to further our understanding of the system by characterizing the behavior. In addition, an attempt is made to assess performance under different configurations and investigate various important design parameters.

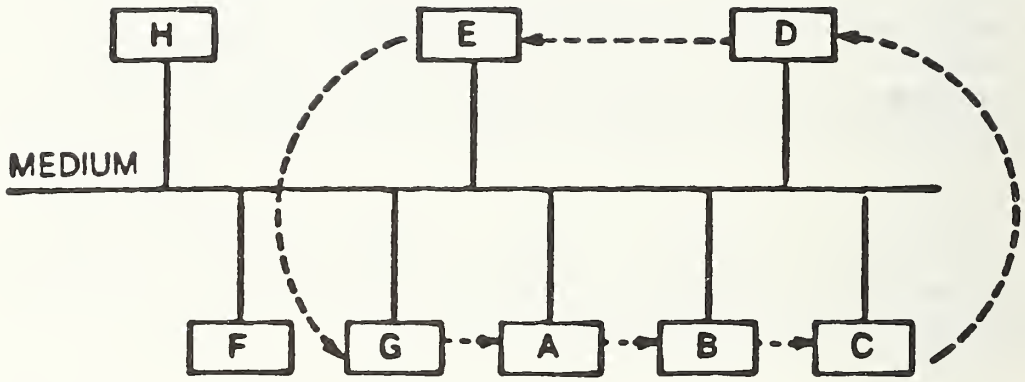
It is worth mentioning that successful use of a local computer network depends on much more than the specific communication medium. Many applications use higher level protocols, including a complete architecture for internetwork communication among numerous different systems. Those topics, however, are beyond the scope of the current study, which is directed primarily at the behavior of the token bus protocol itself.

In the sections which follow, we review the token bus protocol, describe the methodology and the test environment, characterize the behavior of the system, and finally highlight some of the important parameters.

2. Review of the Token Bus Protocol

The Media Access Control Sublayer controls when a station may transmit. Token passing on a bus network is a method by which multiple stations share a common bus without conflict. A control packet known as the token regulates the right to access. The token frame contains a destination address. The station receiving the token acts as the temporary master of the network, and may transmit one or more frames for up to a pre-set time limit.

The stations are assigned logical positions in an ordered sequence. This sequence by which the token is passed among the stations is commonly referred to as a "logical ring". Note that not all active stations need to participate in the logical ring. Non-token-using stations are allowed on the bus. Figure 1 illustrates this concept.



Note:
Participating Stations:
A,B,C,D,E, and G
Non-Token Using Stations:
F and H

Figure 1 — Logical Ring on Physical Bus

This scheme requires considerable maintenance. The maintenance functions include ring initialization, insertion into the logical ring, deletion from the ring, token passing obligation, and fault management. These functions are performed by one or more stations on the bus. However, the token holding station has a bigger share. The token holder may invite new stations to enter the ring periodically, and may remove itself from the ring. The token holding station monitors its own token passing process and detects multiple-token situation.

Lastly, as an option, a token bus system can include classes of service that provide a mechanism of prioritizing access to the bus.

3. The Methodology

The most popular approach to the solution of a complex system model is to simulate it, i.e. to use a program which behaves like the model and observe the behavior of the program. The principal advantage of simulation is its great generality.

The simulation model describes the operation of the system in terms of individual events of the individual elements in the system. The interrelationships among the elements are also built into the model. Then the model allows the computing device to capture the effect of the elements' actions on each other as a dynamic process. Another important characteristic of the simulation model is that simultaneous resource processing can easily be taken into consideration.

Our approach to simulating the token bus protocol is a self-driven, event-advance simulation. The simulation model is driven by generating statistical input data. The simulation program provides the timing mechanism for the simulated system and takes simulated time into consideration in its actions. The approach is to identify significant events in the simulated system, i.e., times when noticeable changes occur. It is at these points in simulated time that the simulation program must take action.

Each event is described by the time at which that event is to occur and by the action that takes place. For a queuing network model, a typical event is the completion of a job's service time.

The simulation program maintains a list of events ordered by time of occurrence. The program cycles through the following steps:

1) Select the event with the earliest time.

2) Set the simulated clock to this time.

3) Perform the action.

At Ungermann-Bass, Inc. we have employed PASCAL to implement our simulation model. This tailored simulation describes the topology and access mechanism to a sufficient level of detail. The list of events is represented by a linked list. Each list element includes the time and the attributes of the event. We have considered the following types of events:

Frame arrival - simulates frame arrival at the station

Station arrival - simulates non-token-using station desire to enter ring

Token holding - simulates when a station hears a token addressed to itself and has the right to transmit

Sending frame - simulates the process of frame transmission

Frame reception - simulates frame received at the destination

Token passing - simulates the token passing obligation

Solicit successor - simulates the sending of the solicit_successor frame, opening response window, resolving contention, and allowing new station to join.

The simulator maintains two waiting queues at each station, the transmit queue and the receive queue. Care is taken not to allocate a station for more than one frame concurrently by queuing up requests for the station in a FIFO manner. However, concurrent services of different frames at different stations are allowed. In fact, the model takes into consideration the various pipelining processes.

3.1 Model Assumptions

The following model assumptions were used in our simulation:

- The network is in a steady state condition where a logical ring has been established and no error conditions are present.

- The Head End Remodulator is in the center of the network configuration. The radius of the network (the distance of the farthest unit to the Head End Remodulator) is less than one mile. The cable propagation delay is 1 microsecond for every 1000 feet. The delay at the Head End Remodulator is 7 microseconds.

- The channel bandwidth is 10 Mbps (Mega-bits per second).

- The six byte addressing structure has been adopted.

- A token frame is recognized by the hardware as soon as the Frame Control (FC) field is read and the Destination Address equals This Station Address. However, the Frame Check Sequence (FCS) field has to be read and checked to make sure that the token frame is good and is not garbled. In other words, the transmitted message must be received to recognize a good token.

- The length of the preamble is 4 bytes.

- The modem delays on the transmit and the receive sides are 0.5 microseconds and 3 microseconds respectively.

- After a station has received a token, the delay to release a new token is one microsecond. The delay to transmit the first pending frame at the transmit queue is one station delay. Station delay is the longest time a station waits to begin transmitting after having heard a frame to which it should respond. The Transmit State Machine will be signaled in this situation.

- Stations not using the optional priority feature transmit every data frame with an access class of 6 (the highest priority).

- The workload model is described as follows: Frames arrive at each station according to Constant or Poisson distribution with mean interarrival time, the length (in bytes) being constant. The destination for the frame is chosen at random from the other station.

- The simulated error rate is 1 bit in 10^9. Since this error rate is so low, frames received in error have been excluded from all of the results reported.

- Non-token-using stations require ring insertion according to a Constant distribution with mean interarrival time.

3.2 Parameters

The simulated configurations are fully parameterized. For the simulator, the following model parameters can be set:

1) Bandwidth of the bus

2) Maximum number of stations that can be connected on a single physical channel

3) Initial number of participating stations on the bus

4) Propagation delay between stations (includes delay in station, cable and Head End Remodulator)

5) The interarrival time of frames at each station

6) The type of arrivals (Constant or Poisson)

7) The length of the frame

8) The interarrival time of non-token-using stations requiring entrance

9) The number of token possessions that determine how often a station opens response window (max_inter_solicit_count)

10) The token hold timer that determines how long a station can transmit frames

11) The station delay

12) The slot time

13) The length of the simulation run

3.3 Statistics Collection

As frames are processed by the model, the following statistics are collected:

1) Simulated time

2) The number of stations that have been granted ring entrance after the logical ring was initialized

3) The acquisition delay -- measured from the time when a frame arrives at the transmit queue until the first bit is transmitted onto the wire.

4) The latency -- the average delay in sending a data frame, which includes the queuing delay, the token delay, the station delay, and the transmit delay

5) The total offered load -- the total number of data bits generated by all the active stations per second (expressed in terms of percentage of channel bandwidth)

6) The network throughput -- the total number of data bits received at the destinations per second (expressed in terms of percentage of channel bandwidth)

7) For each station:

   - the length of time the station has been in the ring
   - the offered load of the station
   - the throughput achieved by that station

Note that the performance measures in numbers 5, 6, and 7 only take the data bits into consideration. The protocol overhead is not included.

4. Performance Characteristics

In this section, the simulator model is used to investigate the performance of the token bus protocol. We have researched the performance of the network under various configurations and loading situations.

A detailed study of the effect of different factors on the system is made to allow general conclusions to be drawn. The critical performance measures are the acquisition delay, the offered load, and the network throughput. Demonstrative examples have been chosen to facilitate the illustration of the following results.

4.1 Stability

The frame interarrival rate, the data length, and the number of active stations are three important parameters of the total offered load. An increase in any of these components will produce an increase in the offered load. The topic of interest is the stability of the protocol as the load increases. We will investigate the stability of the system when these parameters are varied.

For simulations in this section, we assume all the stations on the bus are active. Since every station generates traffic with the same rate, the load offered by each one should be statistically identical. Additionally, the following parameters are constant.

frame arrival type = POISSON
simulation time = 10 seconds
station delay = 100 microseconds
slot time = 250 microseconds
token hold timer = 4000 microseconds
max_inter_solicit_count = 100

4.1.1 Varying Frame Arrival Rate

The offered load was varied by changing the arrival rate. Note that the arrival rate is reciprocal to the interarrival time.

We start with a modest offered load, and then increase that level by increasing the load being offered by each station.

The data in Figure 2 shows the effect of changing arrival rates and represents the offered load and the network throughput measured with 100 active stations transmitting constant 500 byte frames. Each station generates this data length according to Poisson distribution with mean rates of 5, 10, 15, 20, 25, 30, 35, and 40 frames per second. These conditions permitted measuring the performance of the system with total offered loads of 20% to 160% of the channel capacity. We have observed that the network throughput is an increasing function of the arrival rate at each station.

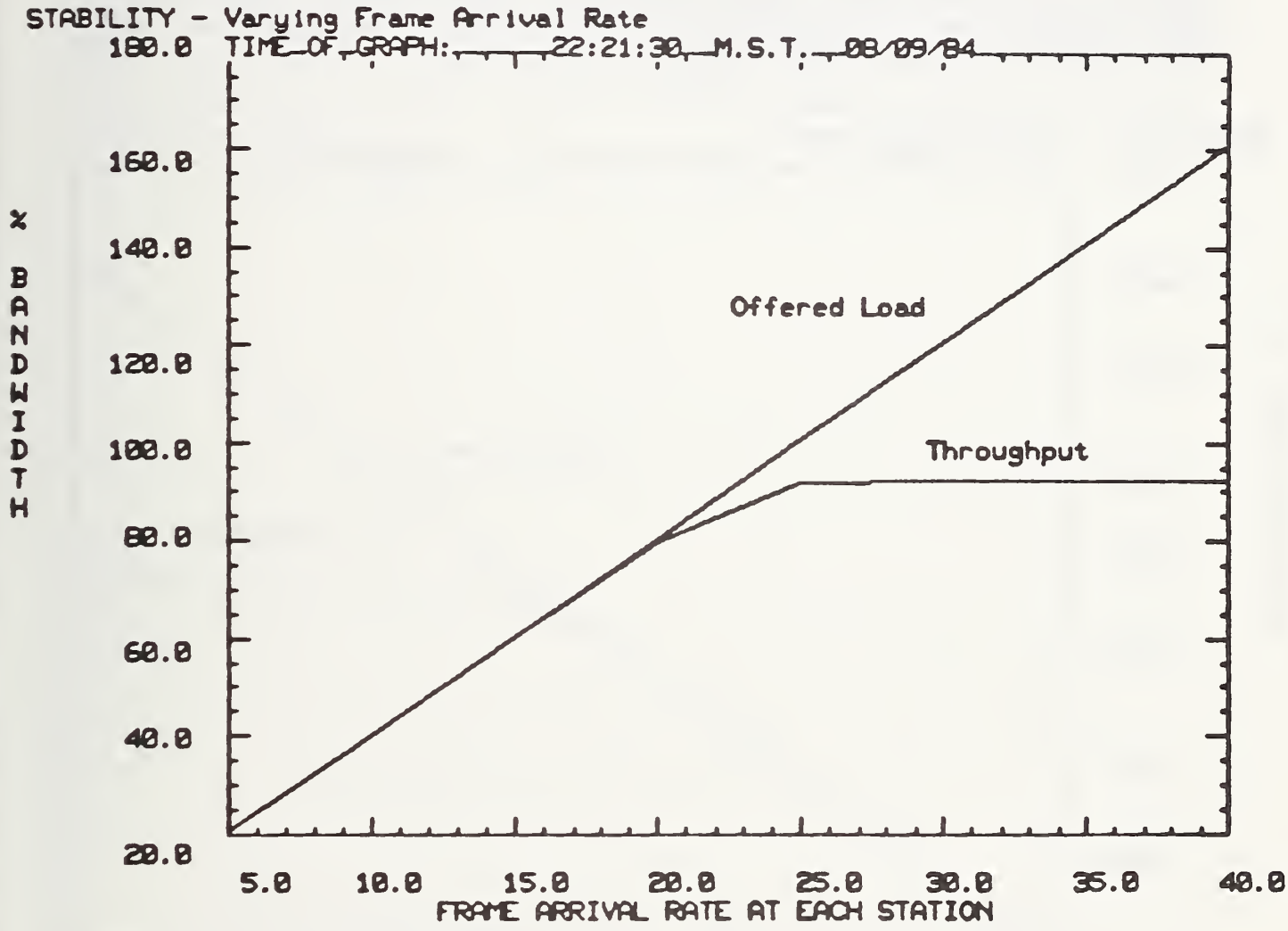


Figure 2 — Stability — Varying Frame Arrival Rate

4.1.2 Varying Data Length

The effect on stability of different data length is now investigated. The simulated configuration is 100 stations with frame arrival rate of 10 frames per second. The constant data lengths are 250, 500, 750, 1000, 1250, 1500, 1750, and 2000 bytes. The total offered load will vary from 20% to 160% of the channel capacity. Figure 3 shows that an increase in the data length produces an increase in the total offered load, and thus an increase in the network throughput.

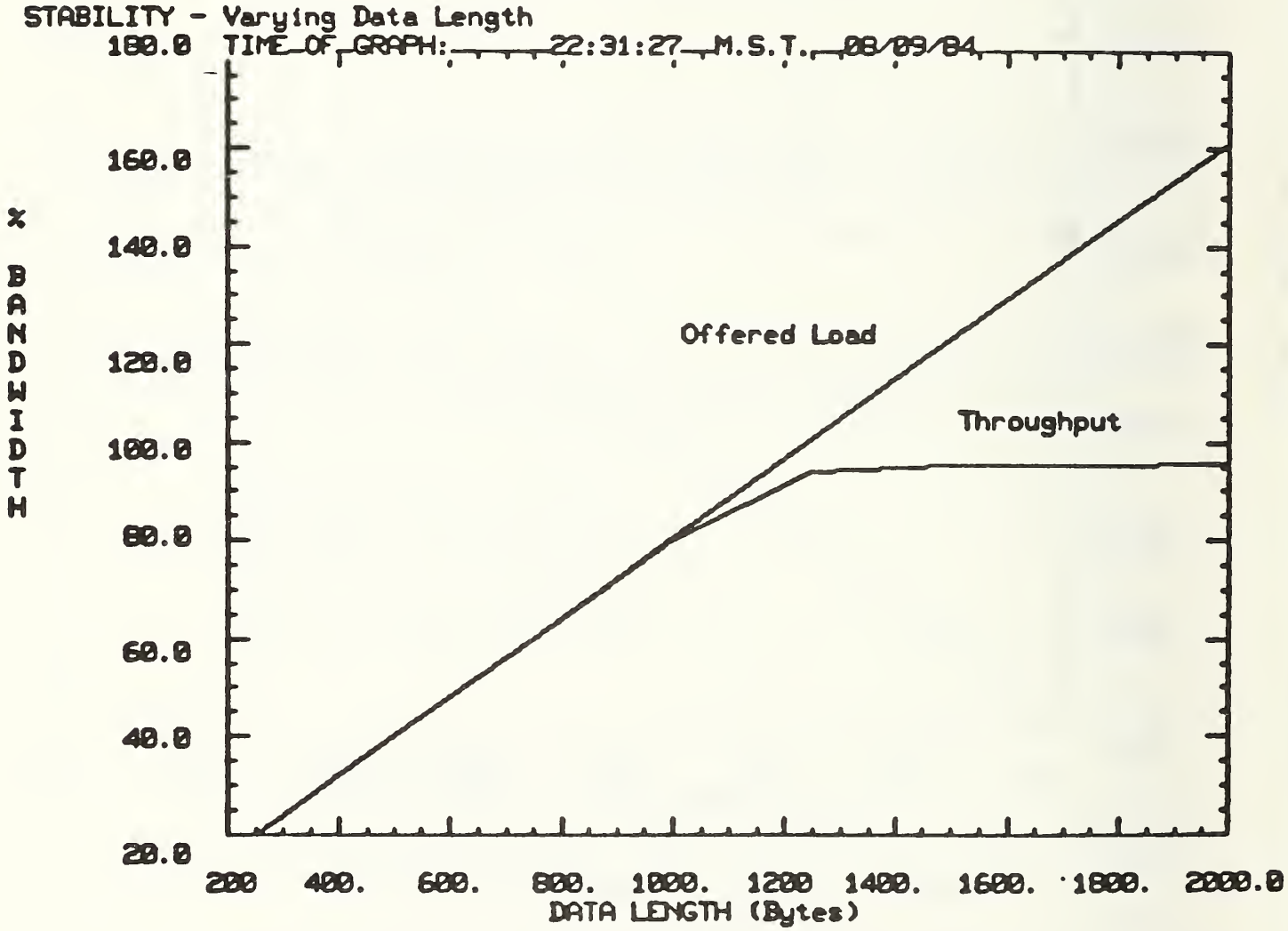


Figure 3 — Stability — Varying Data Length

4.1.3 Varying Number of Active Stations

We start with a light offered load, and then increase that level by adding more stations. The frame arrival rate of each station is 10 frames per second. The data length is 500 bytes. We have cases including 50 stations, 100 stations, 150 stations, 200 stations, 250 stations, 300 stations, 350 stations, and 400 stations. This results in total offered loads of 20% to 160% of channel capacity. Again, we observe an increase in the number of active stations causes an increase in network throughput. The results are plotted in Figure 4.

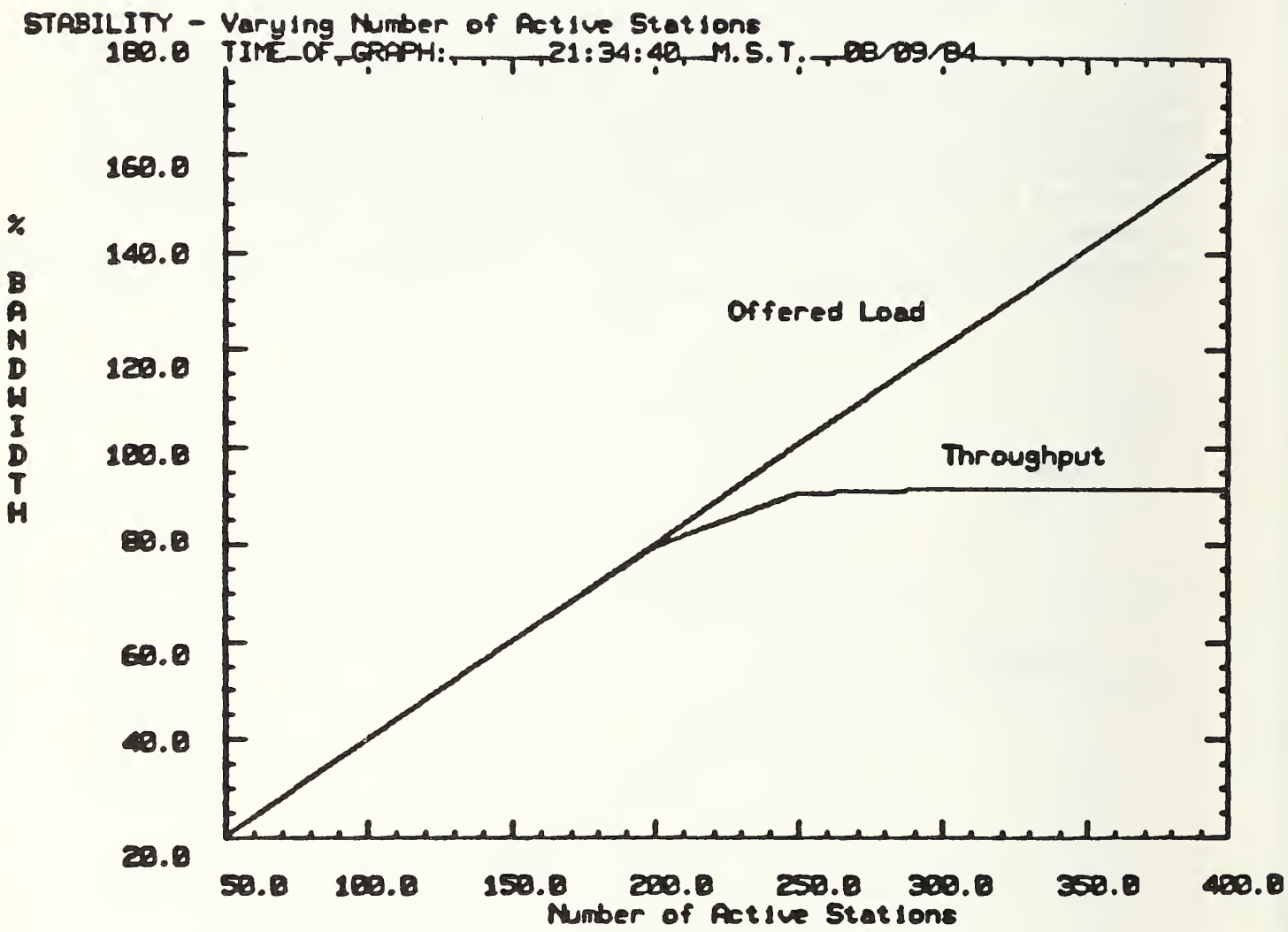


Figure 4 — Stability — Varying Number of Active Stations

The stability of the system for varying the above three components of the offered load has been examined. Due to the deterministic nature of the protocol, it is not surprising that the network throughput is a function of the offered load and has a positive slope. As the above result shows, the system remains stable under overload situations. The throughput remains high and shows no signs of suddenly decreasing or becoming unstable.

4.2 Fairness

The token passing access mechanism is deterministic. However, the implementation of the method on a local area network may introduce nondeterministic factors such as the dynamics of the insertion and deletion of a station in the logical ring. The issue of fairness is addressed in this section.

As a first attempt, let us assume that all the active stations transmit every data frame with an access class of 6. If the network is fair, each station should have an equal access to the channel. Thus, the throughput achieved by each station should be proportional to the load offered by that station. This ratio is defined to be R. The protocol is very fair in its allocation of channel capacity if all the active stations have the same R value. More mathematically, the standard deviation of the R values for all active stations at any given load is small.

Under normal load, the system should be able to accommodate most of the traffic. Thus, all the frames sent should be able to get through and each station does not need to fight for its desired bandwidth. Fairness is in general not an issue. However, for a heavy load system, this is not necessarily true.

The following parameters are constant in all simulations in this section.

frame arrival type = CONSTANT
frame interarrival time = 0.0025 second
(frame interarrival rate = 400 frames/sec)
data length = 500 bytes
maximum no. of stations = 10
station delay = 100 microseconds
slot time = 250 microseconds
token hold timer = 4000 microseconds
max_inter_solicit_count = 100

4.2.1 No Insertion After Initialization

Suppose all 10 stations were granted entrance at ring initialization, and they have been in the logical ring for about the same amount of time. The following simulation captures the behavior of the system after 10 seconds simulated time. The total offered load is 160% of the channel bandwidth. The system is very congested. The network throughput is 92.41%. The average acquisition delay is 2112.117 msec. The offered load, the achieved throughput, and the R ratio of each station is tabulated as follows.

Station No.	Time in ring (sec)	Station offered load	Station throughput	R ratio
10	10	16.00	9.25	0.57825
9	10	16.00	9.26	0.57850
8	10	16.00	9.24	0.57725
7	10	16.00	9.22	0.57650
6	10	16.00	9.23	0.57675
5	10	16.00	9.23	0.57700
4	10	16.00	9.24	0.57725
3	10	16.00	9.24	0.57775
2	10	16.00	9.25	0.57800
1	10	16.00	9.25	0.57825

Table 1: No Ring Insertion (10 sec)

Figure 5 depicts the above result.

Statistics for the R ratio:

mean = 0.57755
standard deviation = 0.0007
coefficient of variation = 0.0012

The coefficient of variation is defined to be the ratio of standard deviation to mean. The low variance implies that the protocol is very fair to all stations that have been in ring for the same amount of time.

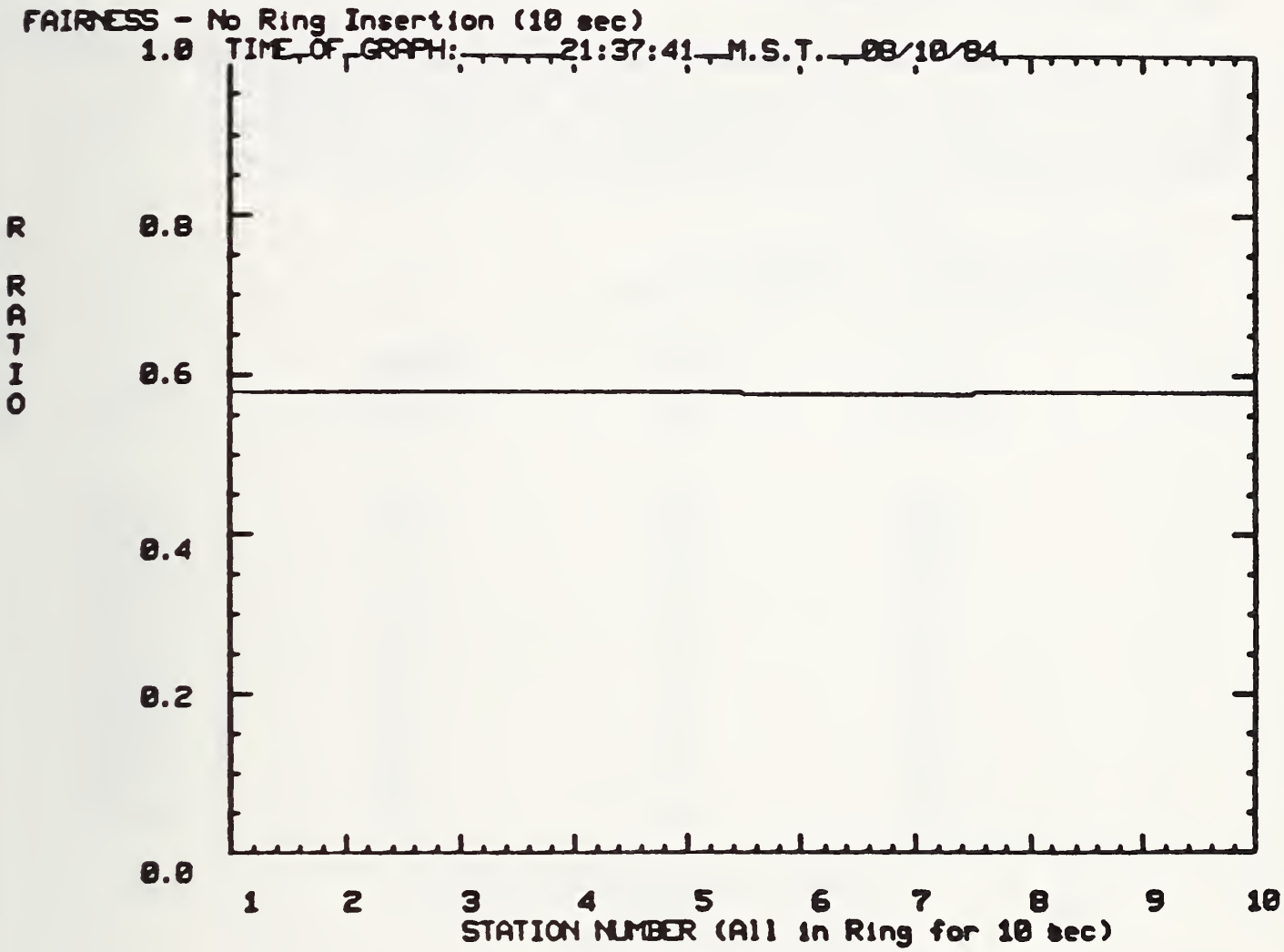


Figure 5 — Fairness — No Ring Insertion (10 sec)

4.2.2 Insertion After Initialization

Next, we introduce the dynamics of station insertion after ring initialization. We have assumed that there are two stations that have joined at initialization. Every 0.9 second, a non-token-using station will desire ring entrance. A snapshot of the performance of the system after 10 seconds simulated time is depicted as follows.

offered load = 94.60%
network throughput = 79.07%
average acquisition delay = 275.491 msec

Station No.	Time in ring (sec)	Offered load	Station throughput	R ratio
8	10.000	16.00	14.09	0.880750
4	10.000	16.00	14.07	0.879500
9	9.098	14.56	12.64	0.868370
2	8.181	13.09	11.17	0.853301
6	7.253	11.60	9.69	0.835229
5	6.362	10.18	8.24	0.810142
1	3.784	6.05	4.28	0.707204
10	3.784	6.05	4.28	0.707204
3	0.370	0.59	0.32	0.544218
7	0.308	0.49	0.28	0.569106

Table 2: Ring Insertion (10 sec)

Statistics for the R ratio:

mean = 0.7655
standard deviation = 0.1273
coefficient of variation = 0.1663

The variance is high, suggesting that the protocol is not allocating channel bandwidth fairly to the participating stations. The longer the station has been in the ring, the more bandwidth it gets. The results are plotted in Figure 6.

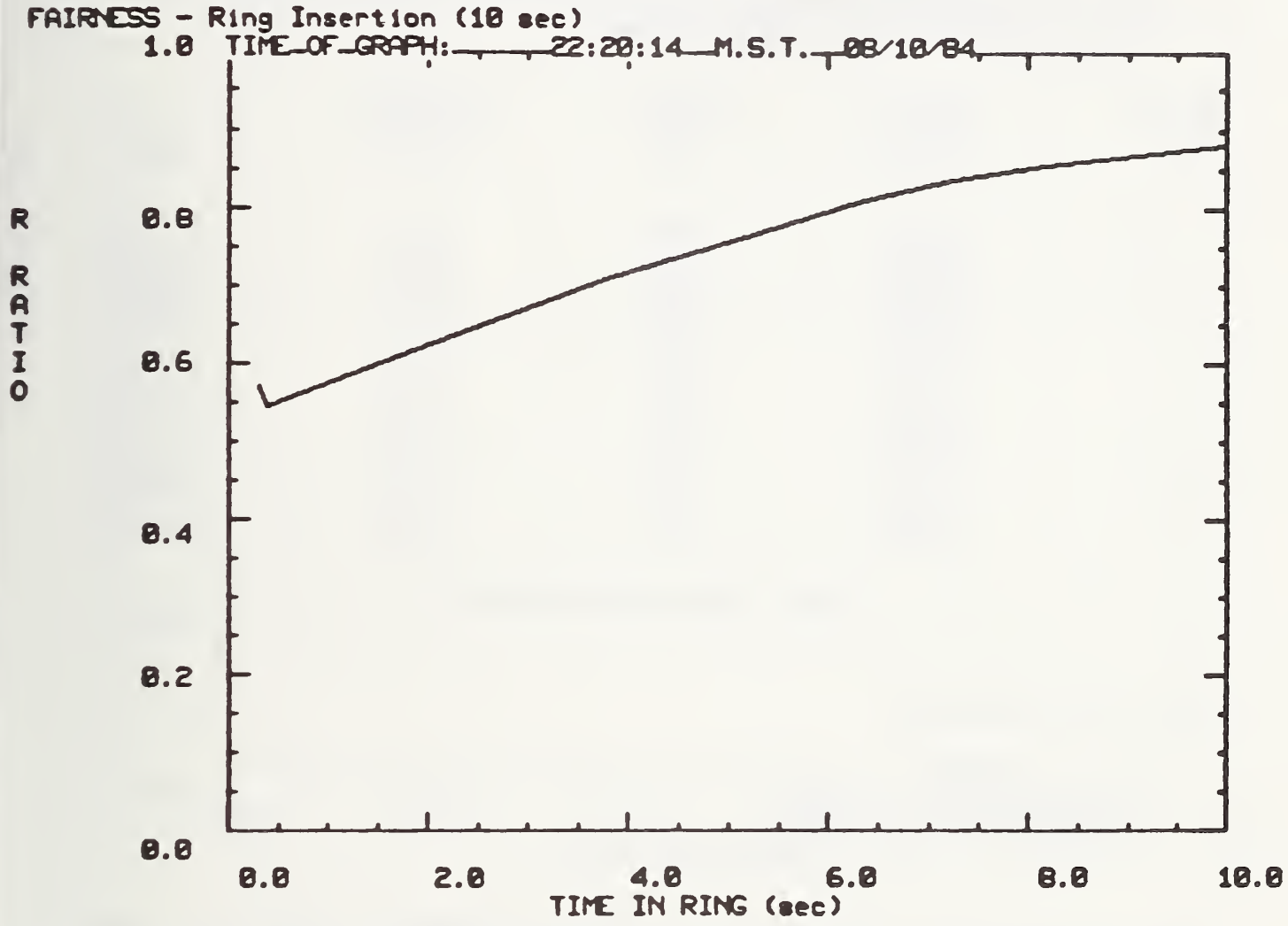


Figure 6 — Fairness — Ring Insertion (10 sec)

To better examine the behavior of the system, we extend our experiment to 120 seconds simulated time and gather the following results.

offered load = 154.55%
network throughput = 91.32%
average acquisition delay = 22467.865 msec

Station No.	Time in ring (sec)	Offered load	Station throughput	R ratio
8	120.000	16.00	9.65	0.602979
4	120.000	16.00	9.65	0.602875
9	119.098	15.88	9.53	0.599929
2	118.181	15.76	9.40	0.596801
6	117.253	15.63	9.28	0.593719
5	116.362	15.51	9.16	0.590538
1	113.784	15.17	8.83	0.582032
10	113.784	15.17	8.83	0.582032
3	110.370	14.72	8.50	0.577616
7	110.308	14.71	8.50	0.577703

Table 3: Ring Insertion (120 sec)

Statistics for the R ratio:

mean = 0.590622
standard deviation = 0.01012
coefficient of variation = 0.01714

The variance is lower, demonstrating that fairness can probably be attained if enough time is allowed for the system to become saturated and no other nondeterministic factors are introduced. See Figure 7 for details.

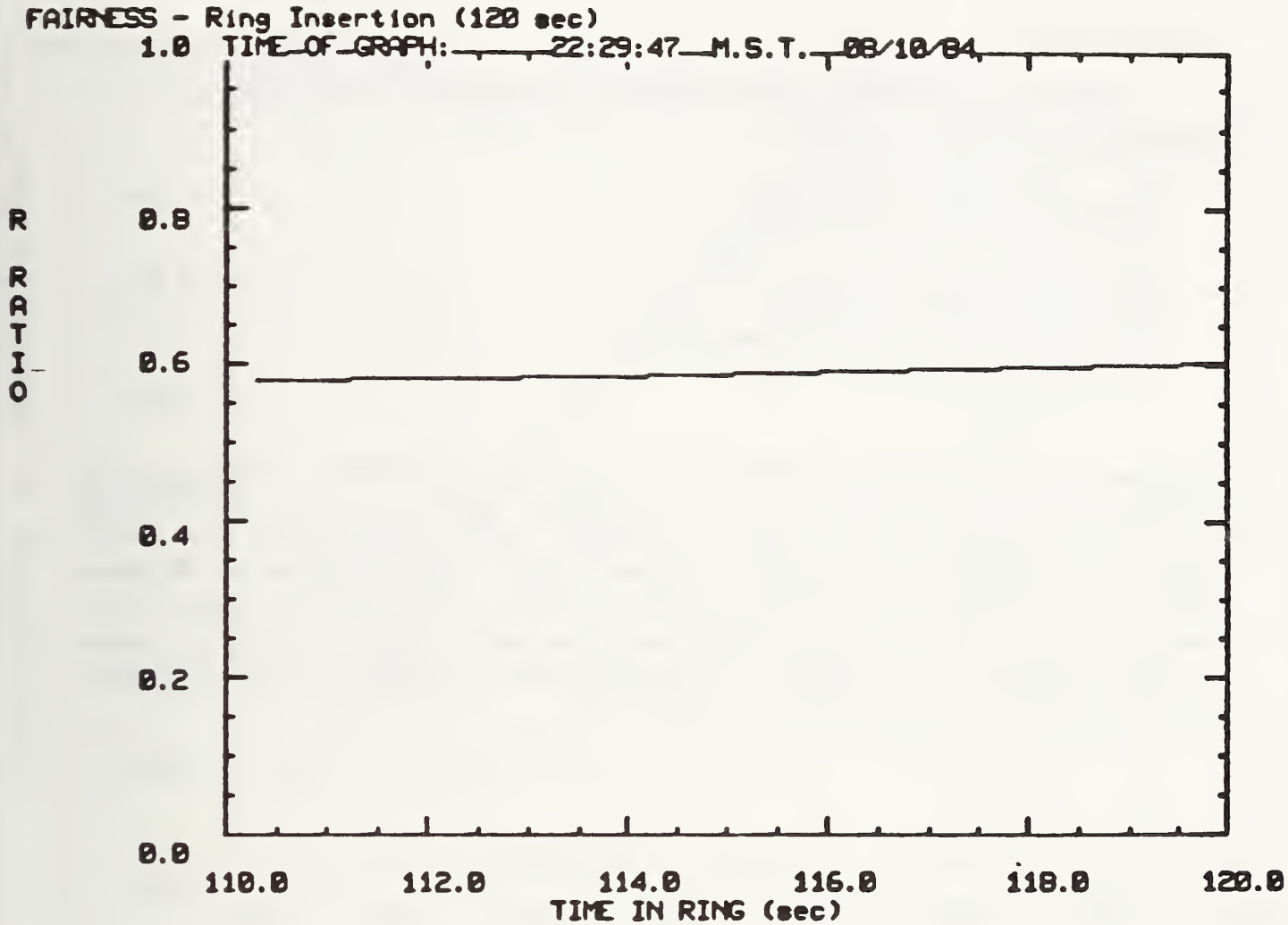


Figure 7 — Fairness — Ring Insertion (120 sec)

4.3 Data Length

The effect on performance of different data lengths is now investigated. The constant parameters are as follows:

frame arrival type = POISSON
simulation time = 10 seconds
station delay = 100 microseconds
slot time = 250 microseconds
token hold timer = 4000 microseconds
max_inter_solicit_count = 100

The simulated configuration is 100 active stations sending frames. The interarrival rate at each station is varied to produce total offered load in the range of 25% to 150% of channel bandwidth. The network throughput and acquisition delay are plotted against total offered load as shown in Figures 8 and 9. As a comparison, results were collected for 100, 250, 500, 750 and 1000 bytes of data. We notice that bigger frames are transmitted with much greater efficiency than smaller frames. There is a reason for this. As a frame gets longer, the fixed overhead of preamble and control information (23 bytes in our case) becomes relatively less.

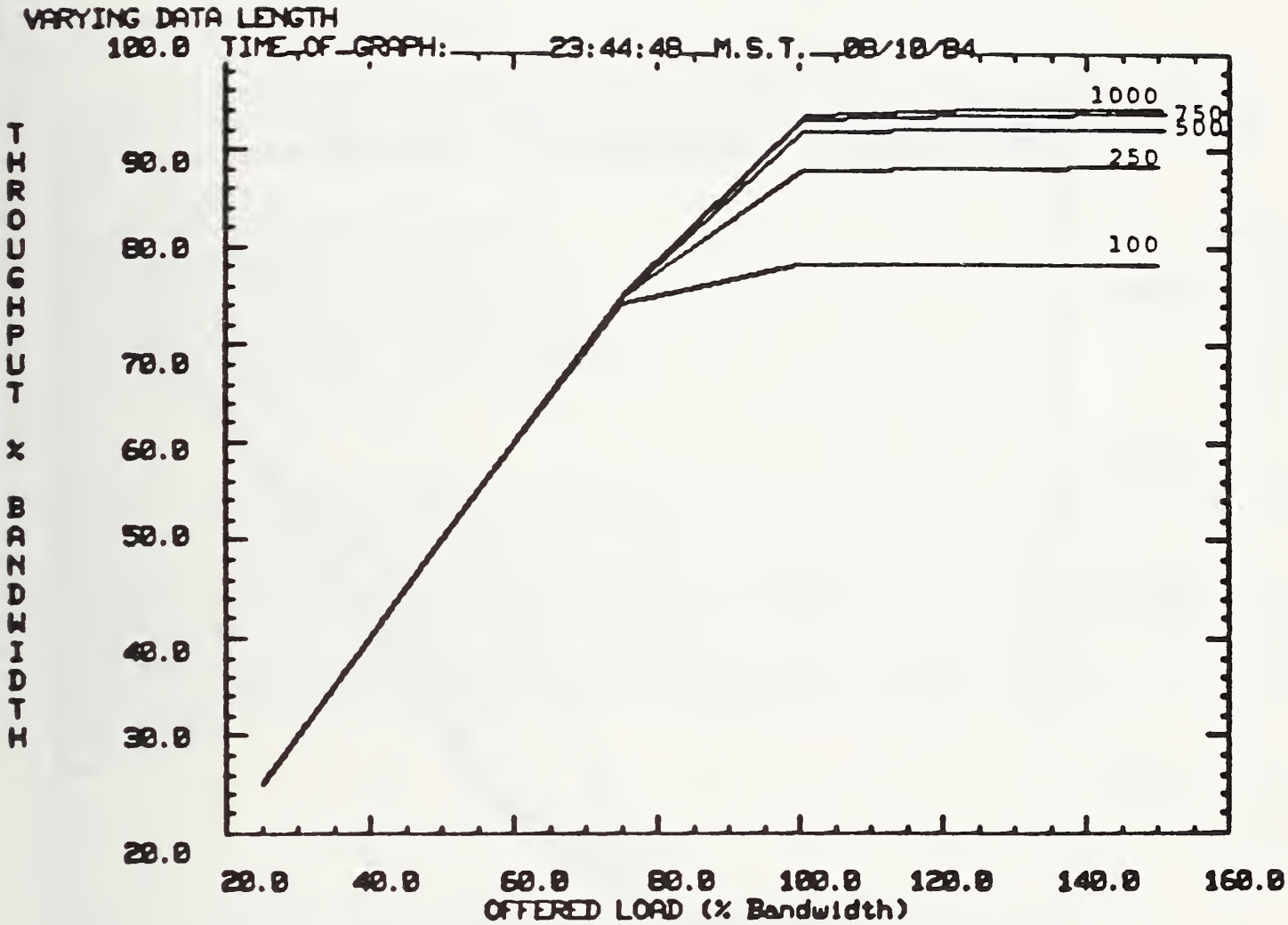


Figure 8 — Varying Data Length
(Throughput versus Offered Load)

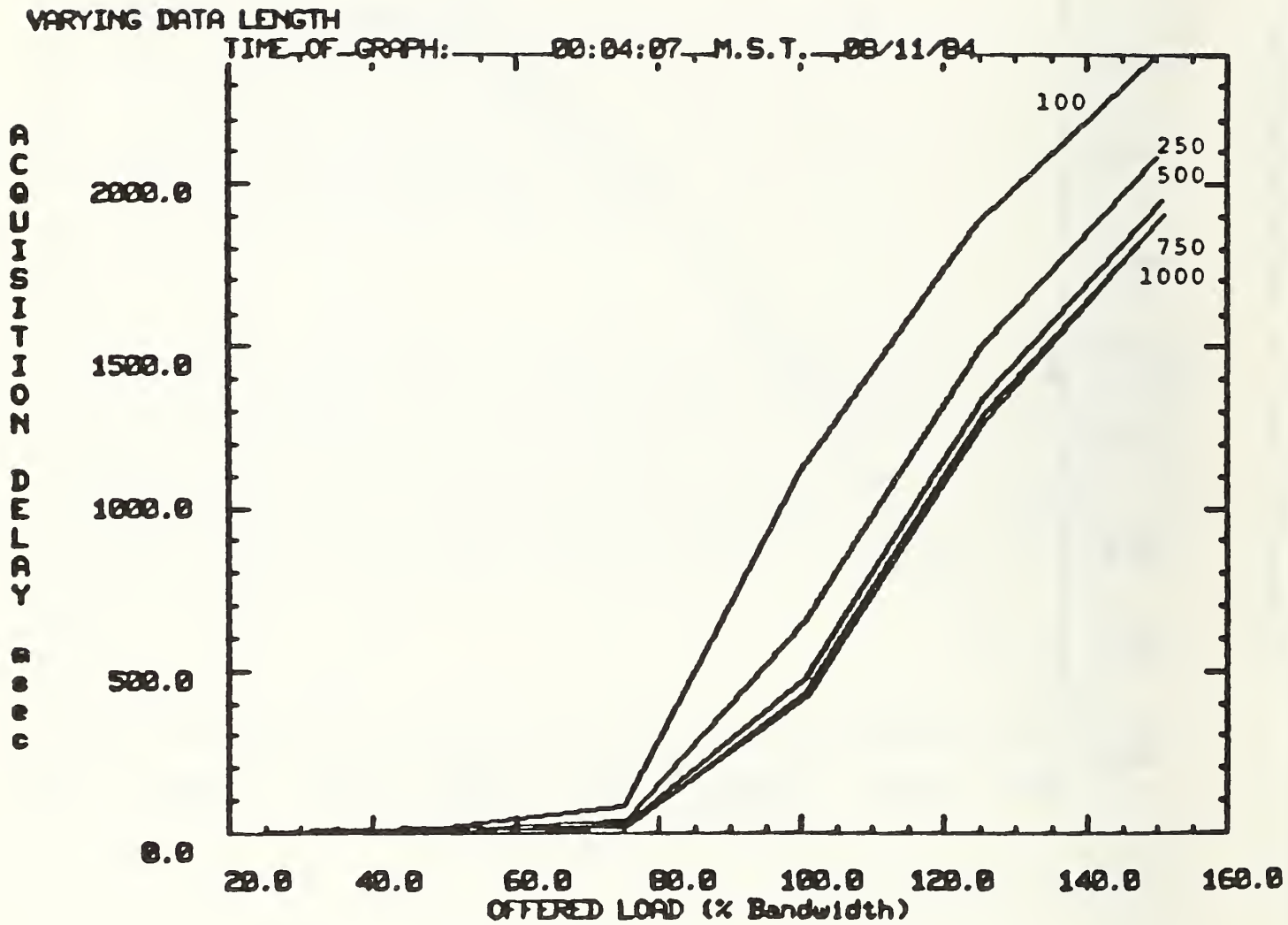


Figure 9 — Varying Data Length
(Acquisition Delay versus Offered Load)

4.4 Number of Active Stations

The performance characteristics of the system for varying network sizes is investigated in this section. In particular, networks with 10, 100, 200, 300 and 400 active stations are considered. Again, the constant parameters are as follows:

frame arrival type = POISSON
simulation time = 10 seconds
station delay = 100 microseconds
slot time = 250 microseconds
token hold timer = 4000 microseconds
Max_inter_solicit_count = 100

For all configurations, the stations are homogeneous in that they all transmit frames with a data length of 500 bytes. Again, we vary the interarrival rate at each station to produce total offered load in the range of 25% to 150% of channel bandwidth. Measurements were made of network throughput and acquisition delay for varying total offered loads. The results are plotted in Figures 10 and 11. We notice that the smaller size network performs better than the larger network with the offered load fixed. The larger network wastes more bandwidth in transmitting the token and maintaining the logical ring.

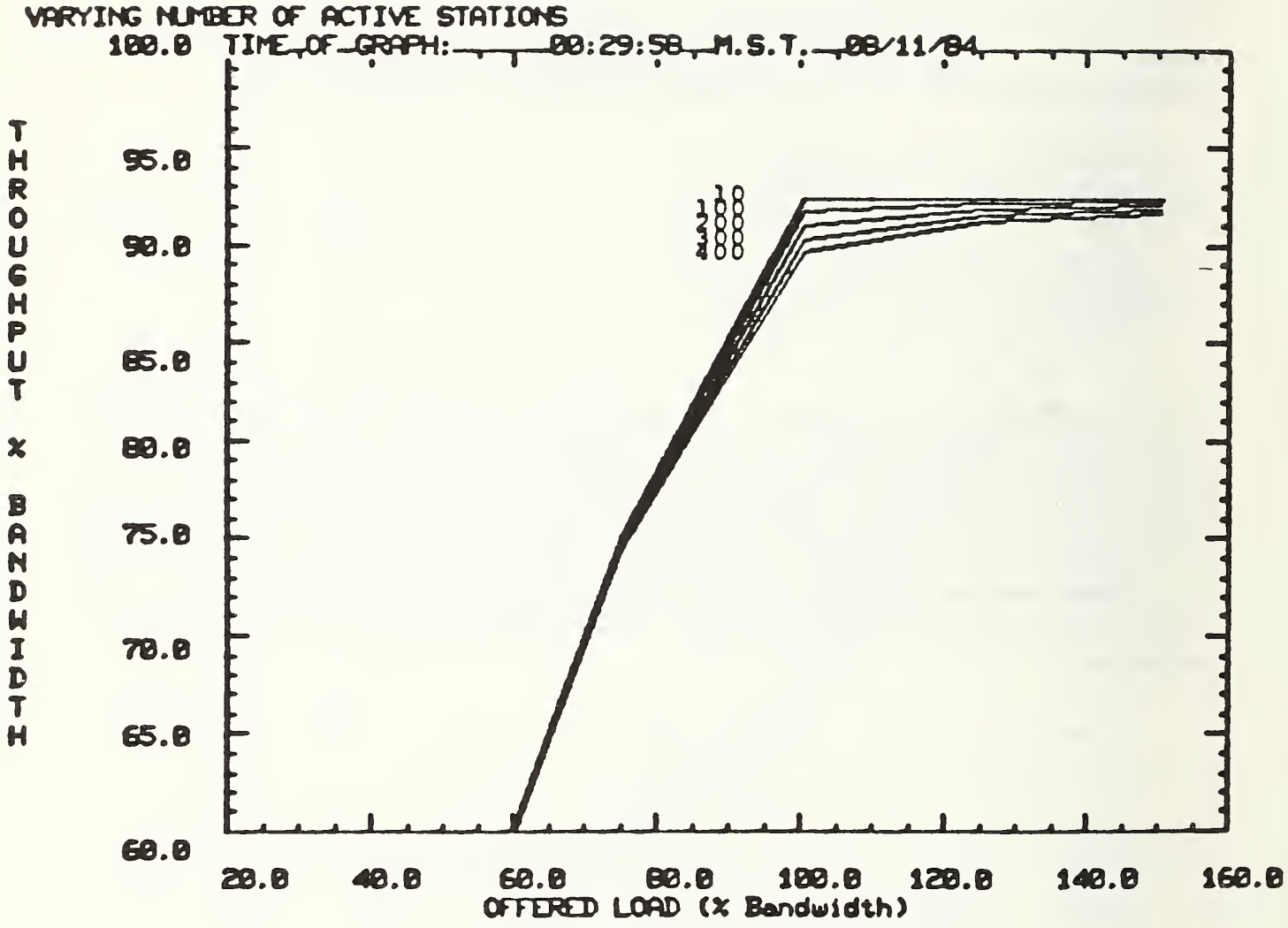


Figure 10 — Varying Number of Active Stations
(Throughput versus Offered Load)

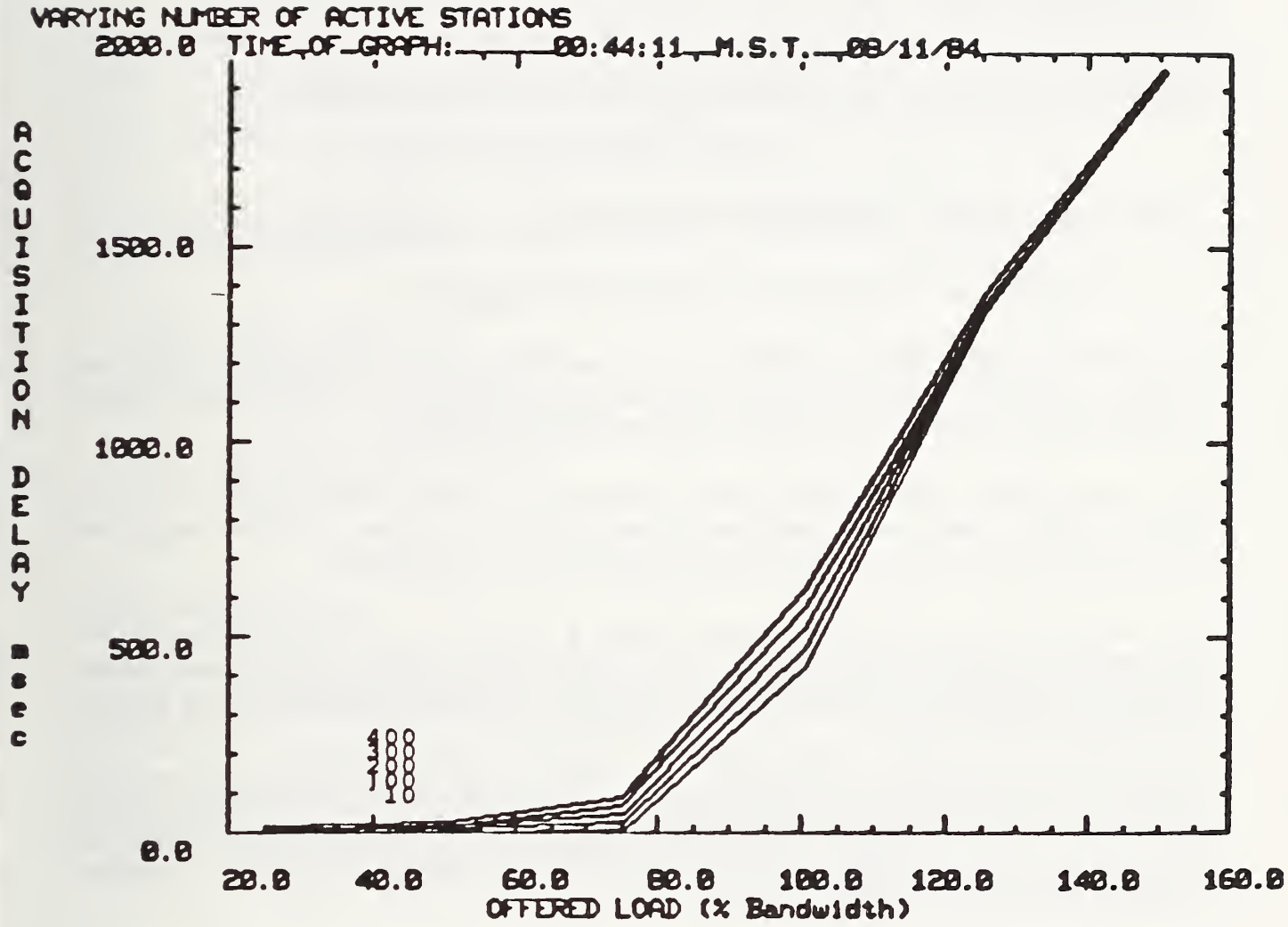


Figure 11 — Varying Number of Active Stations
(Acquisition Delay versus Offered Load)

4.5 Acquisition Delay

As a frame is processed by the model, three time statistics are collected:

- start time - time of arrival in a station's transmit queue
- access time - time the first bit of the frame is transmitted on the cable
- finish time - time frame is received at the destination

From these figures, various statistics can be calculated. For instance, the acquisition delay is defined to be the difference of access time and start time. This latency usually consists of two portions, the queuing delay and the token delay.

The queuing delay is measured from the time when a frame arrives at the end of the FIFO transmit queue until it reaches the beginning of the queue. In other words, the arriving frame has to wait for all those ahead of it to be transmitted.

The token delay is the average length of time a station has to wait for a free token. This latency is stable and bounded by the worse token rotation delay which can be determined by the size of the network, station delay, token hold time, ring maintenance timer, etc.

However, the queuing delay is nondeterministic and heavily relies on the offered load of the system. With an overload situation, we have observed that a frame experiences infinite delay as the throughput approaches the channel capacity and when the system has backlog of packets waiting to be sent.

Refer to Figures 9 and 11 for details. In general, we note that the acquisition delay is sensitive, unstable, and degrades sharply as the load increases.

4.6 Token Hold Time

The token hold timer aims at regulating channel access in such a way that no single station will monopolize the ring. As a first attempt, let us assume that all active stations transmit every data frame with an access class of 6. The effect on performance of different token hold times is investigated. We have uncovered some interesting peculiarities which are summarized in this section.

The simulated configuration is 100 active stations transmitting frames. The constant parameters are as follows:

frame arrival type = POISSON
data length = 500 bytes
simulation time = 10 seconds
station delay = 100 microseconds
slot time = 250 microseconds
max_inter_solicit_count = 100

4.6.1 Moderate Load

We begin our experiment with a moderate system. The interarrival rate at each station is 18.75 frames per second, generating a load of 75% of the channel capacity. The network throughput and acquisition delay are plotted against token hold time (Figures 12 and 13). We notice that frames are transmitted with greater efficiency for longer token hold timers. In fact, increasing the length of the token hold timers does not worsen performance. This is because a station will release its token when it finishes transmitting. The token will not be held until the exhaustion of the hold timer.

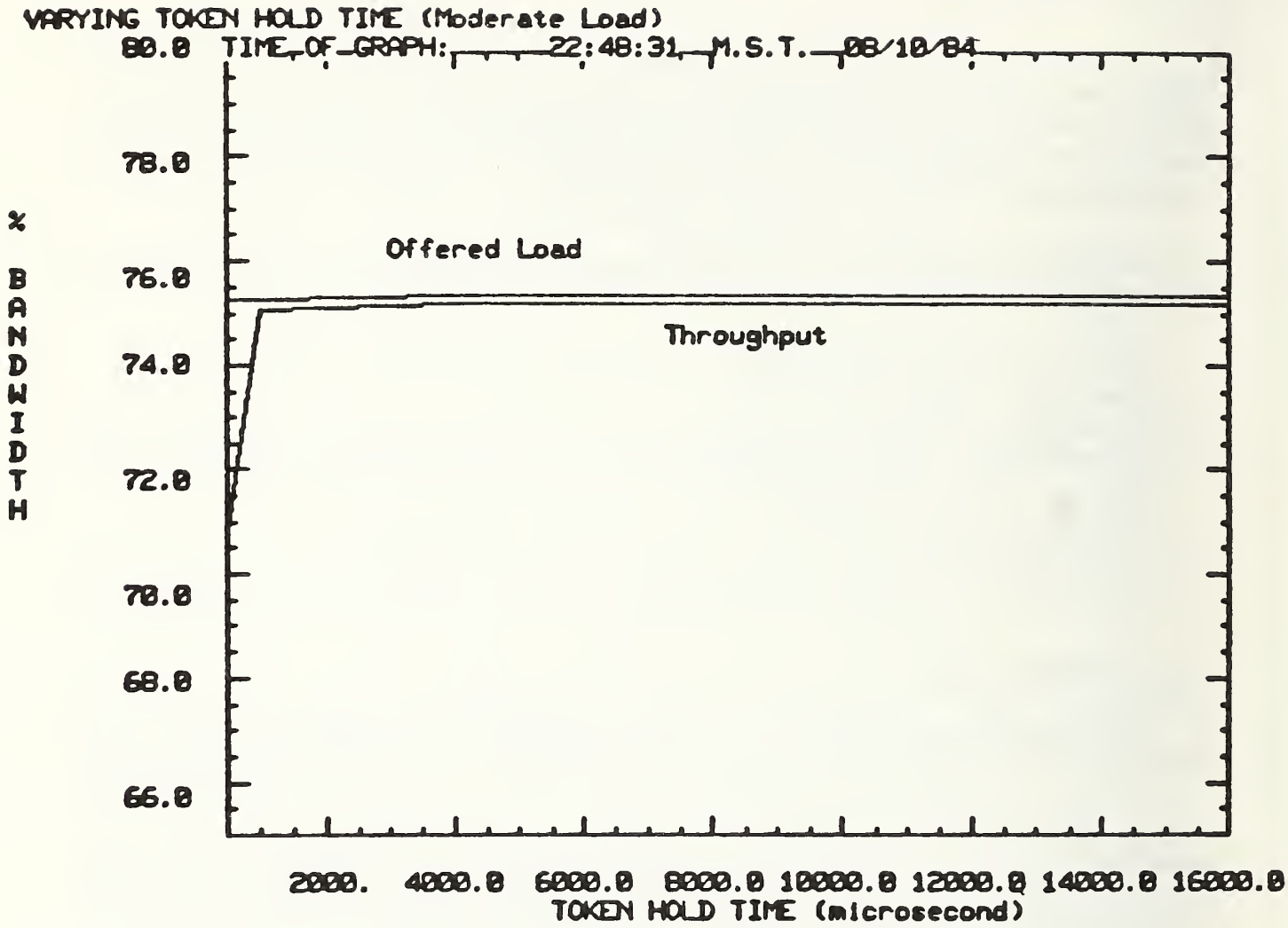


Figure 12 — Varying Token Hold Time (Moderate Load)
(Throughput versus Token Hold Time)

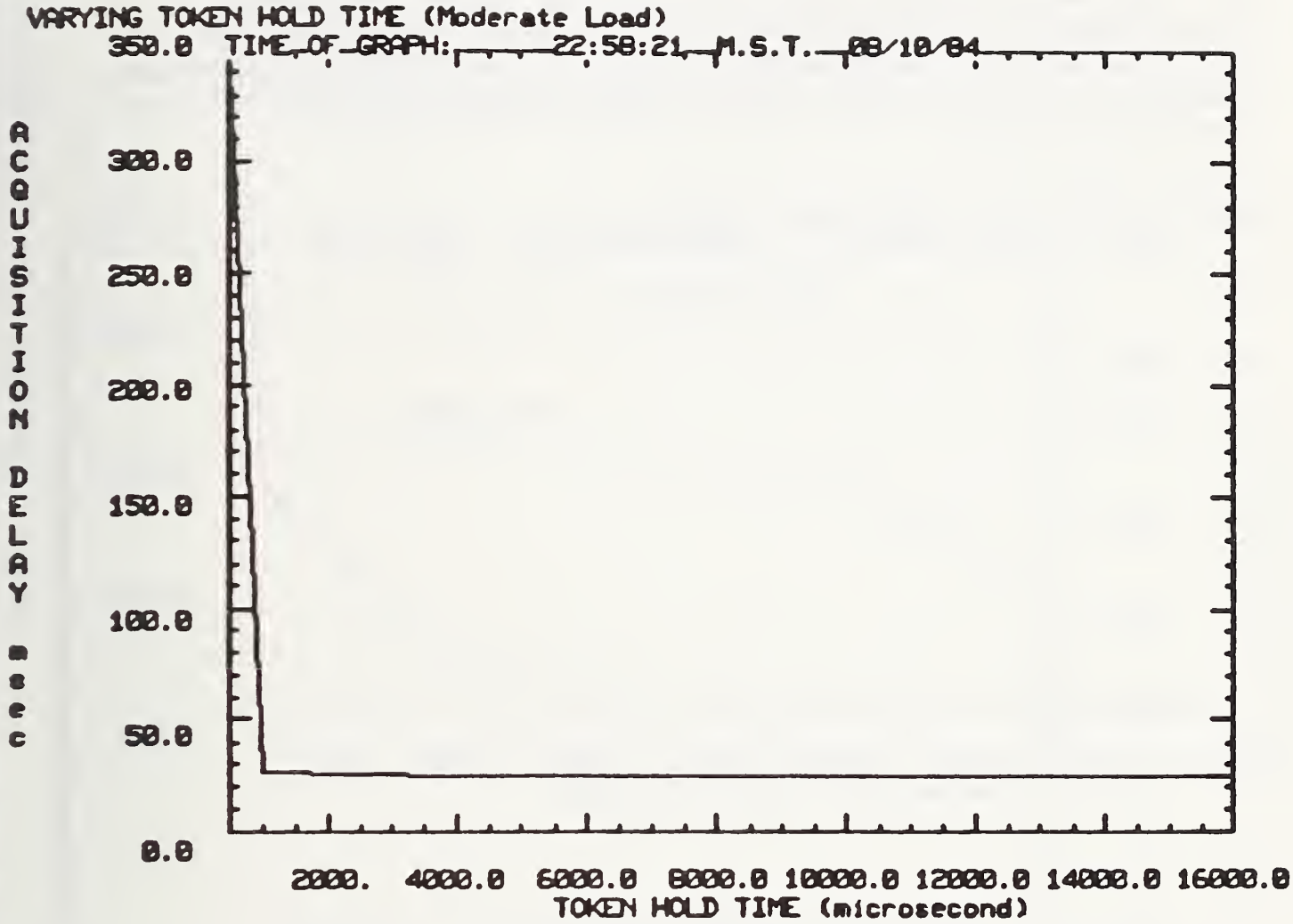


Figure 13 — Varying Token Hold Time (Moderate Load)
(Acquisition Delay versus Token Hold Time)

4.6.2 Heavy Load

Our second experiment deals with an overload situation. The interarrival rate at each station is 25 frames per second, generating a load of 100% of the channel capacity. The results are plotted in Figures 14 and 15. The performance improves by lengthening the token hold time until the backlog of frames has been sent. The system will be saturated then.

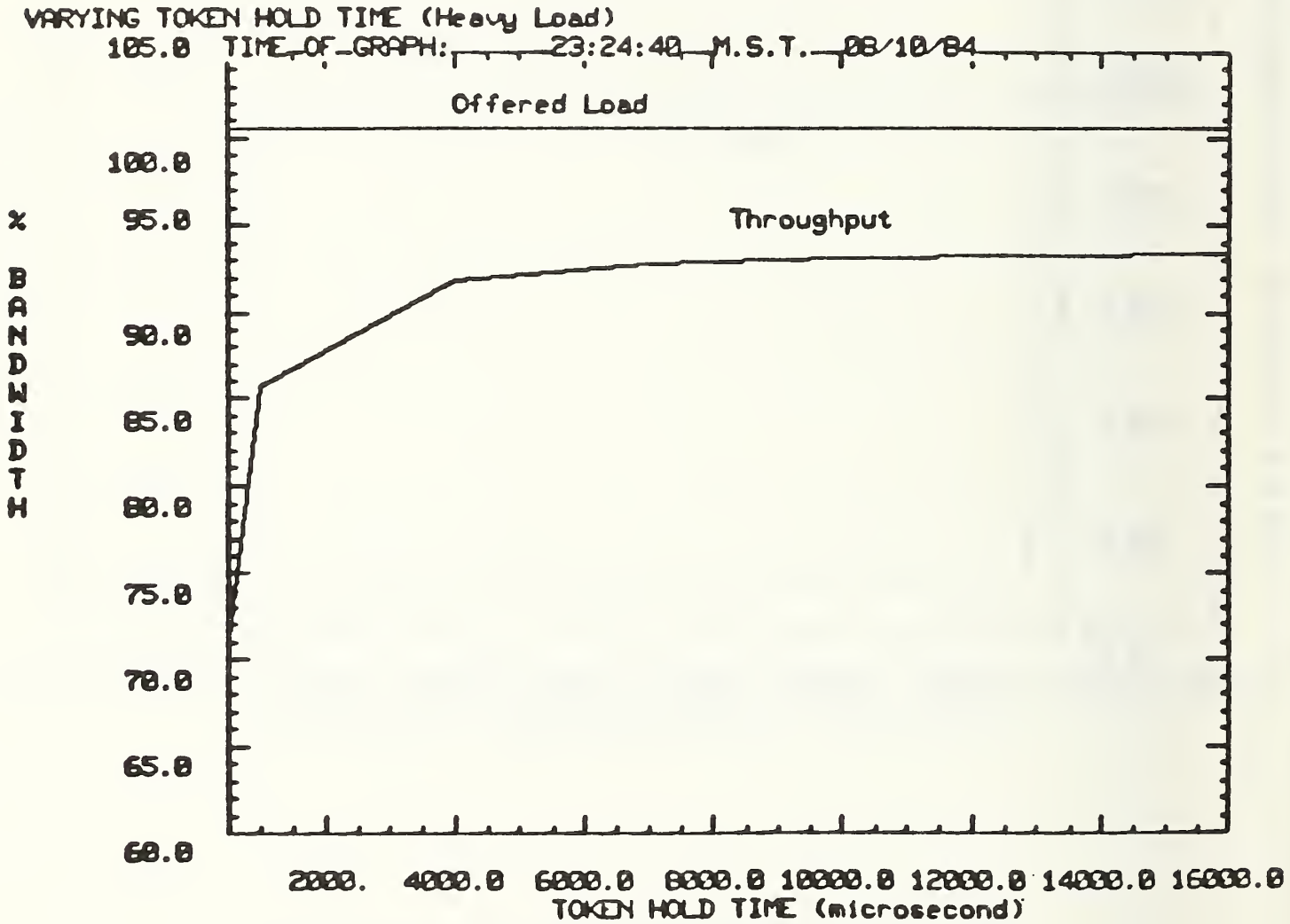


Figure 14 — Varying Token Hold Time (Heavy Load)
(Throughput versus Token Hold Time)

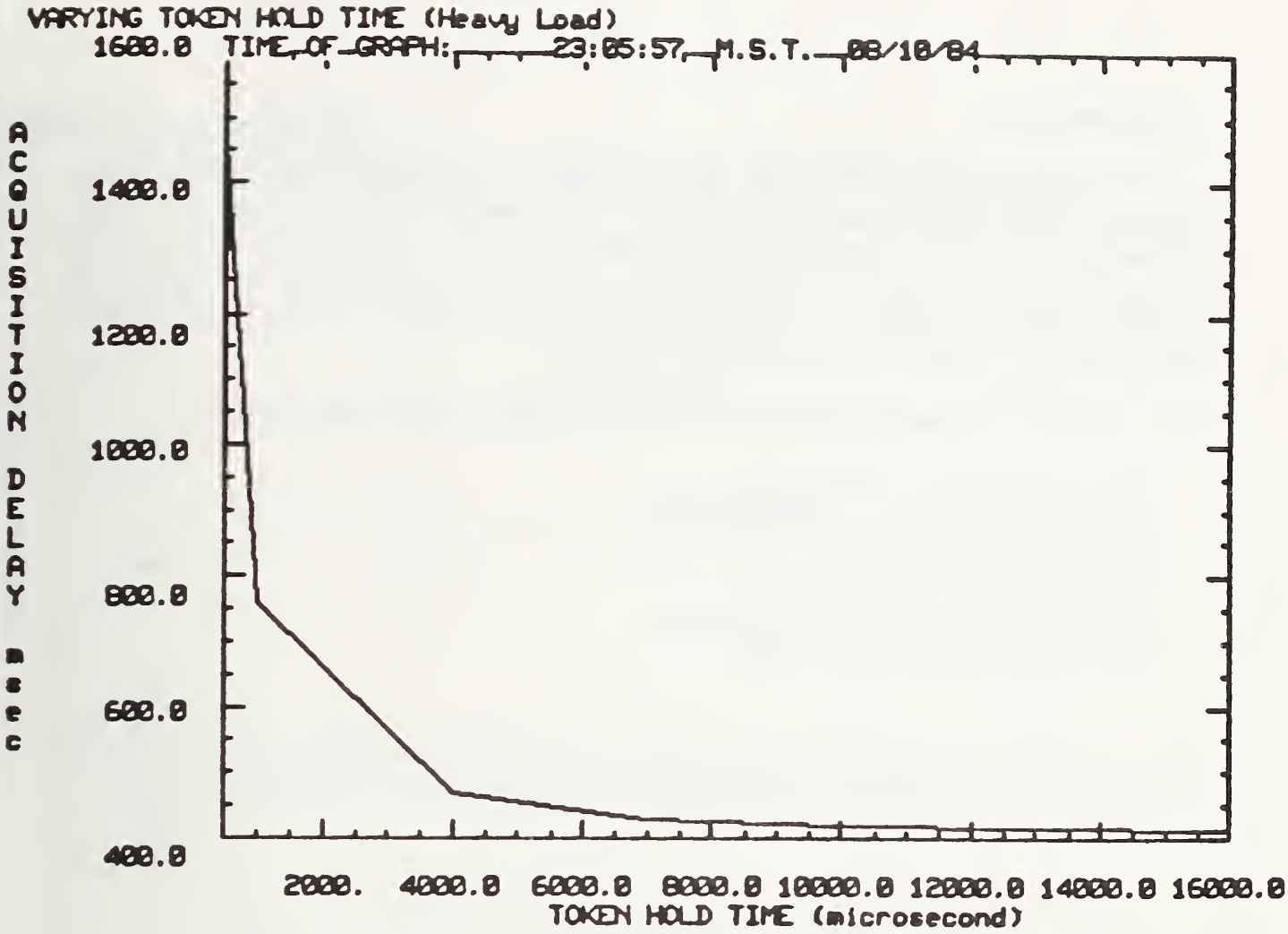


Figure 15 — Varying Token Hold Time (Heavy Load)
(Acquisition Delay versus Token Hold Time)

For our simplified situation, i.e the system is equally loaded by each station and all frames are considered to be high priority, we suggest that exhaustive transmission is more efficient. Channel bandwidth is utilized the most by allowing consecutive transmissions from the same station without passing the token. However, a single busy station monopolizing the channel is definitely a phenomenon that we want to avoid. With the complexity of priority option, the design of such parameters has to be based on the workload and the traffic pattern expected in the system.

4.7 Station Delay

It should not be a surprise that performance degrades as the station delay increases. Recall that acquisition delay is a performance measure of each individual station while network throughput is a perfomance measure of the network. The negative impact of increasing the station delay, which is a critical factor of a station, is obviously shown in Figure 17. Every effort should be made to minimize the station delay and the slot time as defined by IEEE 802.4.

Our configuration consists of 100 active stations. The constant parameters are:

frame arrival type = POISSON
frame interarrival rate = 25 frames/sec
data length = 500 bytes
simulation time = 10 seconds
token hold time = 4000 microseconds
max_inter_solicit_count = 100

The slot time is estimated to be about two times the station delay plus 50 microseconds. The results for varying station delays are plotted in Figures 16 and 17.

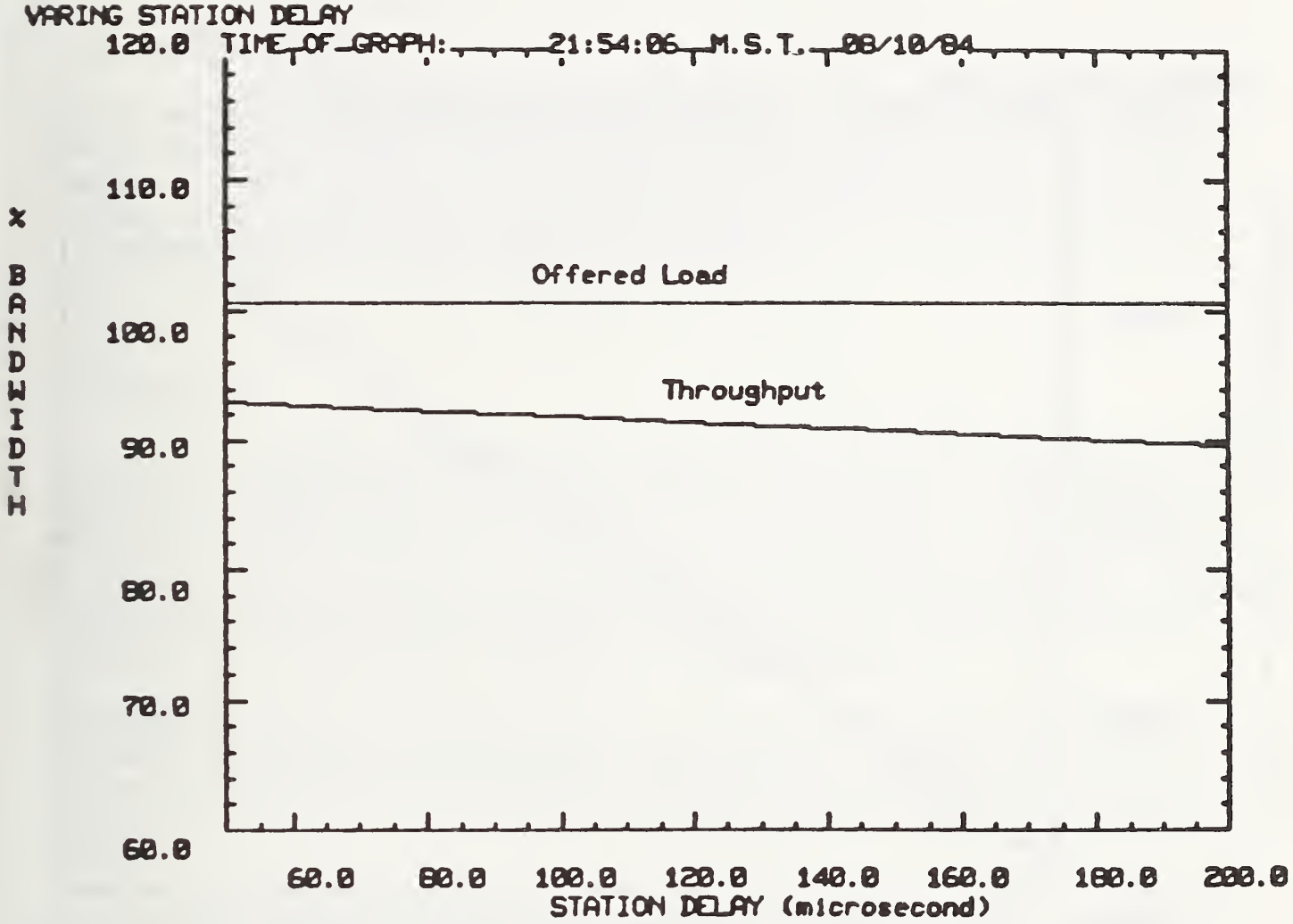


Figure 16 — Varying Station Delay
(Throughput versus Station Delay)

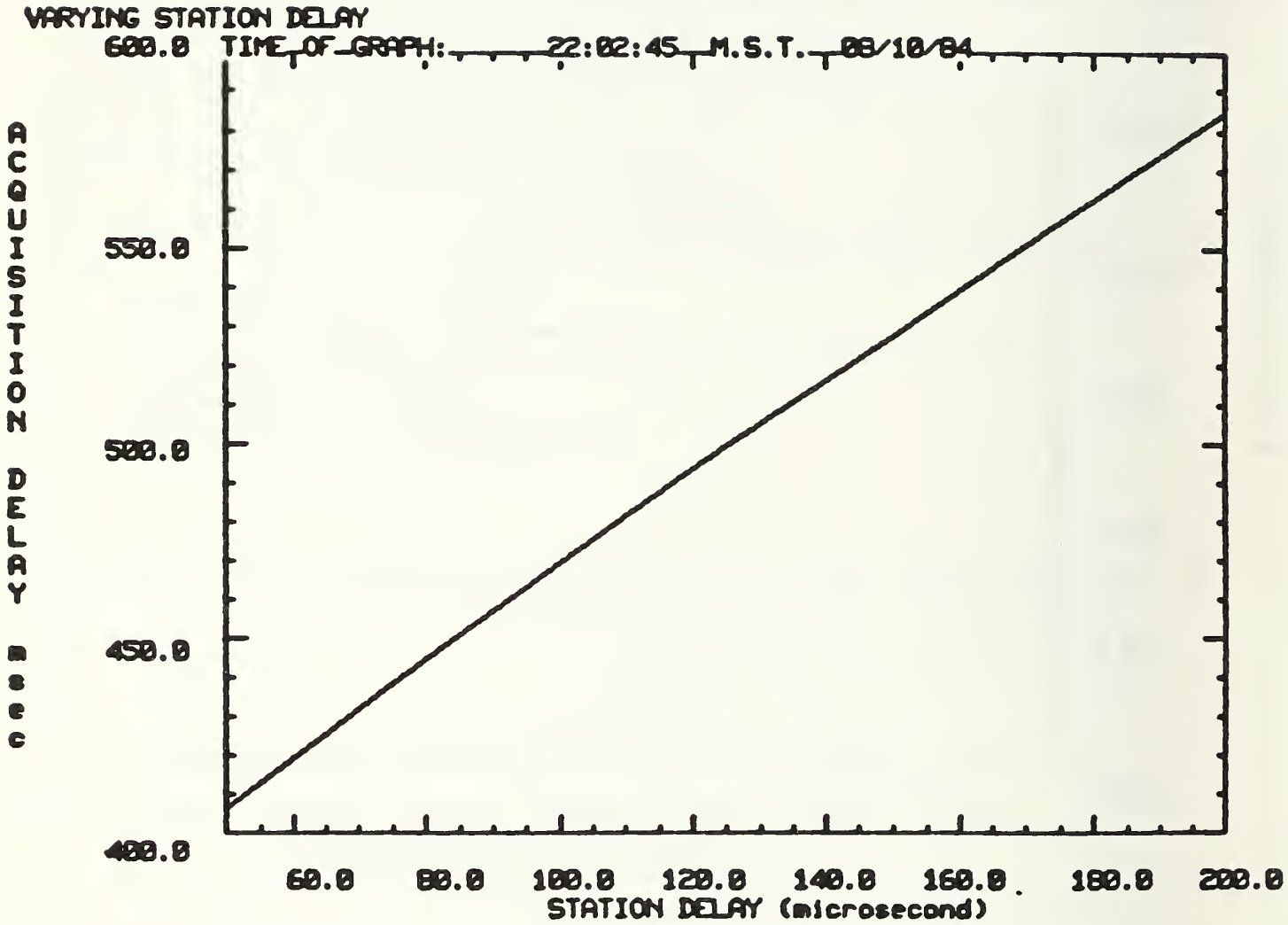


Figure 17 — Varying Station Delay
(Acquisition Delay versus Station Delay)

4.8 Additional Factors

In general, with the offered load fixed, we observe better performance measures for a stable system, where insertion and deletion are not frequent events. One of the reasons is that less bandwidth is wasted in processing these procedures. More importantly, the system adjusts itself better to a stable environment. In other words, a regular input process will be better handled by the protocol.

5. Conclusions

This paper has investigated performance characteristics of the 802.4 token bus protocol. A comprehensive discussion on how the performance of the network is affected by various system parameters has also been included. The intent of the present study is not to predict the performance of any particular real network. Our attempt is to characterize the behavior of the system. It is our hope that this performance analysis can be used as a basis for understanding various aspects of the protocol and serve to stimulate further interest and discussion in this area.

Due to the complexity of the system studied, discrete event simulation was determined to be the appropriate modeling technique. The emphasis of the paper is the discussion of the results. Therefore, the discussion of the methodology was brief. It is necessary, however, that the reader understands the methodology to put the results in proper context.

The insight we have for the token bus protocol is in accordance with results gained from this performance study. Nevertheless, the simulation results serve to demonstrate and quantify the impact of certain parameters critical for the system. It is, however, difficult to ignore the implication on performance when considering different technologies.

Lastly, it is worth mentioning that the communication traffic considered in this study was of general form, i.e. all participating stations are active and have the same frame interarrival rate. The designer of a local area network must understand the specific communication traffic and workload to be able to make intelligent decisions.

References:

[BUX81] Bux, Werner, "Local-Area Subnetworks: A Performance Comparison", IEEE Transactions on Communications, Vol. Com-29, No. 10, pp. 1465-1473, Oct. 1981

[IEEE82] IEEE Standard 802.4 Token-Passing Bus Access Method and Physical Layer Specifications - Draft F, July 1984

[LARS80] Larsen, R. L., Agre, J. R., and Agrawala, A. K., "A Comparative Evaluation of Local Area Communication Technology", Computer Performance Evaluation Users Group (CPEUG) 16th Meeting (NBS-SP-500-65), pp. 87-97, Oct. 1980

[RAHI83] Rahimi, S. K. and Jelatis, G. D., "Simulation Modelling of IEEE 802 Token Bus Media Access Control Protocol", Proceedings of Workshop Performance and Evaluation of Local Area Network, pp. 127-154, March 24-25, 1983

[SAUE81] Sauer, Charles H. and Chandy, K. Mani, "Computer Systems Performance Modeling", Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632

[SHOC79] Shoch, J. F. and Hupp, J. A., "Performance of an Ethernet Local Network: A Preliminary Report", Proceedings of the Local Area Communications Network Symposium, pp. 113-123, May 1979

[SWEE83] Sweeton, David C., "Simulation Results for Factory Floor Networks", Eleventh International Meeting, International Purdue Workshop on Industrial Computer Systems, pp. 713-726, Oct. 4, 1983

[TOEN83] Toense, Robert, "Performance Analysis of NBSNET", Proceeding of Performance and Evaluation of Local Area Network, pp. 7-26, March 24-25, 1983

PERFORMANCE ISSUES OF 802.4 TOKEN BUS LAN'S

prepared for

The Workshop on Analytical Modeling of IEEE 802.4 Token Bus
Session III: Performance Issues
April 29-30, 1985

by

Bruce A. Loyer
Motorola SPS
Phoenix, Arizona

Doron Kolton
Motorola MSIL
Tel Aviv, Israel

ABSTRACT

This paper presents curves generated via a software simulator that deal with several aspects of 802.4 Token Bus performance. The areas considered include dependence on station address allocation, the number of stations, the cable length, the frame length, the number of stations transmitting, and the token hold time. A brief description of the simulator is first presented and each area of performance impact is then discussed.

OVERVIEW

As part of its program to provide 802.4 products, Motorola developed a network simulator. The Token Bus Network Simulator is a software tool used with the specific objectives of:

1. 802.4 protocol verification and development through identification of deadlock, error, and failure conditions.

2. Aiding in the configuration and fine-tuning of token bus networks.

3. Evaluating network performance.

4. Aiding in the design of VLSI devices.

Coded in Pascal, it is a discrete event driven simulator providing predictions of delay, throughput, and many other performance measures as a function of offered load. By varying system parameters, it is possible to characterize token bus network performance through a series of simulation runs. In the following paragraphs, the effects of various parameters are presented and the following terminology is used:

Offered load	= % of bandwidth requested by data
Throughput	= % of bandwidth consumed by data
THT6	= Priority 6 token hold time
TRT	= Token rotation time
Queuing length	= Time from data request to time data actually sent

DEPENDENCE ON FRAME LENGTH

The maximum length of data frames transmitted is a parameter negotiated at the Transport Layer and may be affected by the user's buffer management structure (buffer size and allocation). It is

therefore important to know how the frame length impacts throughput.

An intuitive guess would be that larger frame sizes would provide best performance because there is less overhead in terms of headers, etc. Simulation shows that, independent of token hold time, throughput increases with increasing frame length until a limit is reached somewhere between 512 octets and 1024 octets. Above the 1024 octet frame size, throughput is virtually the same clear up to the 802.4 limit of 8191 octets. Exhibit 1 shows throughput rolloff for frame sizes of 512 octets and below, and Exhibit 2 shows nearly identical throughput for frame sizes of 1024 octets and above. These results show that if other system limitations (such as buffer size) require files to be broken up into smaller sizes, it will not impact overall network performance if a frame size of 1k octets is maintained.

DEPENDENCE ON CABLE LENGTH

Token bus networks will be used in applications varying from short (less than 100 meters) distance carrier band subnets to a broadband spine covering an entire campus or factory of several thousand meters. Modeling has shown that performance in terms of token rotation time is nearly identical for 1000 meter, 1500 meter, and 5000 meter cables. Exhibit 3 shows that TRT increases sharply above 80% offered load for all cable lengths. The graph shows that cable delay does not add significantly to the token rotation time.

DEPENDENCE ON NUMBER OF STATIONS TRANSMITTING WITH A FIXED NUMBER OF NODES

Network performance can be simulated with a few number of nodes each with a heavy load or with many nodes each with a small load. The latter case is more representative of a real environment. Exhibits 4 and 5 vary the number of stations transmitting on a 100 node system where all nodes are members of the logical ring.

When the stations have a large token hold time and can transmit all of their frames, there is little difference in throughput (Exhibit 4), token rotation time, or quequing delay. When THT6 is short, such as 1 frame only, there is a significant difference (Exhibit 5). The thoughput at 100% offered load is 90% with all 100 nodes transmitting, while it drops to 78% with only 25 nodes transmitting.

DEPENDENCE ON NUMBER OF STATIONS WITH ALL NODES TRANSMITTING

The previous performance examples were based on a fixed number of stations on the cable, however, the number of stations transmitting was varied depending on message load and other factors. To test the impact of the number of transmitting stations, several runs were made with all stations transmitting and varying the number of total stations.

Exhibit 6 shows that system throughput is relatively independent of the number of stations.

Exhibit 7 shows that token rotation time follows expected results in that TRT is 10 times longer for 100 stations than for 10 stations. Note that the token hold time is limited to one frame as would be the case where TRT is desired to be a minimum. Also note that TRT increases drastically under heavy offered load.

Exhibit 8 shows quequing delay. It is surprising to find that quequing was not significantly longer for 100 stations.

DEPENDENCE ON HIGH PRIORITY TOKEN HOLD TIME

The high priority token hold time (THT6) is an important parameter that must be optimized for every network. As this number is increased, one would expect better performance because each station

can transmit more frames per token. One also expects token rotation time to increase which can be a major limiting factor. The actual value of THT6 chosen is a compromise between these two characteristics.

Exhibits 9 and 10 show the relationship between THT6 and TRT. As expected, having a short THT6 gives the worst throughput under very heavy offered load (curve A, Exhibit 9) but the shortest TRT (curve A, Exhibit 10). However, it can be seen in Exhibit 9 that when THT6 is made very long (transmit-all-frames), throughput increases only 5% at 100% offered load (almost zero at lesser loads) while TRT increases over 200% at 100% offered load (Exhibit 10). This indicates that THT6 should be set to a low number (about 2 frames) to keep a short TRT, unless a network has extremely high utilization ()90%) and TRT is not a critical parameter. Additional work is needed in this area to determine the effect of low priority queues.

DEPENDENCE ON ADDRESS ALLOCATION

Because the token bus protocol has an ordered ring, it is possible to assign station address based on location in an attempt to get more performance. Simulation runs were made with addresses aasigned by random allocation, cyclic allocation, and hand allocation.

Throughput was almost identical in all cases (Exhibit 11). However, under very heavy offered load a significant difference is shown in token rotation time (Exhibit 12). TRT can be minimized under heavy loading conditions by intelligently assigning station addresses.

SUMMARY

The simulation results shown in this paper are offered to the user to better understand the attributes of the IEEE 802.4 Token Bus. Special thanks must go to Doron Kolton of Motorola Semiconductor Israel whose special efforts in producing these results made this paper possible.

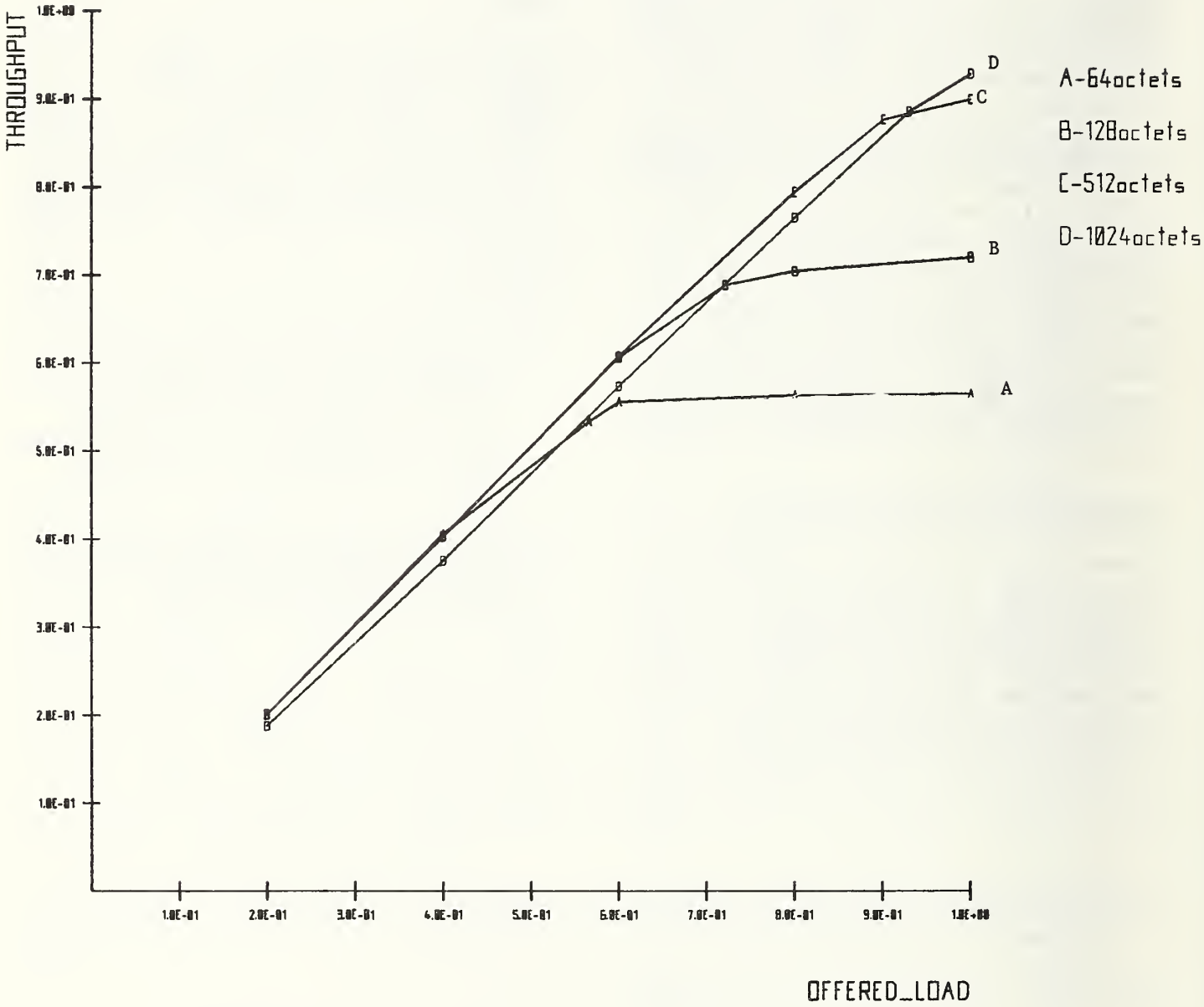
REFERENCES

1. Kermien, Orly, "Using Simulation for the Token Bus Protocol Verification," Workshop on Analytic and Simulation Modeling of IEEE 802.4 Token Bus, Session II, Apr 29-30, 1985.

2. Token-Passing Bus Access Method and Physical Layer Specifications, IEEE Std 802.4-1985, 1985.

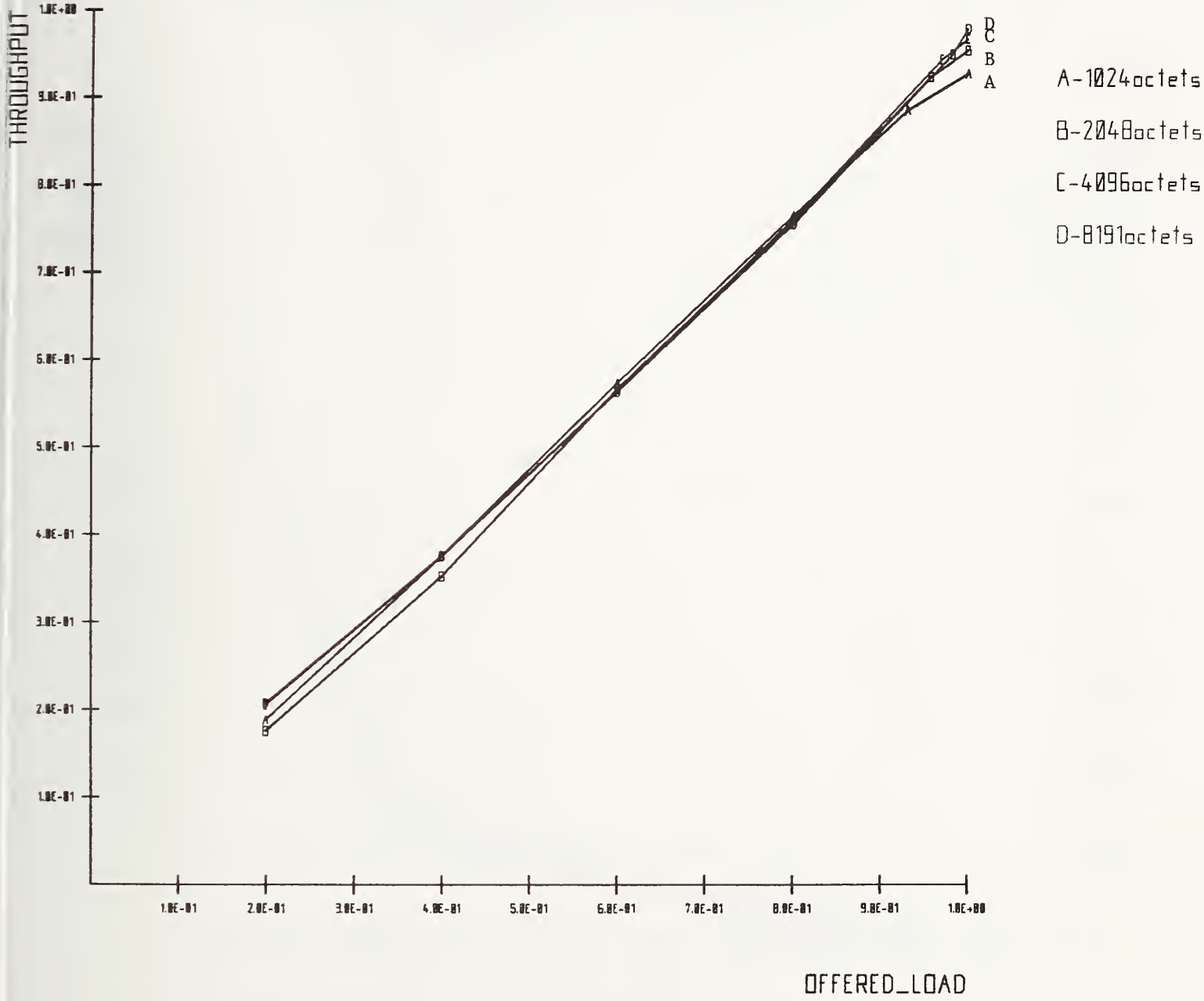
DEPENDENCE_ON_FRAME_LENGTH
100nodes;cabel_length=1500m;
THT6-1farme;address_allocation-cyclic;

EXHIBIT 1



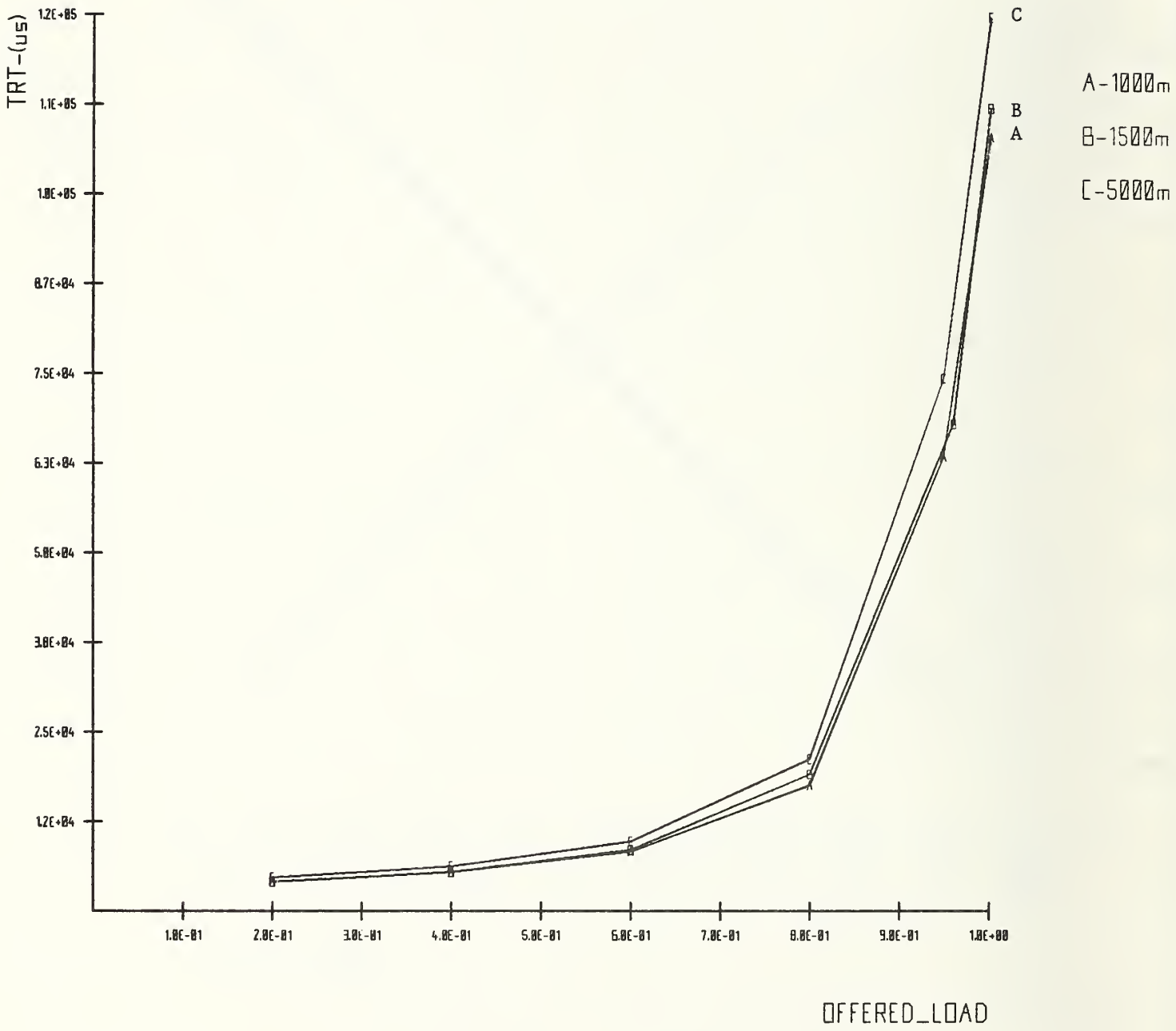
DEPENDENCE_ON_FRAME_LENGTH
100nodes;cabel_length=1500m;
THT6-1frame;address_allocation-cyclic;

EXHIBIT 2



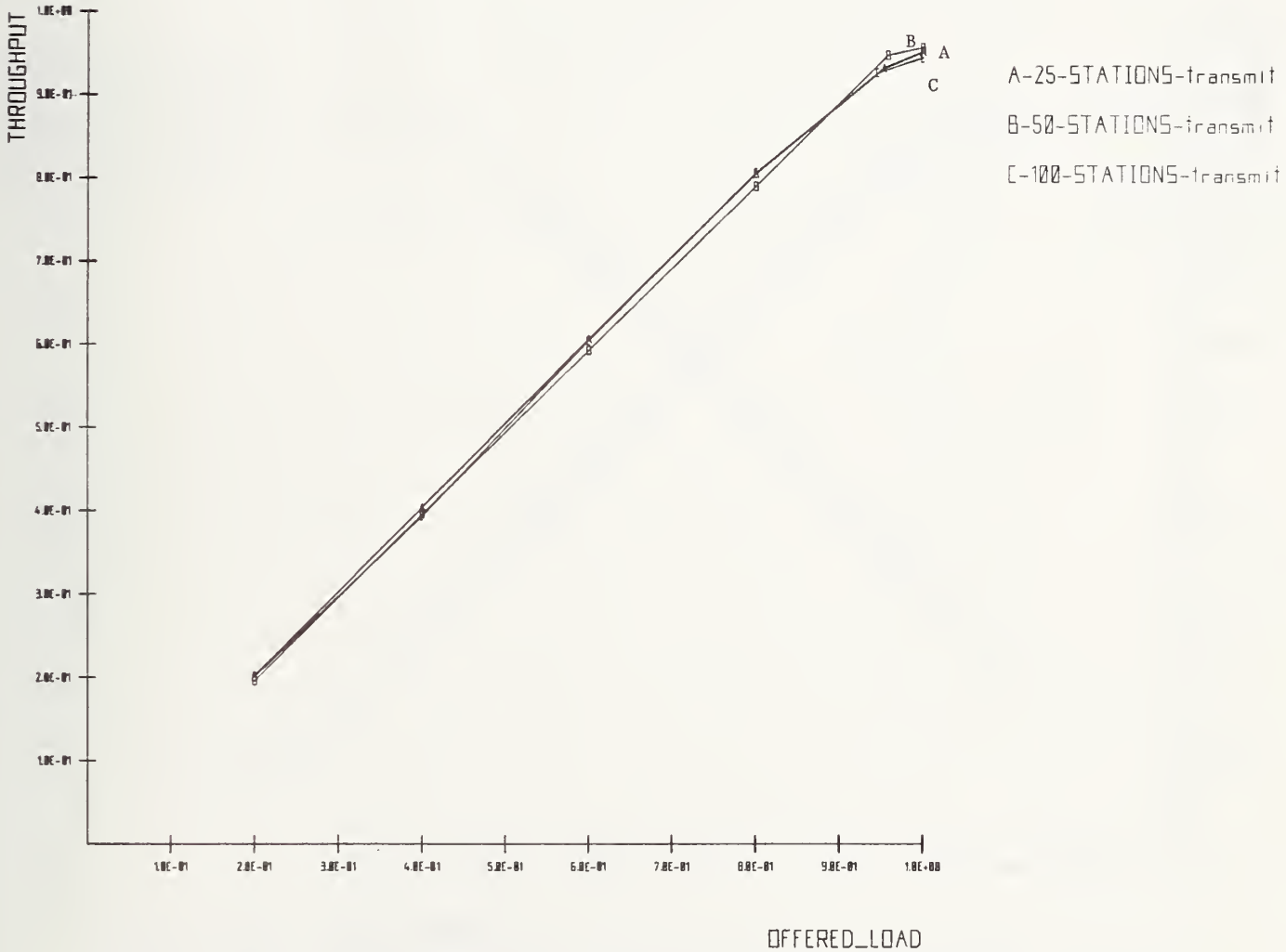
DEPENDENCE_ON_CABLE_LENGTH
100nodes;frame_length=512octets;
THT6-infinite;address_allocation-random;

EXHIBIT 3



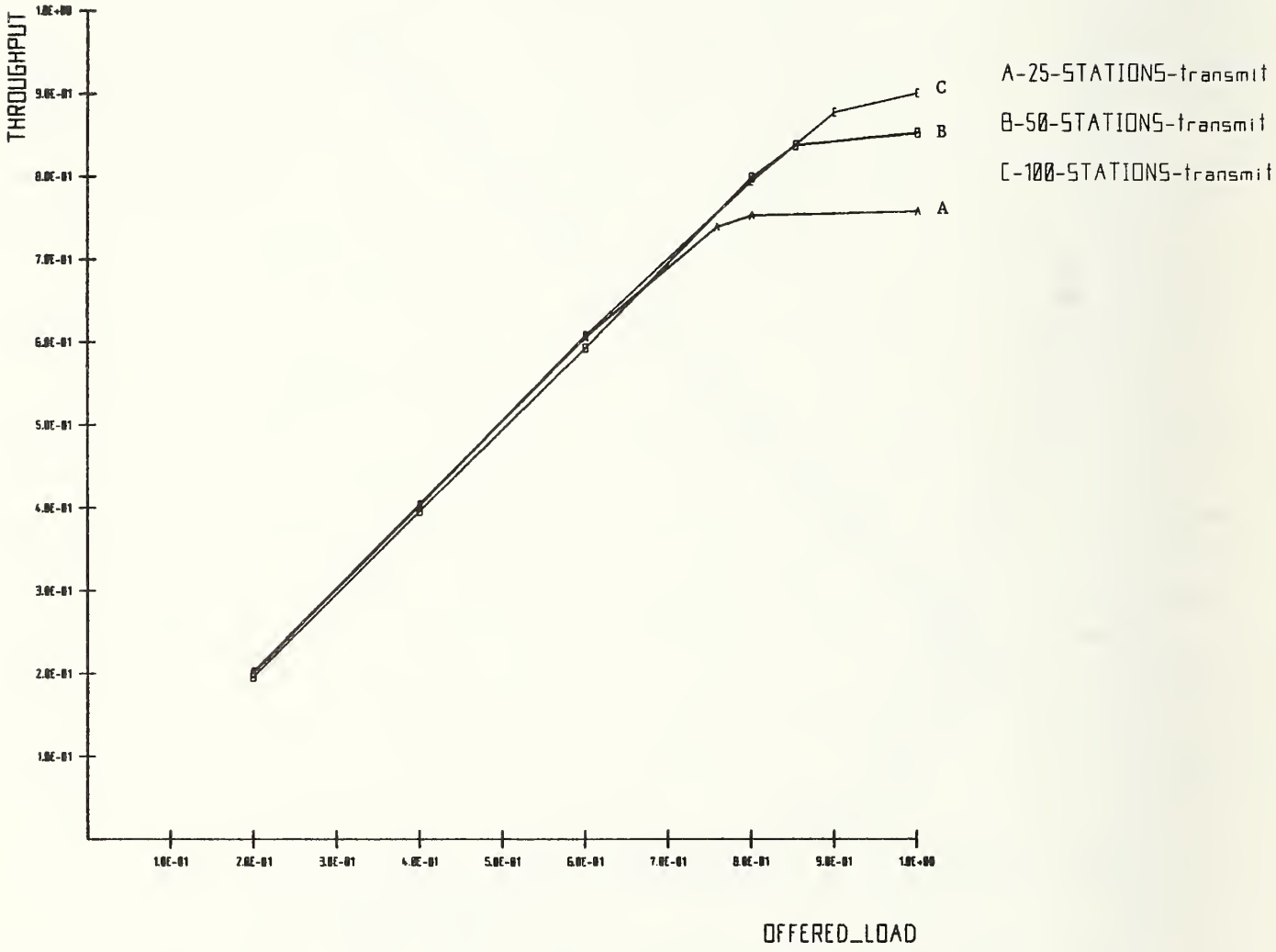
DEPENDENCE_ON_#STATION_TRANSMITTING
100nodes;frame_length=512octets;cabel_length=1500m;
THT6-infinite;address_allocation-cyclic;

EXHIBIT 4



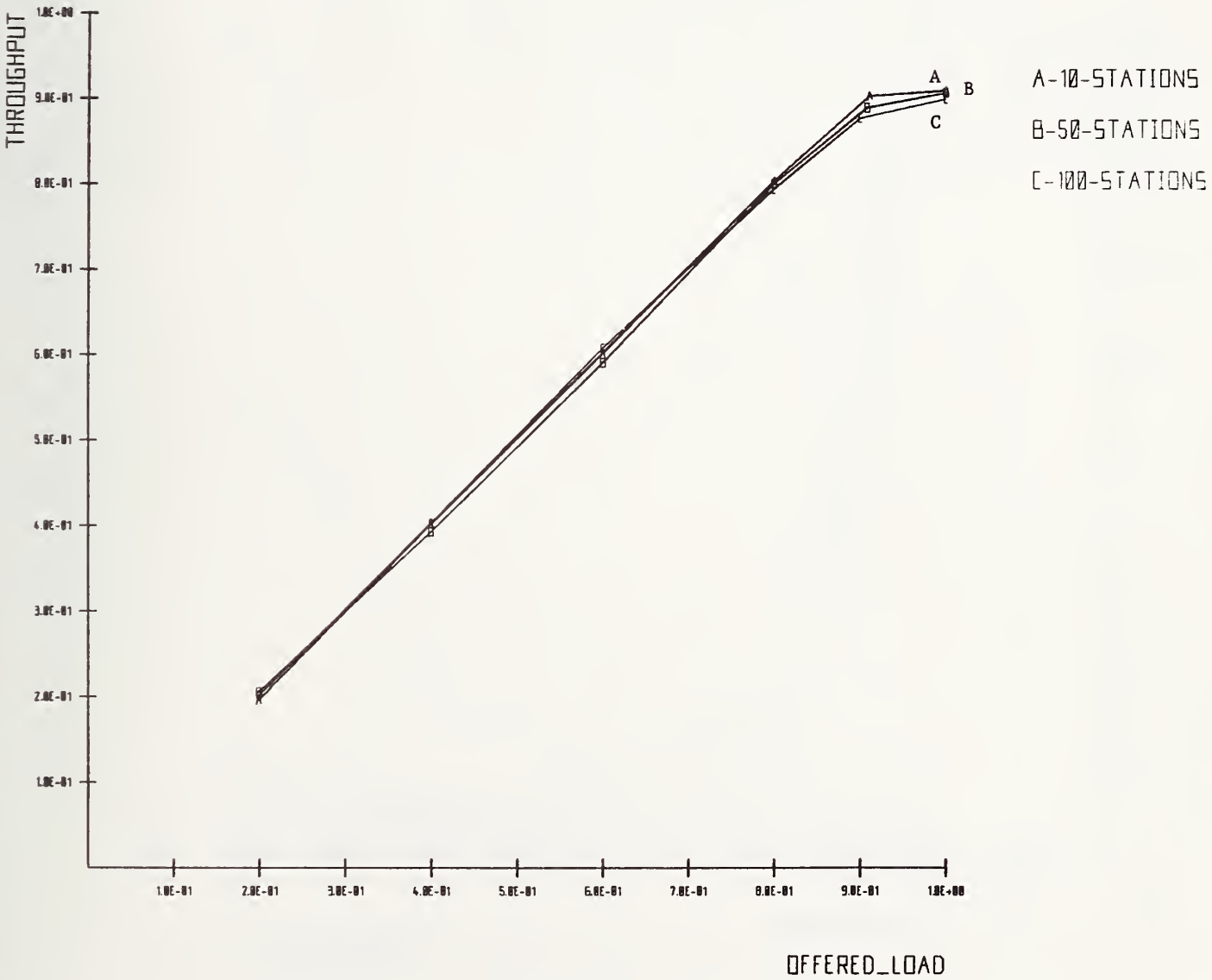
DEPENDENCE_ON_#STATION_TRANSMITTING
100nodes;frame_length=512octets;cabel_length=1500m;
THT6-1frame;address_allocation-cyclic;

EXHIBIT 5



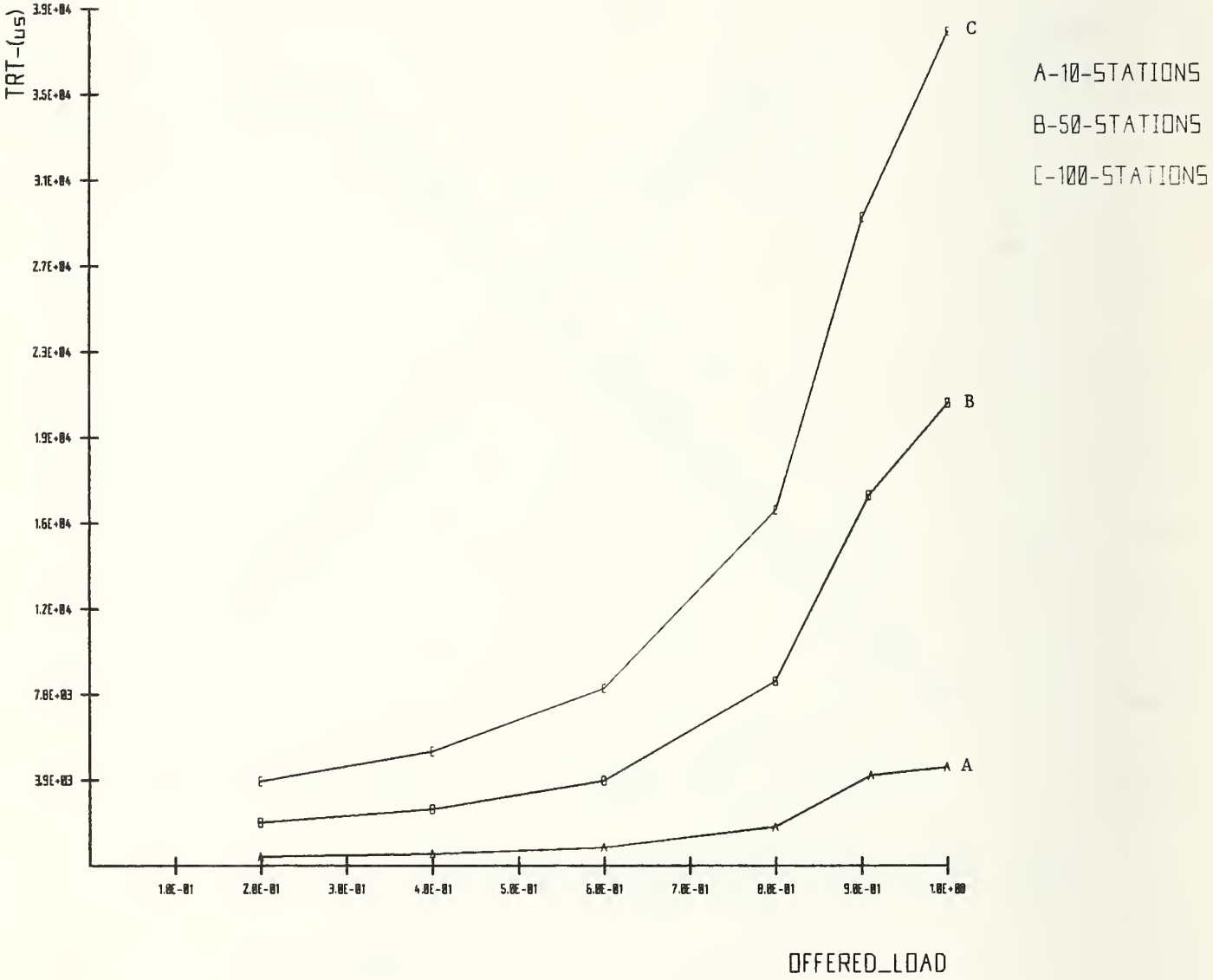
DEPENDENCE_ON_NUMBER_OF_STATIONS
frame_length=512octets;cabel_length=1500m;
THT6-1frame;address_allocation-cyclic;

EXHIBIT 6



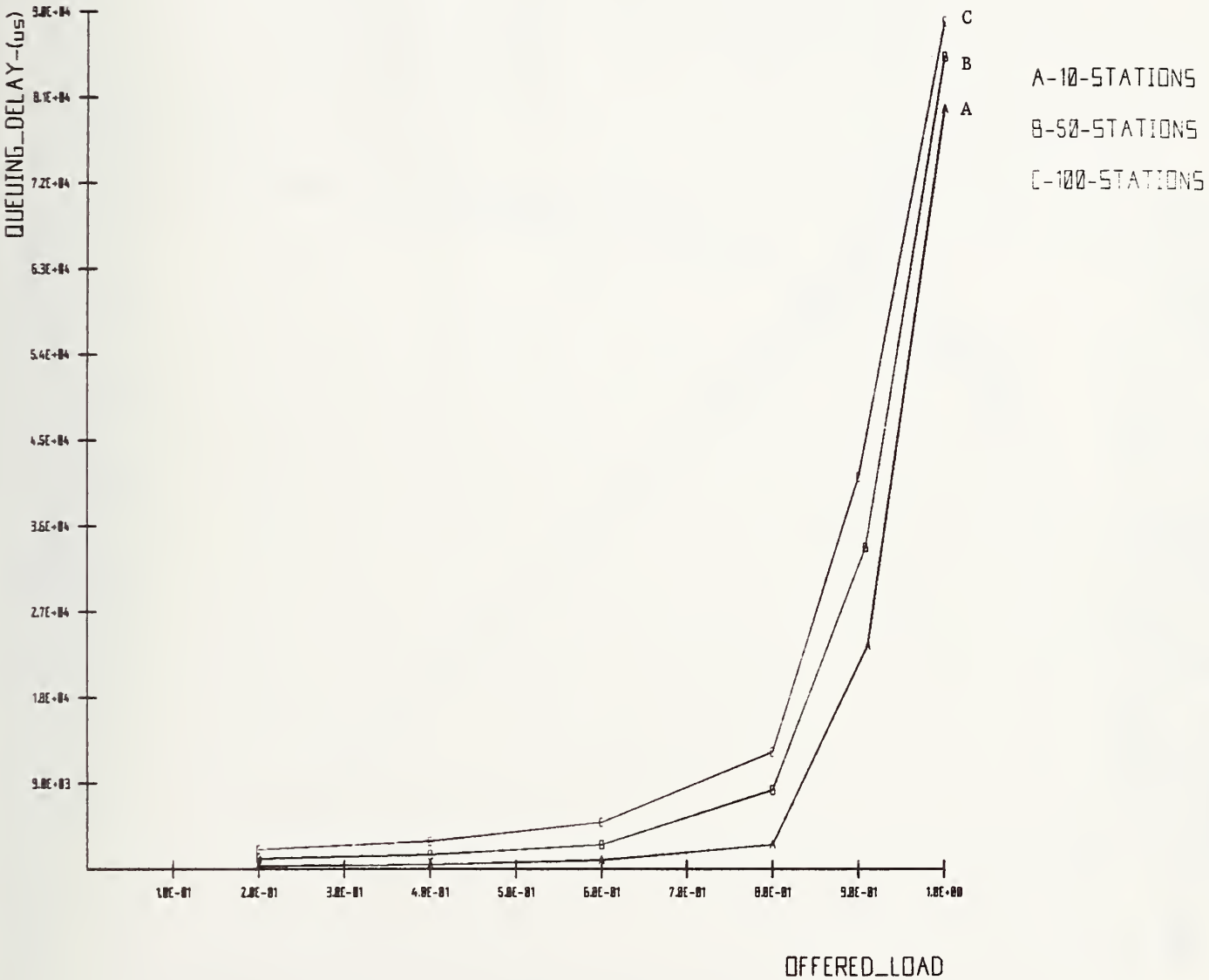
# DEPENDENCE_ON_NUMBER_OF_STATIONS
frame_length=512octets;cabel_length=1500m;
THT6-1frame;address_allocation-cyclic;

EXHIBIT 7



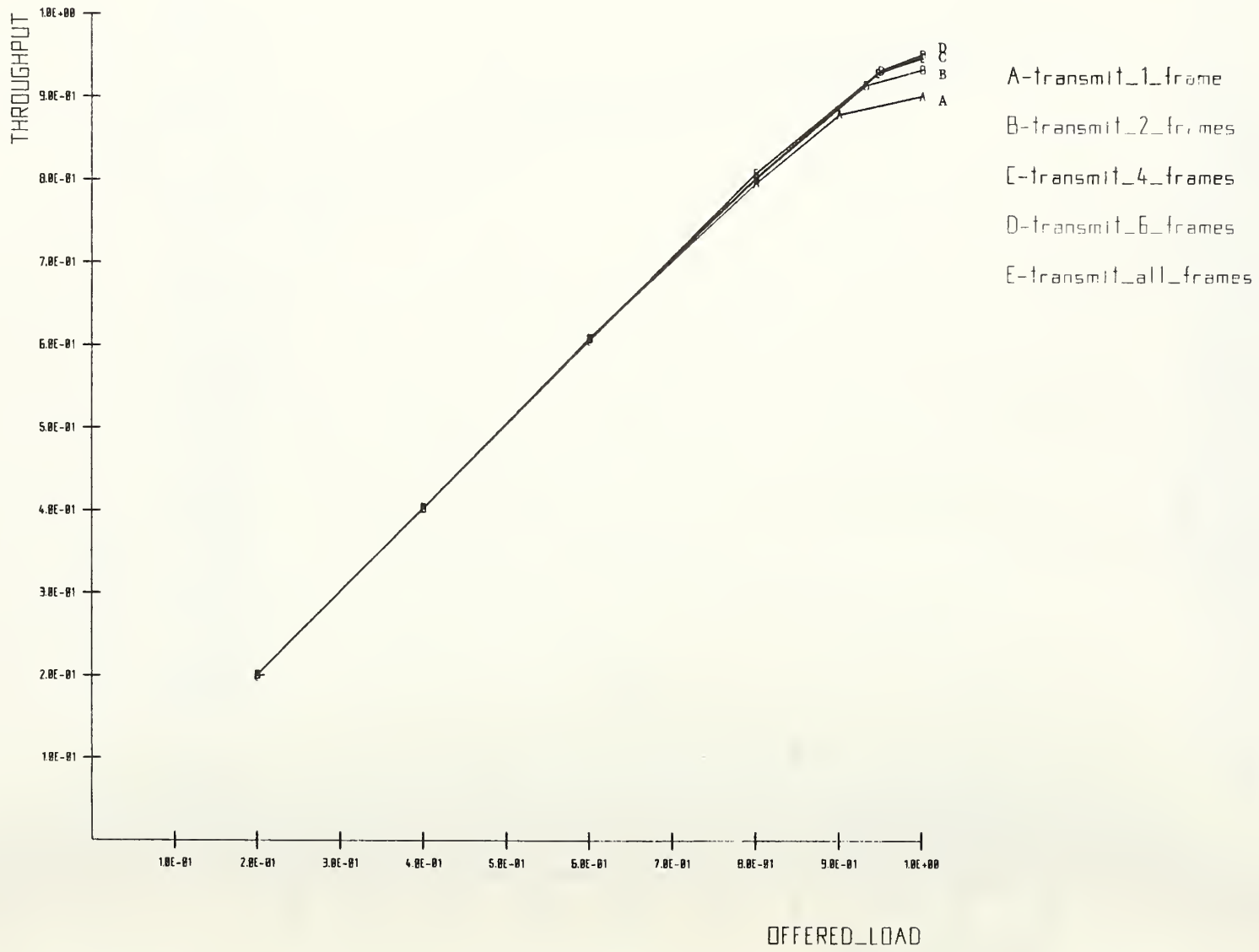
DEPENDENCE_ON_NUMBER_OF_STATIONS
frame_length=512octets;cabel_length=1500m;
THT6-1frame;address_allocation-cyclic;

EXHIBIT 8



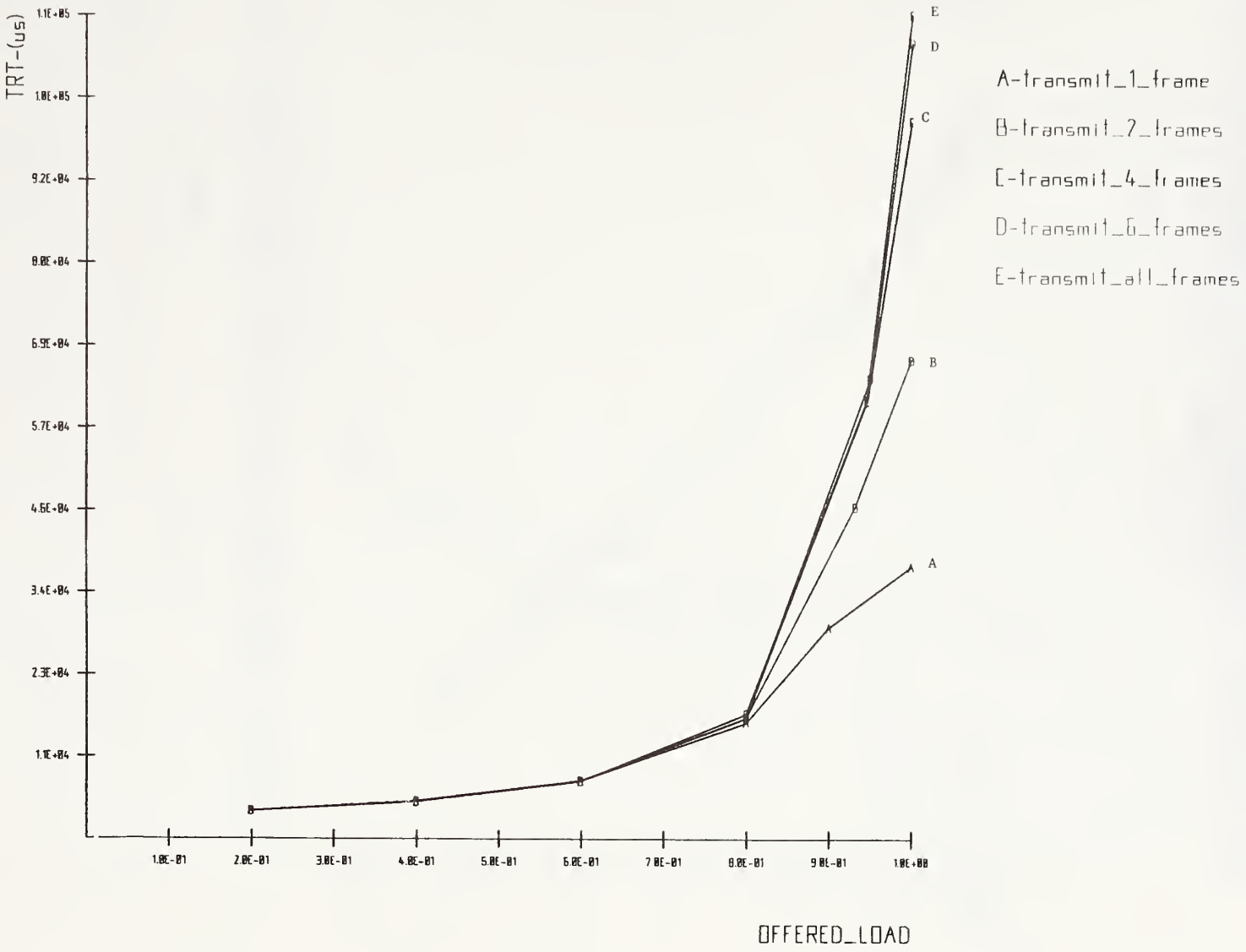
DEPENDENCE_ON_TOKEN_HOLD_TIME-6
100nodes;cabel_length=1500m;frame_length=512;
address_allocation-cyclic;

EXHIBIT 9



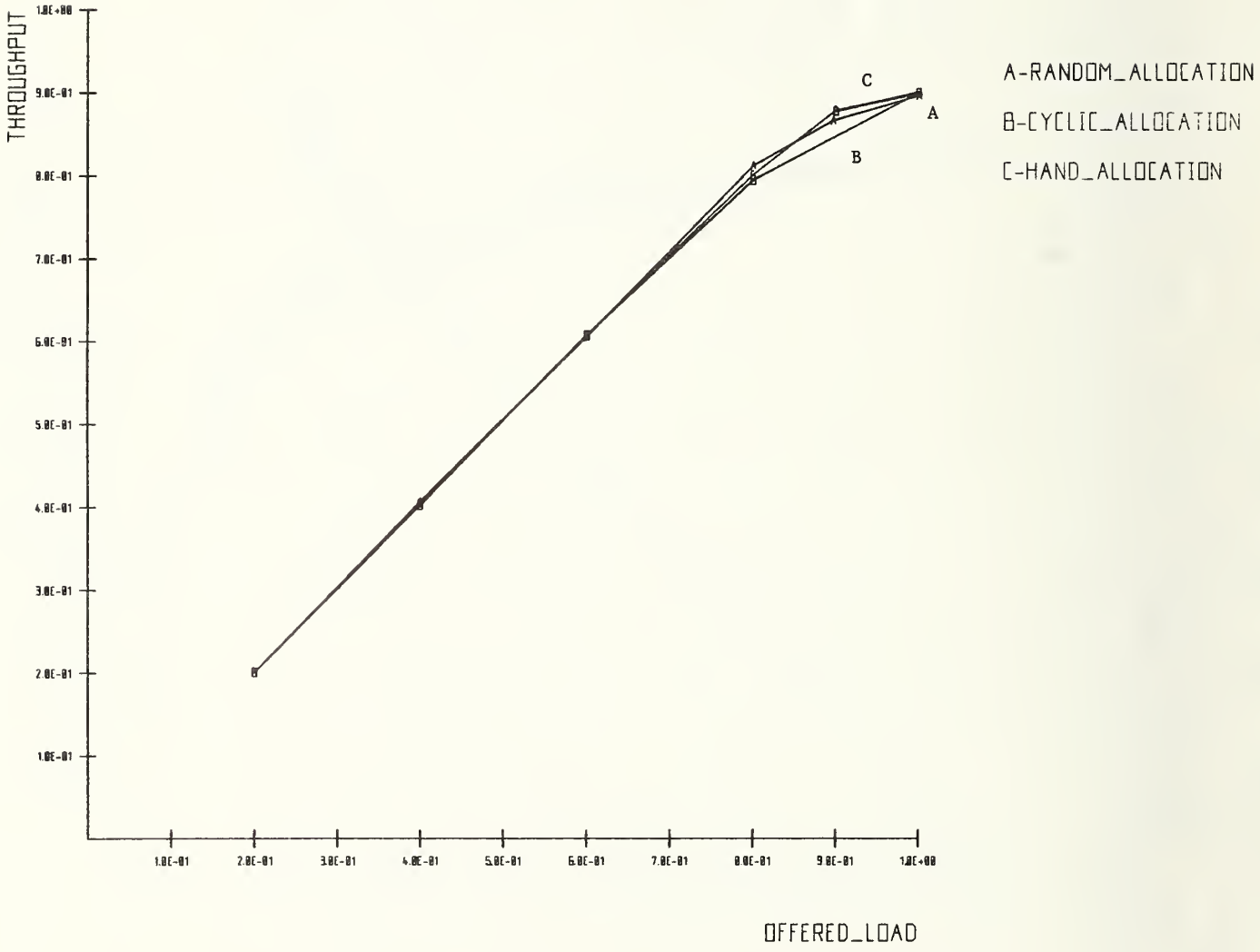
DEPENDENCE_ON_TOKEN_HOLD_TIME-6
100nodes;cabel_length=1500m;frame_length=512;
address_allocation-cyclic;

EXHIBIT 10



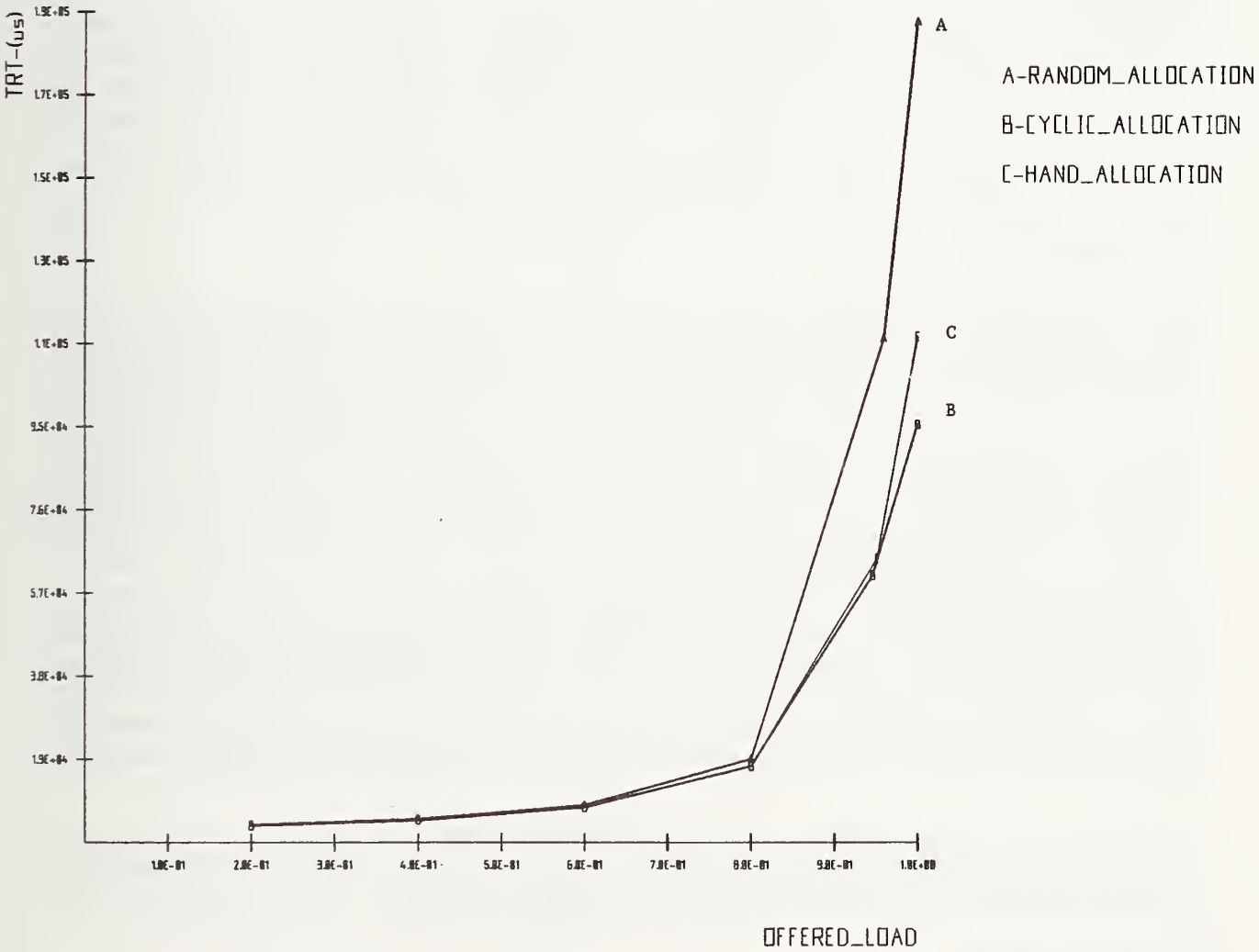
DEPENDENCE_ON_ADDRESS_ALLOCATION
100nodes;cable_length=1500m;frame_length=512octets;
THT6-1frame;

EXHIBIT 11



DEPENDENCE_ON_ADDRESS_ALLOCATION
100nodes;cable_length=1500m;frame_length=512octets;
THTG-infinite;

EXHIBIT 12



SIMULATION OF A TOKEN PASSING BUS USING A STATIC LOGICAL RING

M.E. Ulug and N.R. Shapiro

General Electric Corporate Research and Development
Schenectady, New York, 12345

ABSTRACT

An explicit token passing static local area network (LAN) was simulated using the General Purpose Simulation System (GPSS) on a VAX 780. Two types of token holding strategies were simulated. The first strategy allows each station to transmit only one information packet during a token holding time; the second allows each station to empty its buffer when it receives the token.

The results indicated that, for a given server utilization, both strategies produced the same mean token rotation time. The strategy that allows each station to empty its buffer when it gets the token results in approximately three times smaller mean waiting times at 50% bus loading. This improvement in waiting time becomes greater as the bus loading is increased.

Regardless of which of the two strategies is used, the mean waiting time and token rotation time are independent of the packet length distribution.

INTRODUCTION

A token passing bus LAN is one in which all the stations form a logical ring and the token is passed from station to station along the ring.[1,2] Such systems may involve either static or dynamic logical rings. In the dynamic ring, the set of stations in the logical ring changes on a demand basis; the set of stations that comprise the static ring, on the other hand, remains relatively stable, with only malfunctioning stations allowed to exit the logical ring. Therefore, if demand for the transmission media is great, the set of stations in the dynamic ring will grow until it is equal to the size of the static ring. When demand is low, the dynamic ring will be a smaller subset of the set of all stations that may enter the ring. (For the analysis of the dynamic ring operation see reference 3.)

Explicit token passing bus systems with many stations suffer from large overheads and long waiting times. This is because a great deal of time is wasted by passing tokens to stations that are not active. Moreover, in real-time systems the stations are allowed to transmit only one information packet per token rotation in order to keep the bus access delays bounded. The efficiency of the system improves when each station is allowed to empty its buffer when it gets the token. (For the analysis of this multipacket transmission algorithm see reference 4.)

This paper presents computer simulations of the static token passing bus to compare the performance of systems using these two types of token holding strategies. (For discussion of analytical approaches to related questions, see references 1-6.)

SYSTEM MODEL

A 50-station network was modeled. Each station received packets at a Poisson rate of $G/50$ packets/second, where G was the packet arrival rate for the system (which varied). Both a deterministic and exponential packet length distribution were used, and a 5 Mb/s network with packet lengths of 128 bytes was assumed. Thus, for the deterministic packet

length runs, packet transmission time was fixed at 205 μs, and for the exponential distribution this was the distribution mean. In Experiment 1 a packet arrival rate of 40 packets/second/station was assumed. The token passing time was varied from 25 to 175 μs. In Experiment 2 a fixed token passing delay of 175 μs was assumed, and the packet arrival rate was varied from 10 to 50 packets/second/station.

The token packet is only 20 bytes long, and at 5 Mb/s it can be transmitted in 32 μs. However, this is only the transmission time. In reality, many other delays are involved in passing a token; for example, delays are caused by processing time at stations, delays are caused by the remodulator, and many delays result from the execution of the media access protocol. Most of these occur in a random manner, as in the transmission of a "who follows" packet or in a complete logical ring reconfiguration.

In addition, delays are encountered in the execution of the link, network, and transport level protocols. These result in the reduction of the available channel bandwidth. Some of these delays are caused by CRC errors, redundant transmissions, acknowledgments, buffer overflows, window closings, etc. In this experiment, certain worst case delays or reductions in the channel bandwidth were assumed and were included in the token passing time. Clearly, every LAN design and every implementation is different; the assumptions in these experiments should not be regarded as representative of any one system.

Two different simulation models were used to assess two different token holding time algorithms. The first model (reported below in experiments 1A and 2A) allows each station one packet transmission per token possession. All other arrivals are queued for later transmission on a first come, first served basis.

A second algorithm modeled via the simulation (reported below in experiments 1B and 2B) was one in which stations could, upon receipt of a token, transmit all queued packets. However, arrivals during the token holding period were not transmitted, but were queued for the next token holding period.

In each simulation examined a number of relevant network performance measures, including

- Mean queueing delay: The average time spent in a station's queue awaiting transmission.
- Mean rotation time: The average token interarrival time across stations.
- Mean and maximum queue contents: Averaged across stations, the maximum queue size and the average queue contents per rotation (calculated as the number of queue entries divided by the number of rotations).
- Mean number of transmissions per rotation: Calculated as total transmissions divided by number of rotations.
- Mean number of stations with >0 arrivals: Average number of stations with one or more frames arriving for transmission per rotation.

Performance of the two types of simulated systems was assessed in terms of the above variables as a function of token passing time and packet arrival rate. These data are reported as Experiments 1 and 2, respectively, with each experiment divided into two parts, A and B, to separate results from the two system types.

The minimum sampling period required to produce stable results was determined in preliminary experiments. The standard 50-station configuration was used in this exploratory work, with each station receiving packets at a rate of 40 per second. Sampling periods from 100 to 500 rotations were examined; no consistent upward or downward trends were evident in the data in sampling periods from 300 to 500 rotations. Thus, the 500-rotation sampling period was used in all experiments.

EXPERIMENT 1A

This experiment examined the effect of token passing delay. Token-passing times of 25, 75, 125 and 175 μs were used. This time included all station-to-station delays; that is, the total delay from the end of the last data frame to the receipt of the last bit of the token packet by the new token holder.

Packet arrivals to individual stations were exponentially distributed with a mean of 40 packets per second. Each station was allowed one packet transmission per token holding period.

Results

Figures 1-8 show the effect of token passing time on the various measures of LAN performance for fixed and exponential frame lengths. As expected, increased token passing delays result in corresponding increases in other system delays. Figure 1 shows this increase for mean delay in the queue in milliseconds. Note that there is no difference in the times for the fixed or exponential packet size distributions. Figure 2 shows the increase in the standard deviation of the delay.

Figure 3 gives token rotation times in milliseconds for the four token passing times, and gives the fixed and exponential frame length distributions. This measure shows a linear increase as a function of token passing delay. Note that there is no significant delay in the times for the fixed and exponential packet size distributions. Figure 4 shows the standard deviation of the token rotation time. Note that the standard deviation of the packets with exponentially distributed sizes is considerably higher than those of fixed length. Also note that the slope of these standard deviation curves is much smaller.

Figures 5 and 6 also show a performance decrement as the token passing delay is increased. Mean queue length and maximum queue length both increase almost linearly as a function of token passing delay.

The mean number of transmissions and the mean number of stations with packet arrivals also increase as a function of token passing time (see Figures 7 and 8). This is probably a consequence of the longer token rotation times for the increased token passing times—the longer the token rotation, the more packet arrivals per rotation, and the more stations with frames to transmit. This is also supported by the ratio of the two measures presented in these figures. The ratio of the mean number of transmissions per rotation to the mean number of

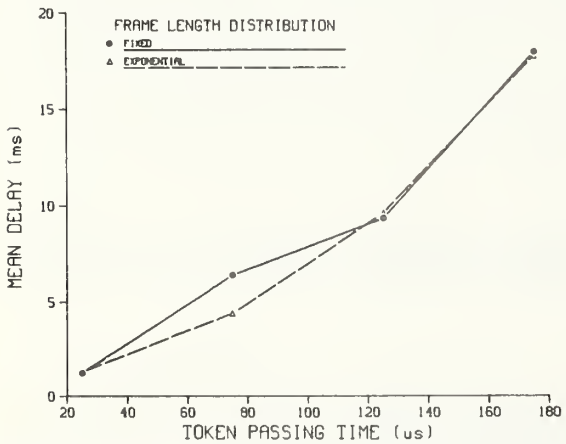


Figure 1. Single packet transmission. 5 Mb/s, 40 packets/second/station, 50 stations, 205 μs packet service time.

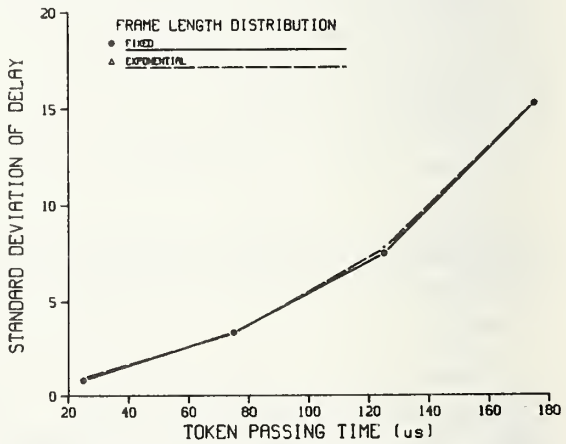


Figure 2. Single packet transmission. 5 Mb/s, 40 packets/second/station, 50 stations, 205 μs packet service time.

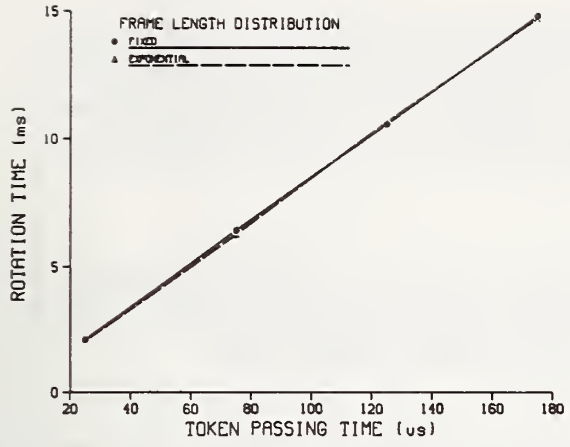


Figure 3. Single packet transmission. 5 Mb/s, 40 packets/second/station, 50 stations, 205 μs packet service time.

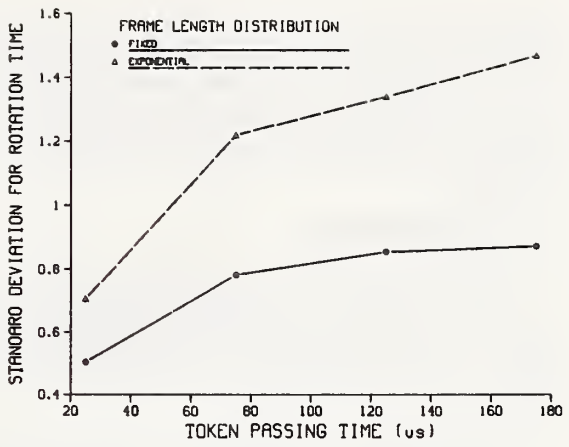


Figure 4. Single packet transmission. 5 Mb/s, 40 packets/second/station, 50 stations, 205 μs packet service time.

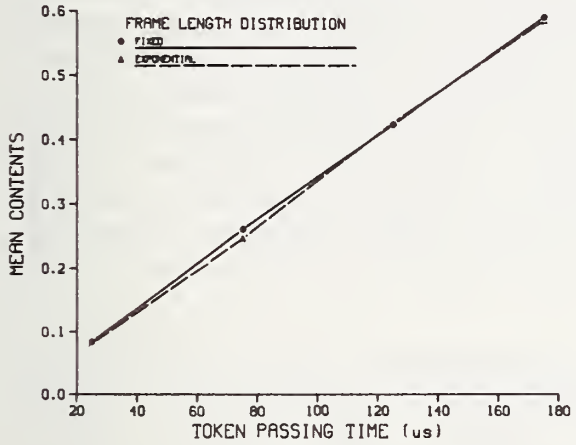


Figure 5. Single packet transmission. 5 Mb/s, 40 packets/second/station, 50 stations, 205 μs packet service time.

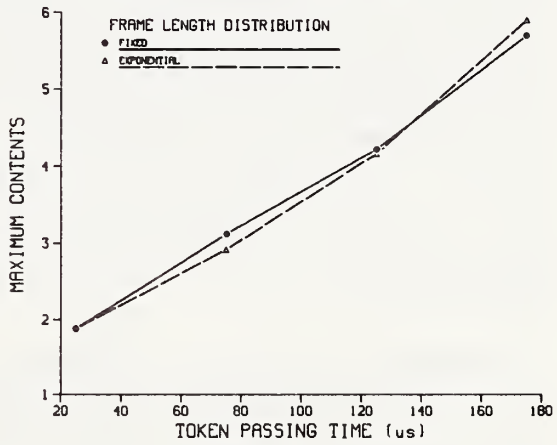


Figure 6. Single packet transmission. 5 Mb/s, 40 packets/second/station, 50 stations, 205 μs packet service time.

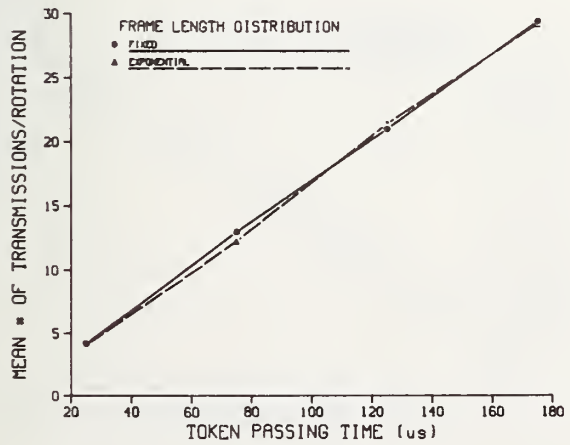


Figure 7. Single packet transmission. 5 Mb/s, 40 packets/second/station, 50 stations, 205 μs packet service time.

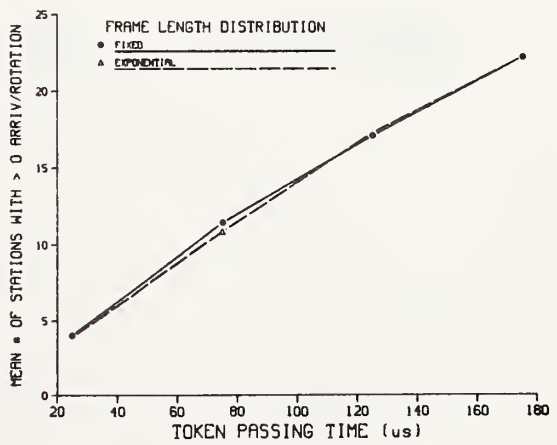


Figure 8. Single packet transmission. 5 Mb/s, 40 packets/second/station, 50 stations, 205 μs packet service time.

stations with more than one arrival per rotation yields an index that reflects the average number of stations that had packets to be transmitted that were left over from a previous rotation. This contributes, of course, to bus loading, because a station with a packet left over from a previous rotation must transmit, whether or not it has received a new frame during the rotation time.

EXPERIMENT 1B

This experiment was identical to Experiment 1A, except that the stations were allowed to transmit the same number of frames that were in the buffer during token receipt. Once again, we were interested in assessing the effects of various token passing delays.

Results

Like Figures 1-8, Figures 9-16 show the effect of token passing time on the various measures of LAN performance for fixed and exponential frame lengths. Units of measure in the figures are as previously described. When compared with each of the figures in the previous experiment, Figures 9-16 illustrate the effect of the new system protocol allowing each station to transmit any packets waiting in the station queue when the token is received. Comparing the figures, the multipacket protocol produces considerably shorter delay and delay variance at the longer token passing times, and comparable delays and delay variance at the shortest token passing time. On all other measures, however, data from the multipacket protocol simulations (Figures 11-16) are comparable to the data from the single packet protocol case (Figures 3-8). Note, however, that the standard deviations of the token rotation time in Figure 12 are generally larger than those of the comparable single packet transmission protocol, shown in Figure 4. Finally, the mean delay and the token rotation time of the single- and multiple-packet transmission algorithms are plotted and compared with each other in Figures 33 and 34.

The ratio of the measures in Figures 15 and 16 can be examined as in the previous experiment; however, in this case the interpretation is slightly different. Recall that these data are for the multipacket algorithm, which implies that a station will transmit all of the frames it has queued when it receives the token. For these data the average number of frames transmitted per rotation reflects the total contribution of one or more frames transmitted per rotation per station. Thus the ratio reflects the average number of frames transmitted per station per rotation.

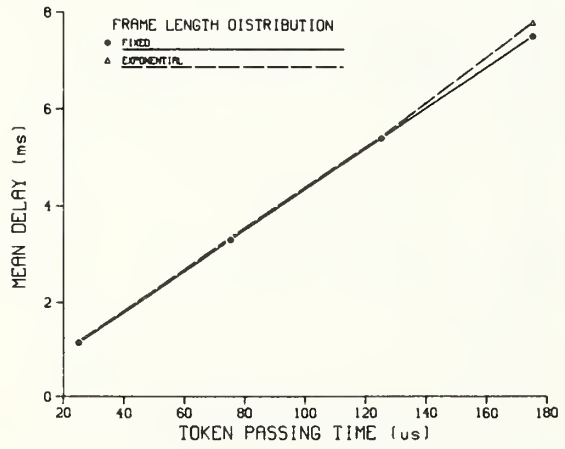


Figure 9. Multiple packet transmission. 5 Mb/s, 40 packets/second/station, 50 stations, 205 μs packet service time.

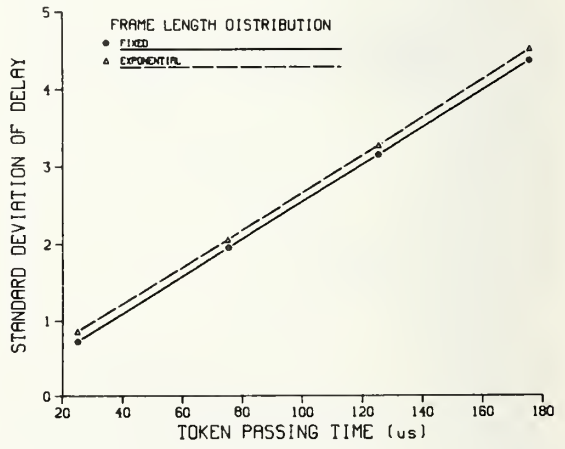


Figure 10. Multiple packet transmission. 5 Mb/s, 40 packets/second/station, 50 stations, 205 μs packet service time.

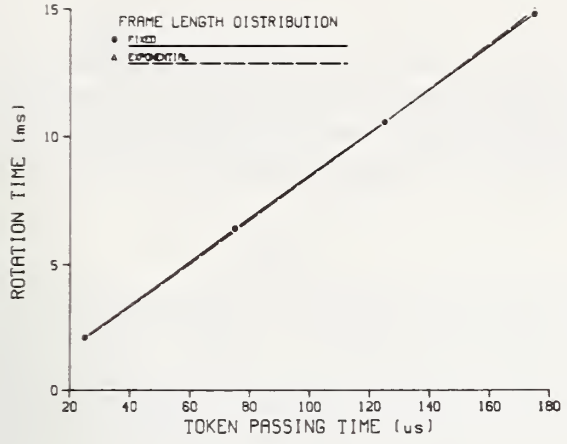


Figure 11. Multiple packet transmission. 5 Mb/s, 40 packets/second/station, 50 stations, 205 μs packet service time.

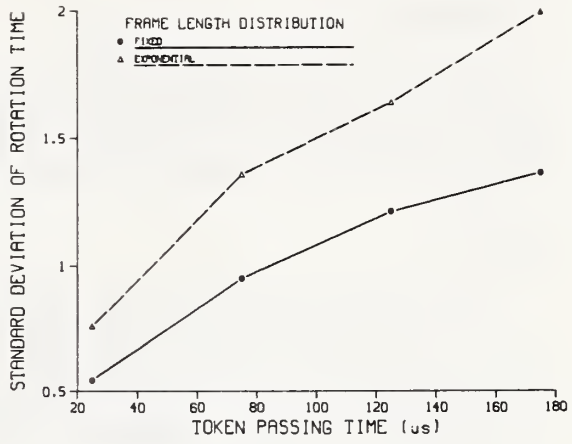


Figure 12. Multiple packet transmission. 5 Mb/s, 40 packets/second/station, 50 stations, 205 μs packet service time.

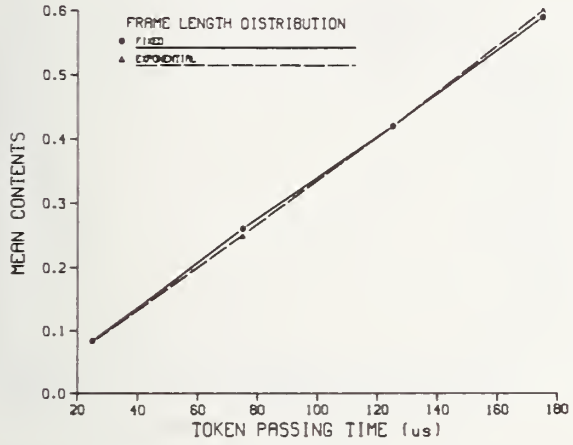


Figure 13. Multiple packet transmission. 5 Mb/s, 40 packets/second/station, 50 stations, 205 μs packet service time.

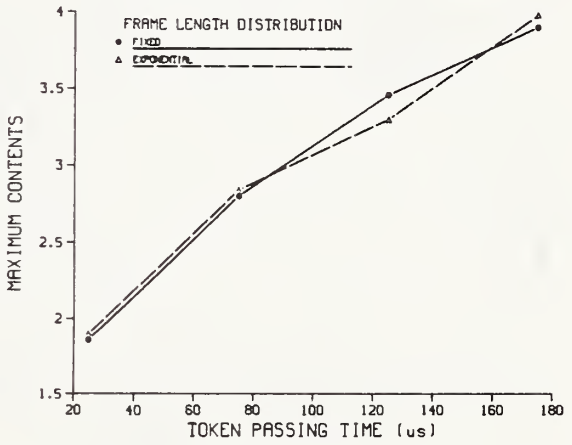


Figure 14. Multiple packet transmission. 5 Mb/s, 40 packets/second/station, 50 stations, 205 μs packet service time.

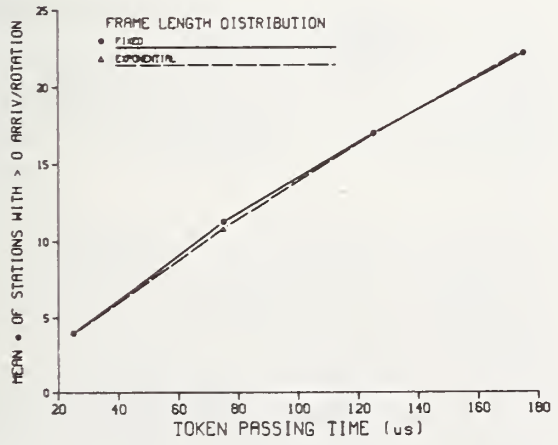


Figure 15. Multiple packet transmission. 5 Mb/s, 40 packets/second/station, 50 stations, 205 μs packet service time.

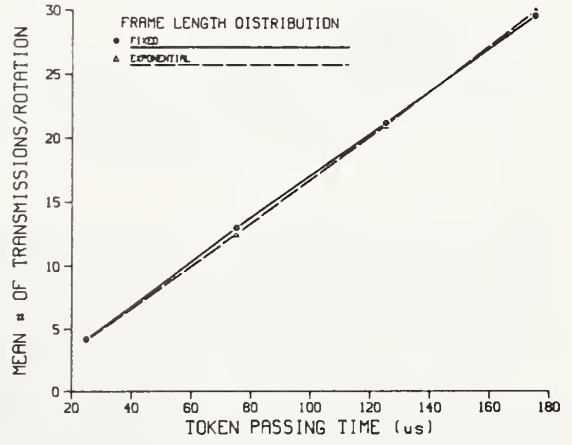


Figure 16. Multiple packet transmission. 5 Mb/s, 40 packets/second/station, 50 stations, 205 μs packet service time.

EXPERIMENT 2A

The second experiment examined the relationship between packet arrival rate and the standard set of network performance measures. Packet arrivals were the same for all stations. Each station received 10, 20, 30, 40, or 50 packets/second, which corresponds to a system arrival rate of 500 to 2500 packets/second. In this experiment the token passing time was taken as 175 μs.

Results

Simulation results for the single packet protocol are shown in Figures 17-24. As packet arrival rate increased, mean waiting time in station queues (Figure 17), mean rotation time (Figure 19), mean and maximum queue length (Figures 21 and 22), and mean number of transmissions per rotation (Figure 23) all showed corresponding increases. As in the previous experiments, there was no difference between the fixed and exponential packet size distributions. However, variance appears to be somewhat higher at the higher arrival rates for the exponential packet length simulations, as evidenced by the higher standard deviations seen in Figures 18 and 20.

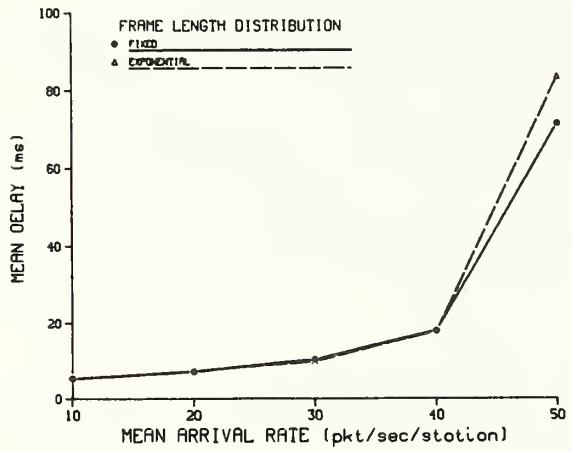


Figure 17. Single packet transmission. 5 Mb/s, 175 μs token passing time, 50 stations, 205 μs packet service time.

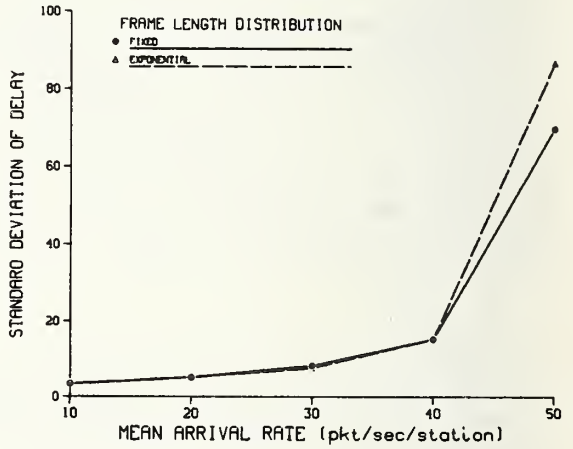


Figure 18. Single packet transmission. 5 Mb/s, 175 μs token passing time, 50 stations, 205 μs packet service time.

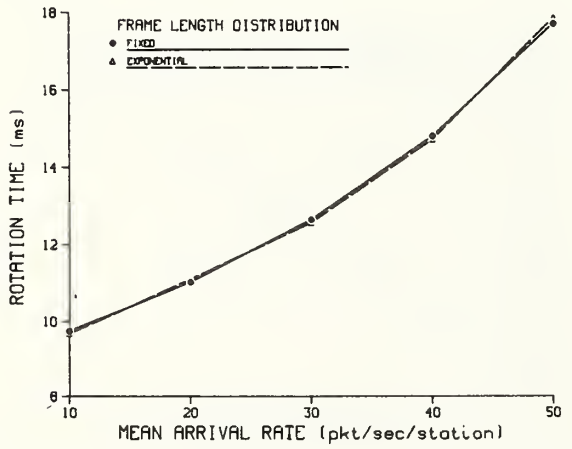


Figure 19. Single packet transmission. 5 Mb/s, 175 μs token passing time, 50 stations, 205 μs packet service time.

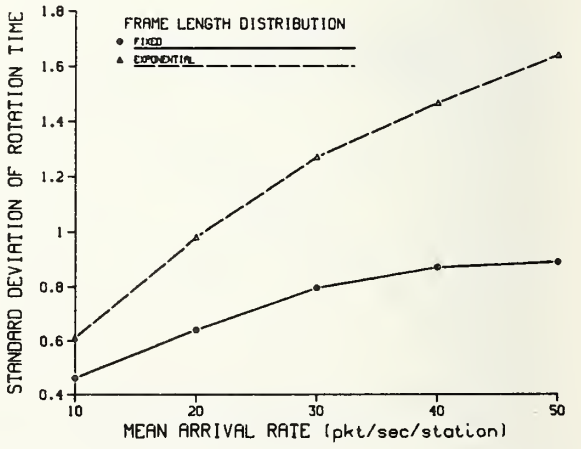


Figure 20. Single packet transmission. 5 Mb/s, 175 μs token passing time, 50 stations, 205 μs packet service time.

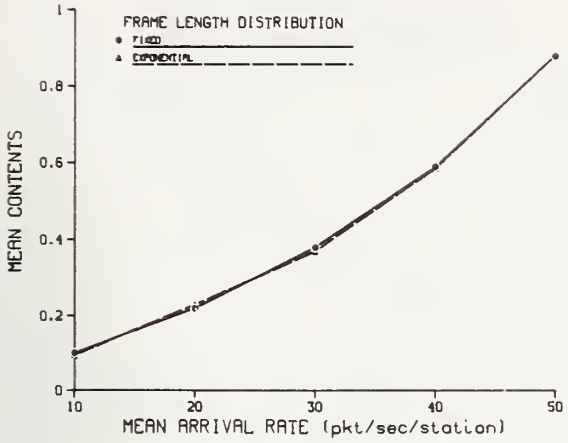


Figure 21. Single packet transmission. 5 Mb/s, 175 μs token passing time, 50 stations, 205 μs packet service time.

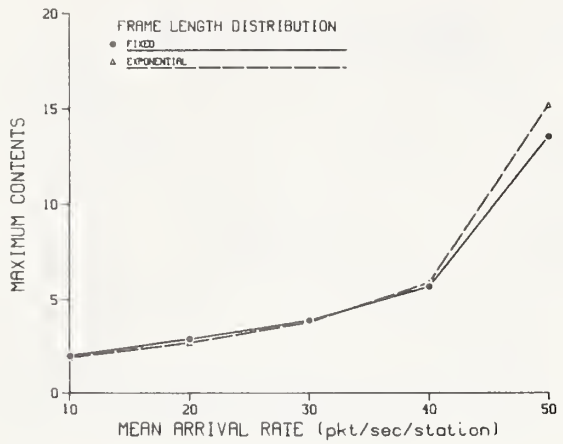


Figure 22. Single packet transmission. 5 Mb/s, 175 μs token passing time, 50 stations, 205 μs packet service time.

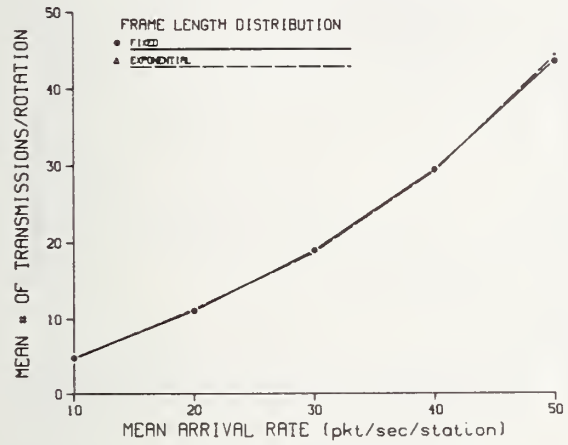


Figure 23. Single packet transmission. 5 Mb/s, 175 μs token passing time, 50 stations, 205 μs packet service time.

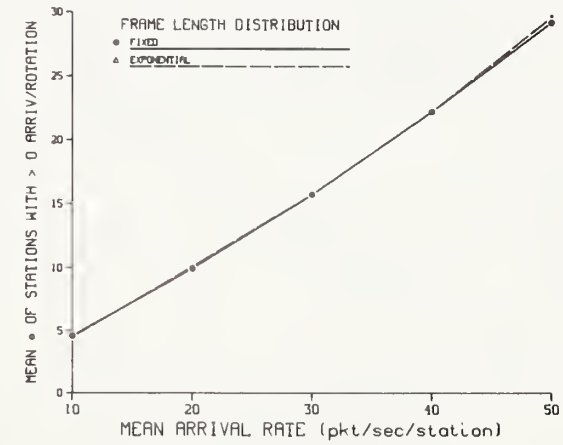


Figure 24. Single packet transmission. 5 Mb/s, 175 μs token passing time, 50 stations, 205 μs packet service time.

EXPERIMENT 2B

This experiment was identical to Experiment 2A, except that the stations were allowed to transmit the same number of frames that were in the buffer during token receipt. Once again, we were interested in assessing the effects of various packet arrival rates.

Results

The results are presented in Figures 25-32. The same general trends were evident: As packet arrival rate increased, mean queueing delay, token rotation times, queue lengths, and transmissions and arrivals per rotation all increased. The increase in queueing delay, however, was much more subdued. In fact, the multipacket algorithm resulted in an almost eightfold decrease in queueing delays (compare Figures 25 and 17). The two algorithms resulted in comparable rotation times, but slightly higher variance in rotation time for the multipacket algorithm (see Figures 20 and 28). Finally, the mean delay and the token rotation time of the single- and multiple-packet transmission algorithms are plotted and compared with each other in Figures 35 and 36.

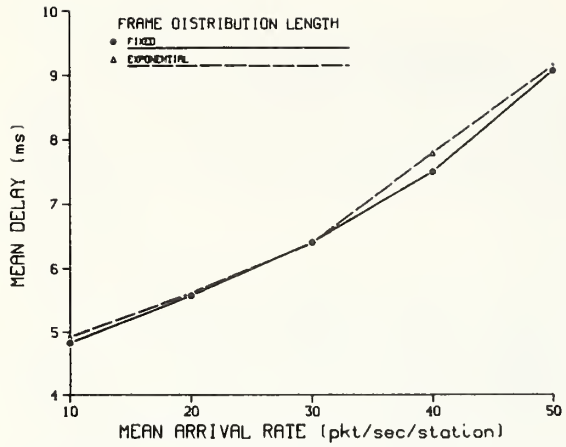


Figure 25. Multiple packet transmission. 5 Mb/s, 175 μs token passing time, 50 stations, 205 μs packet service time.

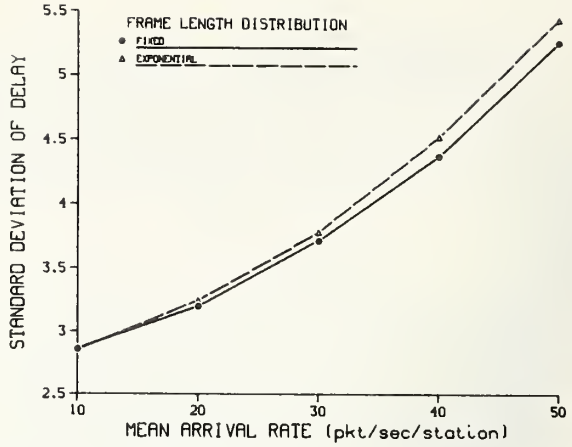


Figure 26. Multiple packet transmission. 5 Mb/s, 175 μs token passing time, 50 stations, 205 μs packet service time.

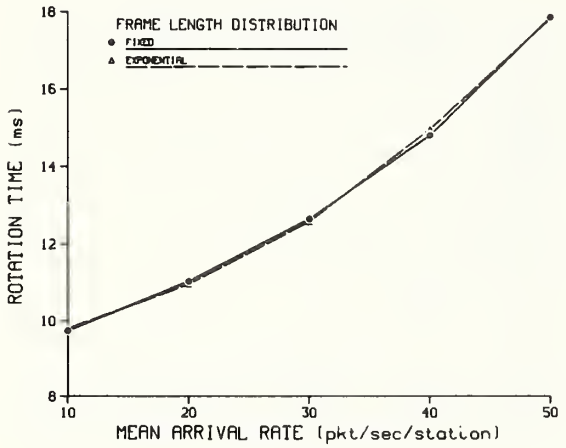


Figure 27. Multiple packet transmission. 5 Mb/s, 175 μs token passing time, 50 stations, 205 μs packet service time.

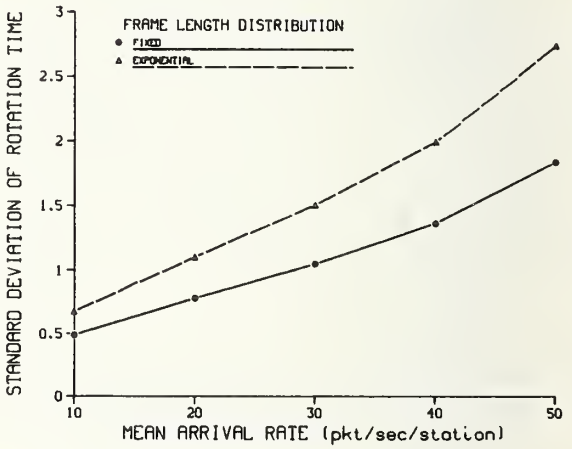


Figure 28. Multiple packet transmission. 5 Mb/s, 175 μs token passing time, 50 stations, 205 μs packet service time.

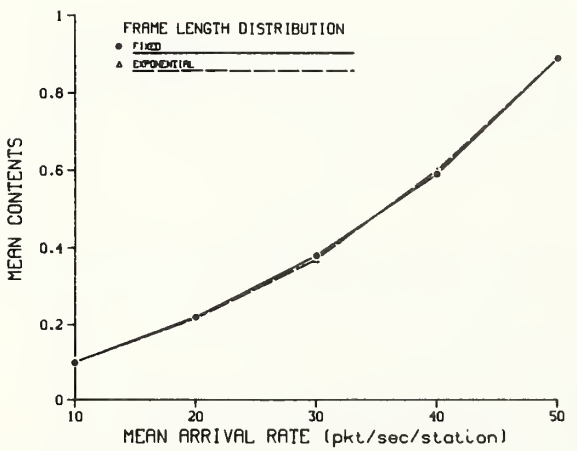


Figure 29. Multiple packet transmission. 5 Mb/s, 175 μs token passing time, 50 stations, 205 μs packet service time.

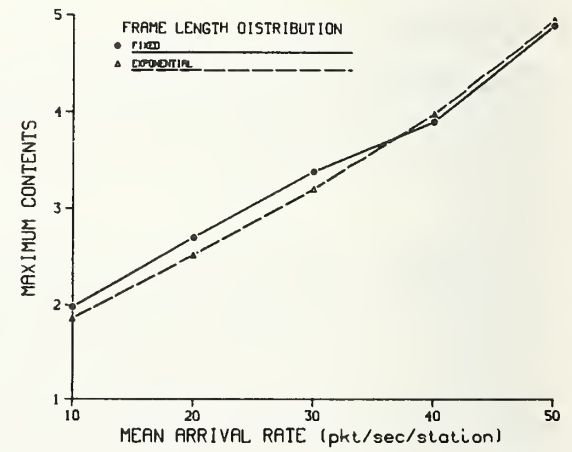


Figure 30. Multiple packet transmission. 5 Mb/s, 175 μs token passing time, 50 stations, 205 μs packet service time.

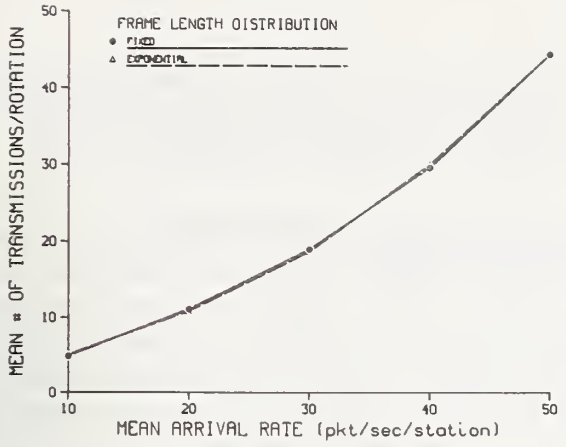


Figure 31. Multiple packet transmission. 5 Mb/s, 175 μs token passing time, 50 stations, 205 μs packet service time.

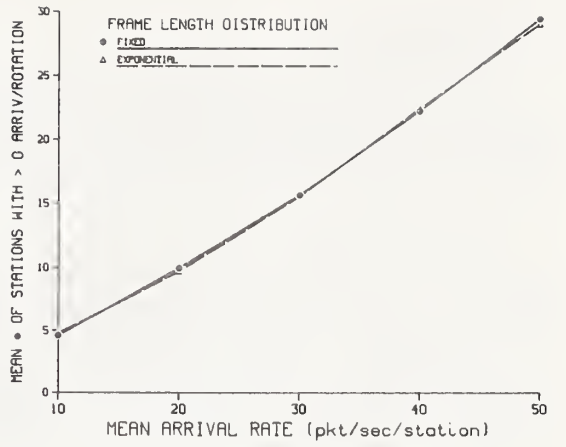


Figure 32. Multiple packet transmission. 5 Mb/s, 175 μs token passing time, 50 stations, 205 μs packet service time.

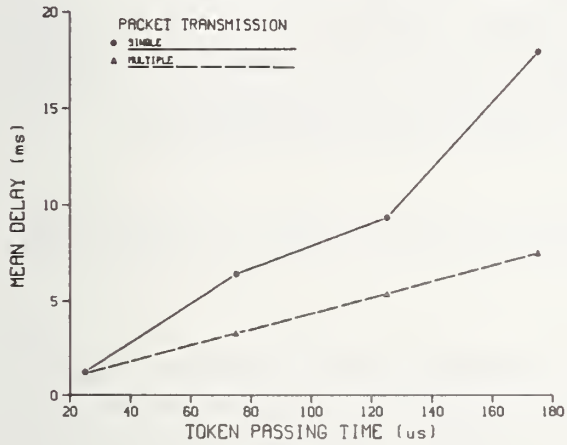


Figure 33. Single vs multiple packet transmission. 5 Mb/s, 40 packets per second per station, 50 stations, 205 μs packet service time.

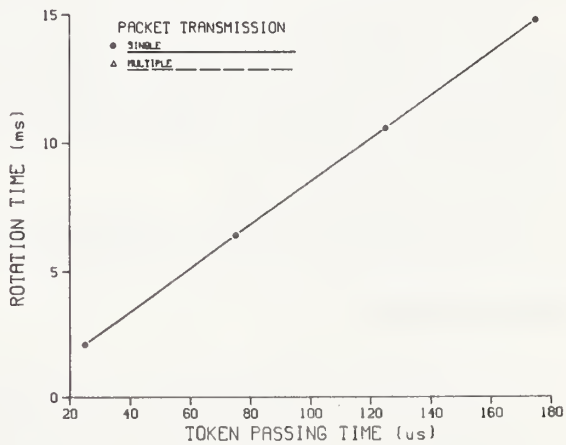


Figure 34. Single vs multiple packet transmission. 5 Mb/s, 40 packets per second per station, 50 stations, 205 μs packet service time.

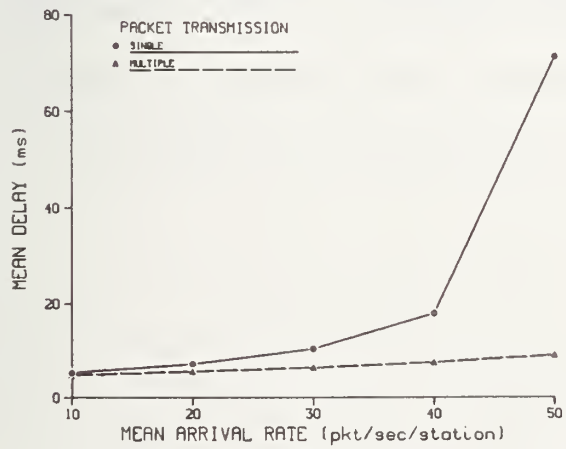


Figure 35. Single vs multiple packet transmission. 5 Mb/s, 175 μs token passing time, 50 stations, 205 μs packet service time.

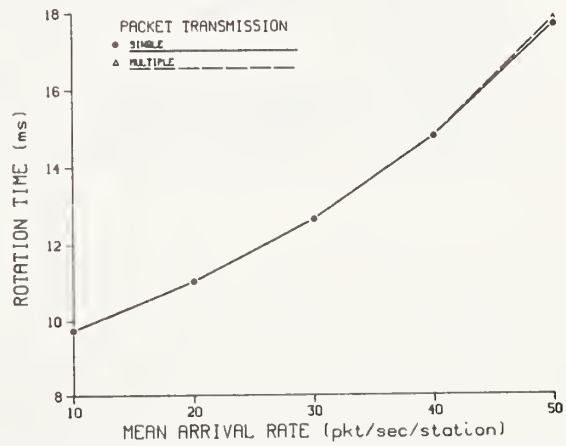


Figure 36. Single vs multiple packet transmission. 5 Mb/s, 175 μs token passing time, 50 stations, 205 μs packet service time.

CONCLUSIONS

The computer simulation of an explicit token passing static LAN with two types of token holding time strategies has been conducted. The first strategy allows each station to transmit only one information packet during a token holding time; the second strategy allows each station to empty its buffer when it gets the token.

Results indicate that for a given server utilization both strategies produce exactly the same mean token rotation time. The strategy that allows each station to empty its buffer when it gets the token results in approximately three times smaller waiting times. This improvement becomes greater as the bus loading is increased. However, in this case the bus access delays are bounded at a higher level. The multipacket transmission algorithm, used with a limit based on the token rotation time as observed by the individual stations, retains most of the benefits resulting from this approach while maintaining a bound on the waiting time.

Computer simulation results indicate that the standard deviation of the token rotation time for the strategy that allows each station to empty its buffer is larger than that of the strategy that allows the transmission of one information packet per token rotation time.

The simulation results also show that the delay, delay variance, and the token rotation time are independent of the distribution of the packet length for both strategies. However, the standard deviation of token rotation time for the system operating with exponential distributed packet lengths is larger than that of those systems using fixed length packets.

ACKNOWLEDGMENTS

The authors would like to express their thanks to Mr. C.R. Stein, Miss M.R. Laliberte and Dr. T. Saydam for their help and advice.

REFERENCES

1. Ulug, M.E., White, G.M., and Adams, W.J., "Bidirectional Token Flow System," *Seventh Data Communications Symposium, Vol. 11, No. 4,* Mexico City, October 1981.
2. Ulug, M.E., "Calculation of Waiting Times for a Real Time Token Passing Bus," *Proceedings of Computer Networking Symposium,* Dec. 1983, Silver Spring, Maryland.
3. Ulug, M.E., "Calculation of Waiting Times for a Dynamic Token Passing Bus," *Proceedings of the Computer Networking Symposium,* Dec. 1984, Gaithersburg, Maryland.
4. Ulug, M.E., "Comparison of Token Holding Time Strategies for a Static Token Passing Ring," *Proceedings of the Computer Networking Symposium,* Dec. 1984, Gaithersburg, Maryland.
5. Konheim, A.G., and Meister, B. "Waiting Lines and Times in a System with Polling," *JACM 21,* 470-490, July 1974.
6. Kleinrock, L. *Queueing Systems, Vol. 1. Theory,* Wiley-Interscience, New York, 1975.

Session IV:

Network Management Issues

Chairperson: Gary Workman
General Motors

A HIERARCHICAL POLICY FOR TIMER ASSIGNMENTS IN IEEE 802.4 NETWORK

K.H. Muralidhar

Communications and Network Laboratory
Industrial Technology Institute
P.O. Box 1485
Ann Arbor MI - 48106.

ABSTRACT

Process oriented and critically timed communications requirements necessitates a real-time, failure-proof network for factory automation. The ability to control the accessing at the data link level by assigning priorities and timers make token passing more advantageous in the factory environment. The performance of token passing schemes depends greatly on the value of various timers that can be controlled at the data link level. A hierarchical policy to assign values for various timers in token passing access method in an optimization framework is reported. The basic idea in this scheme is to decompose the decision making capability into two hierarchically arranged levels. In the higher level, a centralized linear programming problem is solved to maximize the overall bus utilization of the network. In the lower level, a distributed integer programming problem is solved at each station to maximize the buffer utilizations. The higher level problem is solved at a slower time scale compared to lower level problem.

1. INTRODUCTION:

Factory of the future imposes integration of current automated machines in a factory floor. This integration is achieved via data communications networks. Process oriented and critically timed communications requirements necessitates a real-time, failure proof network for factory automation [McGA85]. Several data communications networks with different access methods exist for local area network environment. Some of them are based on the contention schemes and some are based on token passing.

The deterministic nature of token passing access method, which guarantee the time critical aspects of communication and high reliability at peak loads favor the token passing access method for networks in factory environments. Further, ability to control accessing at the data link level by assigning priorities and timers make token passing access method even more advantageous in the factory environment.

The IEEE 802.4 committee [IEEE84] has defined a token passing scheme which is suited for factory environments. The responsibility of the IEEE 802 committee is to specify lower two layers of the ISO's OSI model to provide local area network interconnection capability. However, the problem of inter-communication capability for factory environments is addressed by Manufacturing Automation Protocol [MAP85] specifications.

The performance of token passing schemes depends greatly on timer values that can be assigned in a dynamic or quasi-static fashion at data link level. These timers are: high-priority token holding timer and target rotation timers for other priority classes. Various simulation results are reported in the literature [RAHI83, ARCH84] to study the effects of these timers on the performance of token passing access methods.

Several performance measures are defined for local area networks [STAL84]. The significant ones are network utilization and throughput. Network utilization is defined as the ratio of throughput to capacity or bandwidth of the system. Throughput is defined by the ratio of message transmission time to the sum of message transmission time and overhead time. The overhead time in the case of token passing access method comprise of, station delay, ring maintenance time, token passing time etc., which affect the effective throughput of the network.

In this paper, a hierarchical policy to assign timer values in token passing access method in an optimization framework is reported. The basic idea in this scheme is to decompose the decision making capability into two hierarchically arranged levels. In the higher level, a centralized linear programming problem is solved to maximize the overall throughput of the network. In the lower level, a distributed integer programming problem is solved at each station to maximize buffer utilization. In fact,

a buffer allocation problem is solved to achieve the intended buffer utilization. The higher level problem is solved at a slower time scale compared to lower level problem.

The main features of this hierarchical policy are:

- consideration of multiple objectives such as, throughput and buffer utilization independently in the optimization process;
- consideration of various attributes such as, load demand rates, priorities of messages, priorities of stations etc., in developing cost functions in the optimization problem;
- complexity reduction due to distributed lower level problem;
- cost effective due to exploitation of differences in time scales between higher and lower level problems.

Additional details on this framework and applicability of this framework to other problems of interest in data communications networks are reported in [MURA84].

Organization of this paper is as follows. Certain notations and definitions used throughout this paper is given in section 2. In section 3, higher level problem is formulated while lower level problem is formulated in section 4. Step-by-step algorithm to assign timer values is given in section 5. In section 6, simulation results are given. Finally, conclusions and some possible extensions to this method are given in section 7.

2. NOTATIONS AND DEFINITIONS:

Let $N = \{ 1, 2, 3, \ldots, N \}$ be the set of stations in the logical ring formed in the network. Let T_i^h be the maximum token holding time (time during which a station is allowed to transmit messages) for station $i \in N$. Let $P = \{ 0, 2, 4, 6 \}$ be the set of possible priorities or access classes in the token bus. Let $\rho \subset P$, such that $\rho = \{ 0, 2, 4 \}$. Let T_i^m be the target rotation timer for station $i \in N$ with a priority $m \in \rho$. Let T_i^6 be the high priority token holding time for $i \in N$.

Let C_i be the cost coefficient for station $i \in N$. C_i depends on factors such as,

priority of station, demand rates at station etc. Let L_s denote the slot time in the token bus network. Let T^R be the maximum token rotation time tolerable in the network. Let K be a constant which indicates minimum bus utilization required for the token bus network. Let D_i denote the total load demand rate expressed in packets/sec and P_i denote the priority cost for station $i \in N$.

For a given access class $j \in P$, let d_i^j denote the average load demand rate in packets/sec at station $i \in N$. Let b_i^j and y_i^j denote the buffer in number of packets allocated and cost coefficient defined in lower level problem for a priority $j \in P$ at station $i \in N$ respectively. Let G_i denote a constant to transform the buffer allocated to time units. G_i depends on various parameters such as, data size, data rate, maximum distance between nodes, latency of the headend, and setup time required for packet transmission at station $i \in N$. Let B_i denote total buffer available at station $i \in N$. Let $g:R \rightarrow R$ denote the functional mapping between demand rate d_i^j and buffer b_i^j. With these notations and definitions, the two level problem to assign timers in the token bus network can now be formulated.

3. HIGHER LEVEL PROBLEM:

As described earlier, timer assignment problem is decomposed into two levels; a higher level problem and a lower level problem. Higher level problem deals with selection of suitable token holding timers for active stations in the network. Before actually formulating higher level problem, several considerations that influence the formulation are described in the following.

In a typical factory environment, several different types of stations characterized by machines attached to each node are present. In particular, the network may consist of several programmable logic controllers, robots, numerical control machines, and host computers. Due to the diversity in behavior and requirements of these machines, the timer values that need to be set in these network stations become different. Some of the parameters that influence the selection of timer values are:

- priority of a station in the network. For instance, a host attached to a

network managing the network and other control functions may be extremely important compared to either a programmable logic controller, or a robot, or a numerical control machine. In such a situation, it would be appropriate to assign a larger value for token holding time for that node;

- another important consideration in selecting timer values for stations is amount of data required to be sent out of that station. This parameter is useful in reducing overall queue lengths at those stations who have large amount of data to be transmitted over the network. Further, reducing queue lengths in the stations improves overall delay in the network;

- in a token bus network, throughput of the network depends largely on the amount of time allowed for transmitting messages. In fact, throughput will be maximum when this time is very large. However, it should be noted that this may lead to very unfair situation of a station monopolizing the whole network resources;

- from the network user point of view, there may be a requirement of maximum time delay tolerable between successive capturing of token in the network. This situation is of particular interest in the case of time critical needs of factory environment. Such a requirement necessitates faster token rotations in the network;

- in a factory environment, it is required to allow for each station to transmit a few packets (especially emergency messages).

With the abovementioned considerations, higher level problem can be formulated as an optimization problem to determine the maximum time a station can transmit messages when they acquire token. The objective of this optimization problem is to maximize overall throughput of the network. However, using the additive property of throughput of a station, overall throughput of the network is maximized by maximizing throughput of individual stations. Higher level problem can now be formulated as a linear programming problem as,

$$\max \quad \sum_{i=1}^{N} C_i \cdot T_i^h \qquad \cdots \quad (3.1)$$

subject to

$$(i) \quad \sum_{i=1}^{N} T_i^h \leq T^R \qquad \cdots \quad (3.2)$$

$$(ii) \quad T_i^h \geq K \cdot L_s \quad \forall \; i \in N \qquad \cdots \quad (3.3)$$

$$(iii)\ T_i^h \leq D_i \quad \forall\ i \in N \qquad \cdots \quad (3.4)$$

Cost coefficients C_i in (3.1) depends on the priority of station i and load demand at the station i. A possible expression for C_i can be chosen as,

$$C_i = P_i \cdot D_i \qquad \cdots \quad (3.5)$$

expression for C_i in (3.5) indicates assignment of larger value for timer to those stations which are prioritized in the network and for those stations which have larger amount of data to be sent over the network.

Constraint (3.2) is imposed to take care of maximum rotation time tolerable. Constraint (3.3) is imposed to ensure a minimum throughput from each station and to enable stations to transmit at least a few packets. Constraint (3.4) is imposed from efficiency consideration by assigning timer values to be enough to meet the demand.

It is clear from the problem formulation that solution to (3.1) maximizes overall throughput of the network and at the same time satisfies several requirements in a factory environment.

4. LOWER LEVEL PROBLEM:

In lower level problem, time allocated for message transmission by a station is further distributed among various access classes in a station. A solution to this problem results in selection of appropriate target rotation times.

As described in the specifications, there are four different access classes in each station. The priorities of these classes are given as 0, 2, 4, and 6. Associated with priority 6 is high priority token holding timer which indicates maximum time allowed for transmission of high priority packets. For other classes, there are target rotation timers which dictate transmission of those priority packets.

As opposed to higher level problem, in lower level problem, timer assignment is posed as a buffer utilization problem. This is due to the fact that in a factory network,

a protocol of the nature of MAP necessitate presence of transport layer protocol which ensure a reliable end-to-end data transfer. Hence by limiting buffers available at data link level to the extent that amount of packets can be transmitted when token is acquired leads to efficient buffer utilization.

There are several considerations that are to be given while formulating lower level problem. Some parameters that influence timer selection at this level are:

- how important an access class is for a station. This parameter changes weight factor used for access class. Typically, a network control center may have a smaller relative weights as opposed to a numerical control machine station where cost of high access class is several orders larger than lower access classes;
- another parameter is amount of data belonging to a particular access class that is to be sent from a station;
- actual buffer available at a station.

With the abovementioned considerations, lower level problem can be formulated as an optimization problem to determine buffer allocations and corresponding timer values. A solution to this problem maximizes buffer utilization at a station while satisfying constraints imposed at a station. Lower level problem is formulated as a integer programming problem as,

$$\max \quad \sum_{j \in P} y_i^j \cdot b_i^j \qquad \cdots (4.1)$$

subject to

$$(i) \quad \sum_{j \in P} b_i^j \leq B_i \quad \forall \ i \in N \qquad \cdots \quad (4.2)$$

$$(ii) \quad \sum_{j \in P} G_i \cdot b_i^j \leq T_i^h \ \forall \ i \in N \qquad \cdots \quad (4.3)$$

$$(iii) \quad b_i^j \leq d_i^j \cdot g \qquad \cdots \quad (4.4)$$

Cost coefficient y_i^j in (4.1) can be selected based on various parameters such as, how important an access class is, demand for each class etc. However, in the present case, $y_i^j = p_i^j$ is what is selected as cost coefficient.

Constraint (4.2) follows from total buffer available at a station. Constraint (4.3) is a goal coordination between higher level problem and lower level problem by not letting timer values to exceed the solution obtained by solving higher level problem. G_i in (4.3) depends on data rate and packet length used in the network. Constraint (4.4) is imposed from actual load demand.

After obtaining a solution to (4.1), different timers in a station are assigned as follows.

$$T_i^6 = b_i^6 \cdot G_i \qquad \cdots \quad (4.5)$$

$$T_i^j = \sum_{i=1}^{N} T_i^h + b_i^j \cdot G_i \quad \forall \quad j \in \rho \qquad \cdots \quad (4.6)$$

Solution to lower level problem while maximizing buffer utilization, generate timer values in a station.

5. HIERARCHICAL POLICY:

Solution to timer assignment problem is sought by obtaining solution to higher level and lower level problems. It is evident from the problem formulations, higher level problem is solved in a centralized fashion and lower level problem in a distributed fashion. Some of the salient features of this hierarchical policy are described in the following.

This scheme allows for multiple objectives to be considered independently in the optimization process. In the present case, throughput and buffer utilization are considered as objectives. By suitably decomposing the decision making capability to various levels, additional objectives such as, fairness among stations, average delay for packets can also be considered.

Another feature of this scheme is consideration of parameters specific to a local area network environment such as, factory automation network or office automation network. In the present case, for an factory automation network, priority of a station (based on whether machine attached to a station is network controller, or robot, or

programmable logic controller etc.), load demand rates, and priorities of messages at each station are considered.

In addition, throughput maximization (higher level problem) is solved as a centralized problem and buffer allocation (lower level problem) a distributed one. Information required for throughput maximization is of global in nature and the other problem requires local information only. Further, buffer allocation problem also requires results obtained by solving throughput maximization problem.

Lastly, decomposition of timer assignment problem into centralized and distributed problems allows for exploiting time scale differences between the two. Throughput maximization problem can be solved at a slower time scale compared to buffer utilization problem. This is due to the fact that nature of changes in parameters such as, how many stations in the network, overall demand rate at a station etc., are very slow compared to changes in demand for prioritized messages at a station. This is especially true in the case of time critical nature of factory automation networks. Hence in order to provide a quick response to fastly changing parameters, buffer utilization problem is solved in a distributed fashion using only local information. This property provides a cost effective solution to timer assignment problem.

A step by step algorithm:

Step 1: At every T_{up}, solve linear programming problem (3.1) to obtain $T_i^h \ \forall \ i \in N$ using average demand rates, maximum tolerable token rotation time, and minimum time allowed for each station to transmit messages. Broadcast T_i^h to all stations.

Step 2: At every $\tau = n \cdot T_{up}$, solve integer programming problem to obtain $b_i^j \ \forall \ i \in N$ and $j \in P$, using demand rates for each priority, available buffers, and T_i^h value transmitted from higher level problem.

Step 3: Compute $T_i^j \ \forall \ i \in N, \ j \in P$, using (4.5 and 4.6).

Some remarks on this solution:

Remark1: Solution is obtained with the presumption of existence of a mechanism to transmit demand rates to network control center to solve higher level problem. This can be achieved through a network management function [ISO84].

Remark2: Solution obtained is sensitive to both data rate and packet length used in the network.

Remark3: Solution obtained is based on time average values averaged over certain time intervals. Hence this hierarchical policy is a quasi-static one.

Remark4: Solution is obtained under assumption of heavy load on the network i.e. stations are always prepared to transmit a packet.

6. SIMULATION RESULTS:

A network shown in figure 1 is used for simulating this hierarchical policy. For the sake of comparison, this scheme is simulated using standard mathematical programming packages with three different values of maximum token rotation time tolerable (T^R = 30msec, 60msec and 90msec). Some parameter values used in simulation of the network are, slot-time = 45.6μsec, headend latency = 10μsec, packet length = 1024 bytes, buffers available at each station = 16k, and data rate = 5Mbits/sec. Minimum throughput coefficient K was selected to be 5. Priority cost for stations 1, 2, 3, and 4 were selected to be 1, 2, 3, and 4 respectively. Priority cost for access classes 0, 2, 4, and 6 were selected to be 1, 2, 3, and 4 respectively.

Results for maximum token holding time T_i^h for each station with three different values of T^R for varying load demands are shown in figures 2-5. While changing load demand values for a station, other stations are assumed to have a nominal load demand. It is evident from these figures that, as the maximum rotation time requirement is smaller, T_i^h values for stations having lower priority in the network

becomes smaller decreasing overall throughput of the network. On the other hand, when maximum rotation time requirement is larger, T_i^h values for lower priority also increases thus increasing overall throughput of the network.

In an another set of simulation, values of K were varied from 2.5 to 7.5. Again, T_i^h values are computed for all stations. Results of T_i^h for varying load demands for each station with three different values of K are shown in figures 6-9. Results of this simulation show that there is not much effect on throughput when K is varied. This is due to the fact that a constraint of the form T^R controls the distribution of T_i^h values between lower and higher priority stations and has least effect when K is varied with a constant T^R value.

Using simulation package reported in [PIME85], the network shown in figure 1 was simulated by assigning timers using this policy. T_{up} = 30secs and r = 10secs were used for updating timer values in this policy. Both throughput and bus-utilization is shown in figure 10. It is evident from this graph that performance of the network can be improved greatly using this policy for updating timers.

7. CONCLUSIONS:

A hierarchical policy for assigning timer values in a IEEE 802.4 network is presented. This policy provides an ideal framework for considering multiple objectives. Simulation results have shown considerable improvements in performance using this policy for timer assignments which otherwise would have taken a very expensive simulations to achieve proper timer values. This policy can be used to assign timers in a quasi-static fashion to react to changes in parameters at a station. Computation of these changes to parameters would have been extremely difficult using simulation approach.

Several extensions to this policy are possible. Firstly, by changing lower level objective function to include in its formulation the solution obtained in higher level problem. This would result in a better goal harmony between the two problems. Secondly, a performance evaluation of this hierarchical policy is required to analytically

determine time periods T_{up} and τ. Lastly, additional objectives can be included in problem formulations.

Thus hierarchical policy for timer assignments in IEEE 802.4 network provides an ideal framework for solving some parameter assignment problems to improve performance of the network considerably.

REFERENCES

[IEEE84] IEEE Standard 802.4 Token Passing Bus Access Method and Physical Layer Specifications, Draft F, July 1984.

[MAP85] Manufacturing Automation Protocol, Version 2.0, March 1985.

[ISO84] ISO Standard 7498 Information Processing Systems - Open Systems Interconnection - Basic Reference Model, Oct. 1984.

[McGA85] Susan L. McGarry, " Networking has a Job to do in the Factory", Data Communications, Feb. 1985, pp 119-128.

[STAL84] William Stallings, " Local Network Performance", IEEE Communications Magazine, Vol. 22, No. 2, Feb. 1984, pp 27-36.

[MURA84] K.H. Muralidhar, " Hierarchical Schemes for Routing and Flow Control in Large Communication Networks", Ph.D Dissertation, Department of Electrical and Computer Engineering, University of Arizona, 1984.

[RAHI83] Said K. Rahimi and George D. Jelatis, " LAN Protocol Validation and Evaluation", IEEE Journal on Selected Areas of Communications, Vol. SAC-1, No. 5, Nov. 1983, pp 790-802.

[ARCH84] Jean-Luc Archambault, " An IEEE 802.4 Token Bus Network Simulation", NBS IR 84-2966, National Bureau of Standards, Oct. 1984.

[PIME85] Juan R. Pimentel, " Simulation of IEEE Token Bus Protocol Using SIMAN", to be presented at Workshop on Analytical and Simulation Modeling of IEEE 802.4 Token Bus, NBS Gaithersburg, MD, April 29-30, 1985.

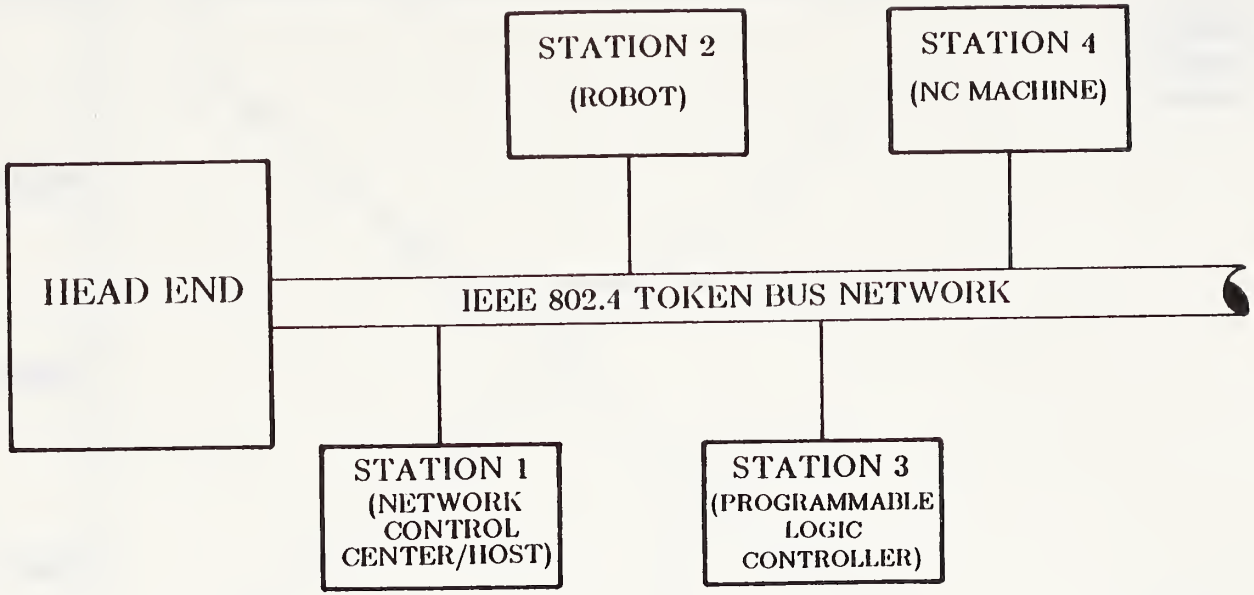


FIGURE 1. NETWORK USED FOR SIMULATION

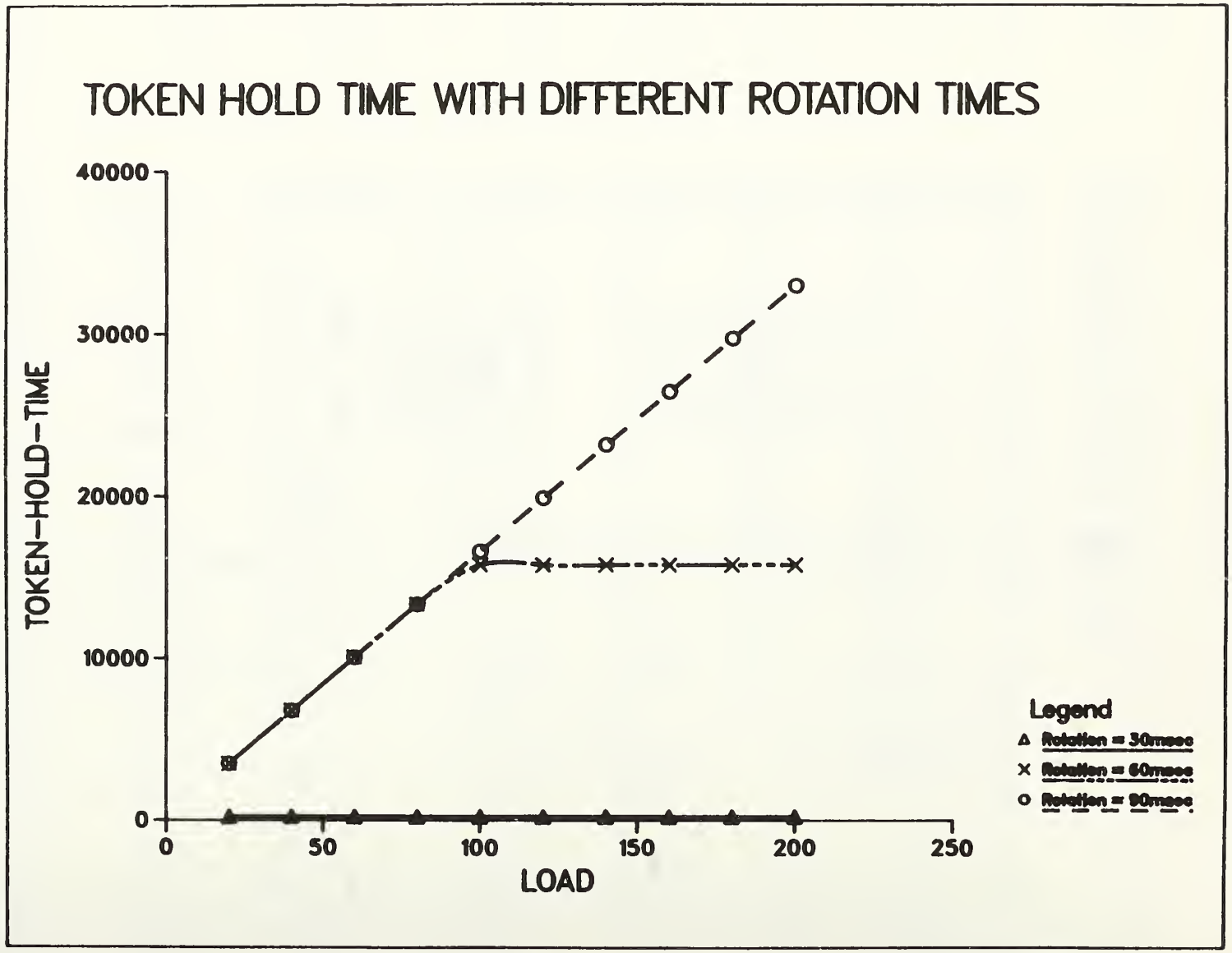


FIGURE 2. STATION 1

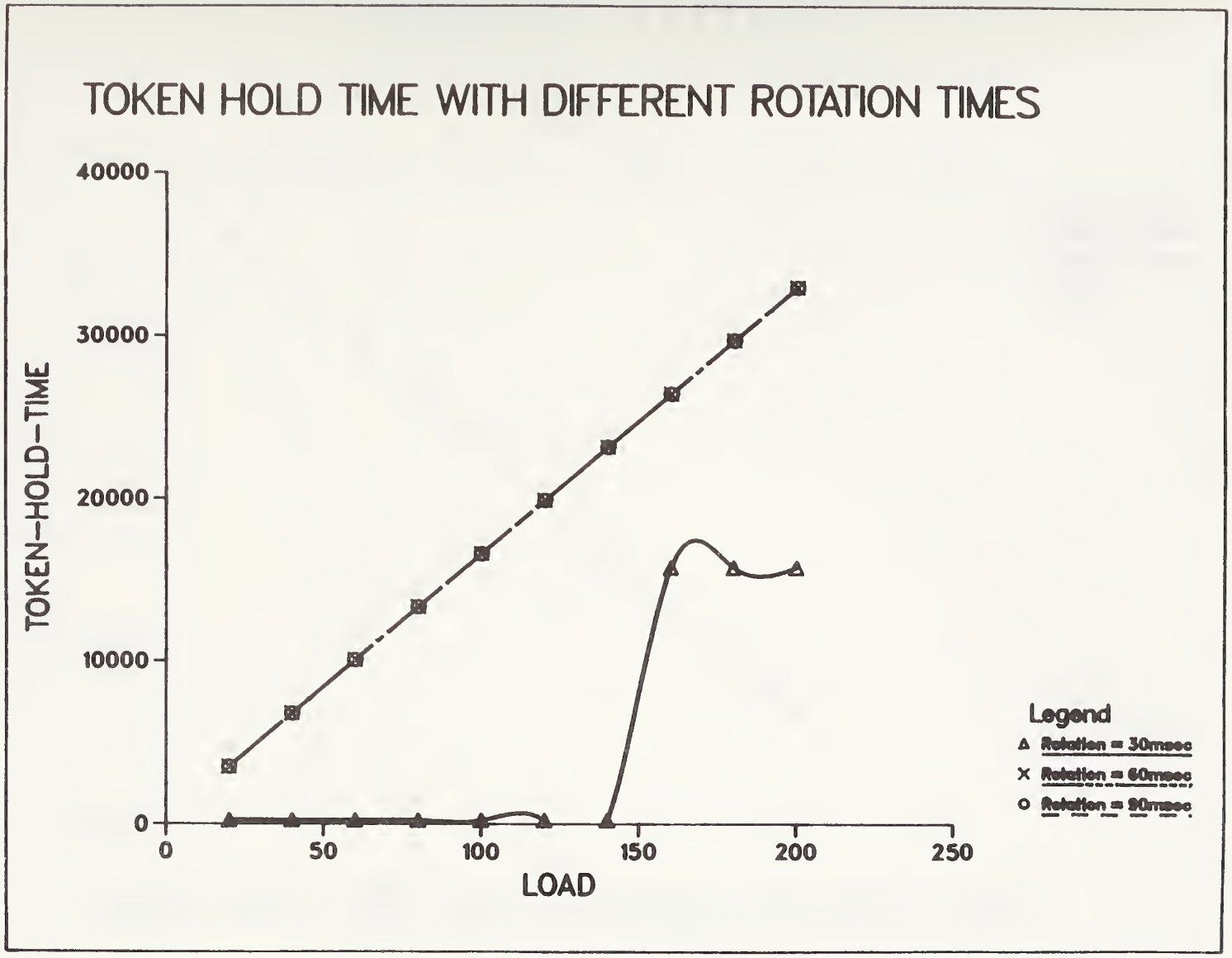


FIGURE 3. STATION 2

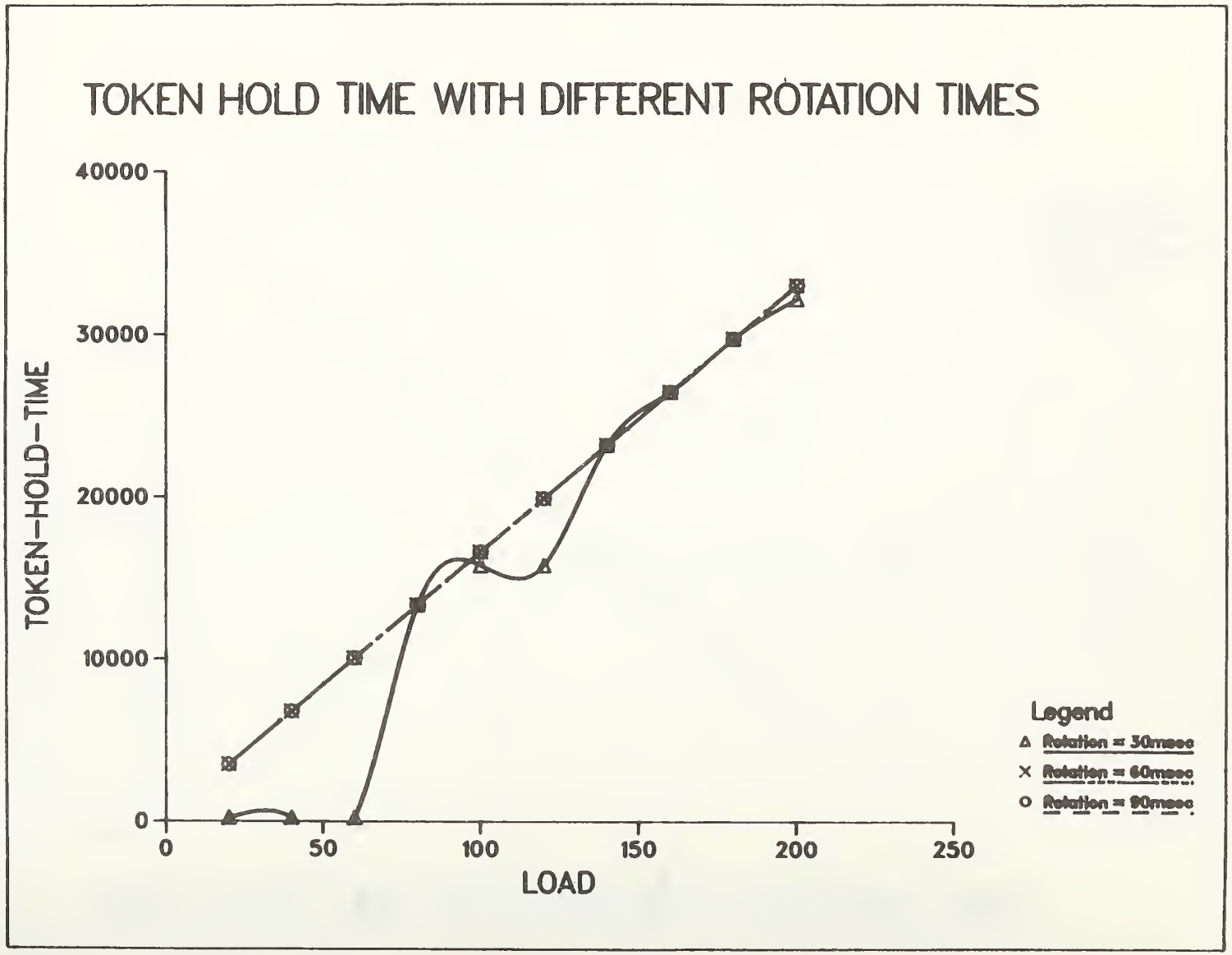


FIGURE 4. STATION 3

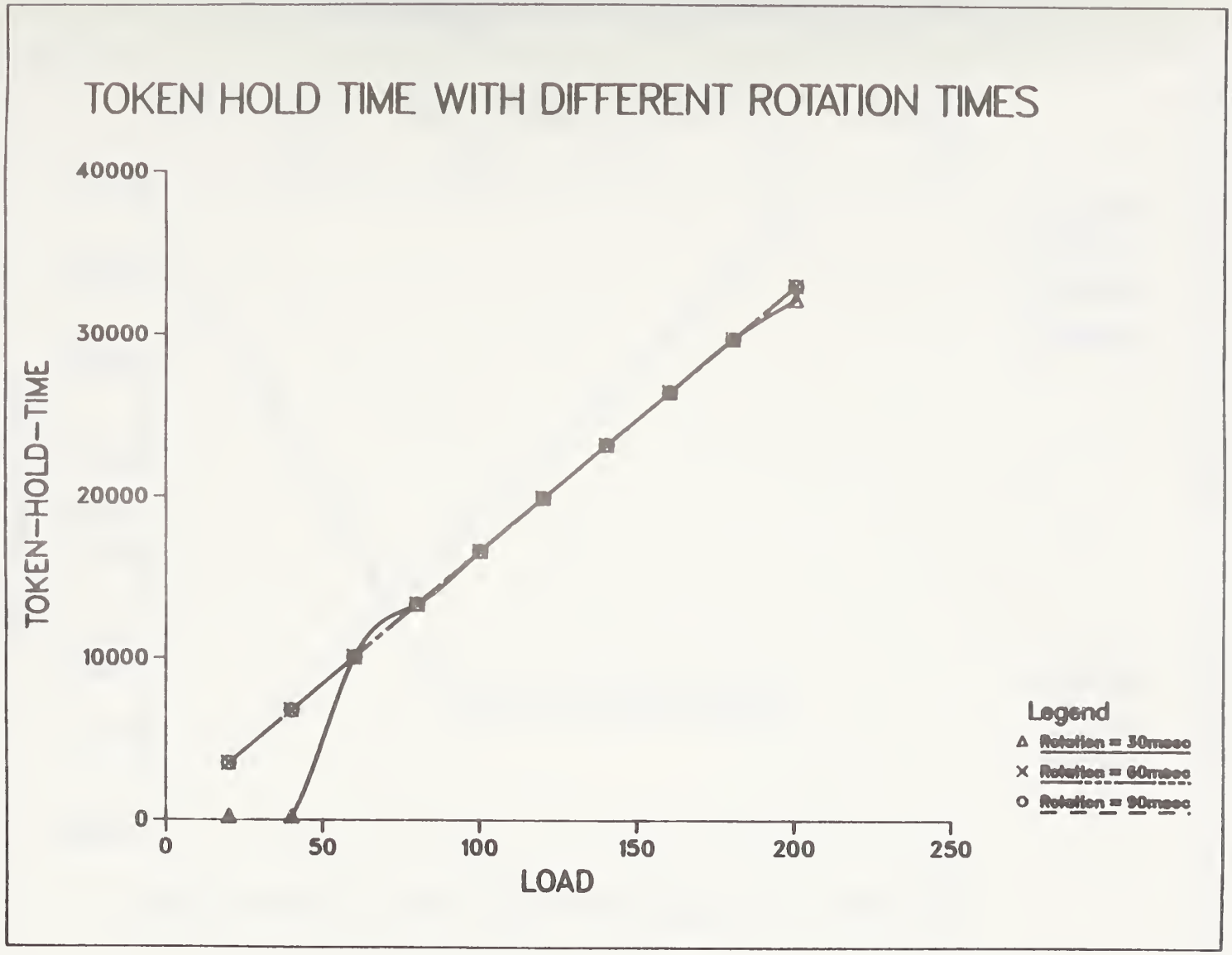


FIGURE 5. STATION 4

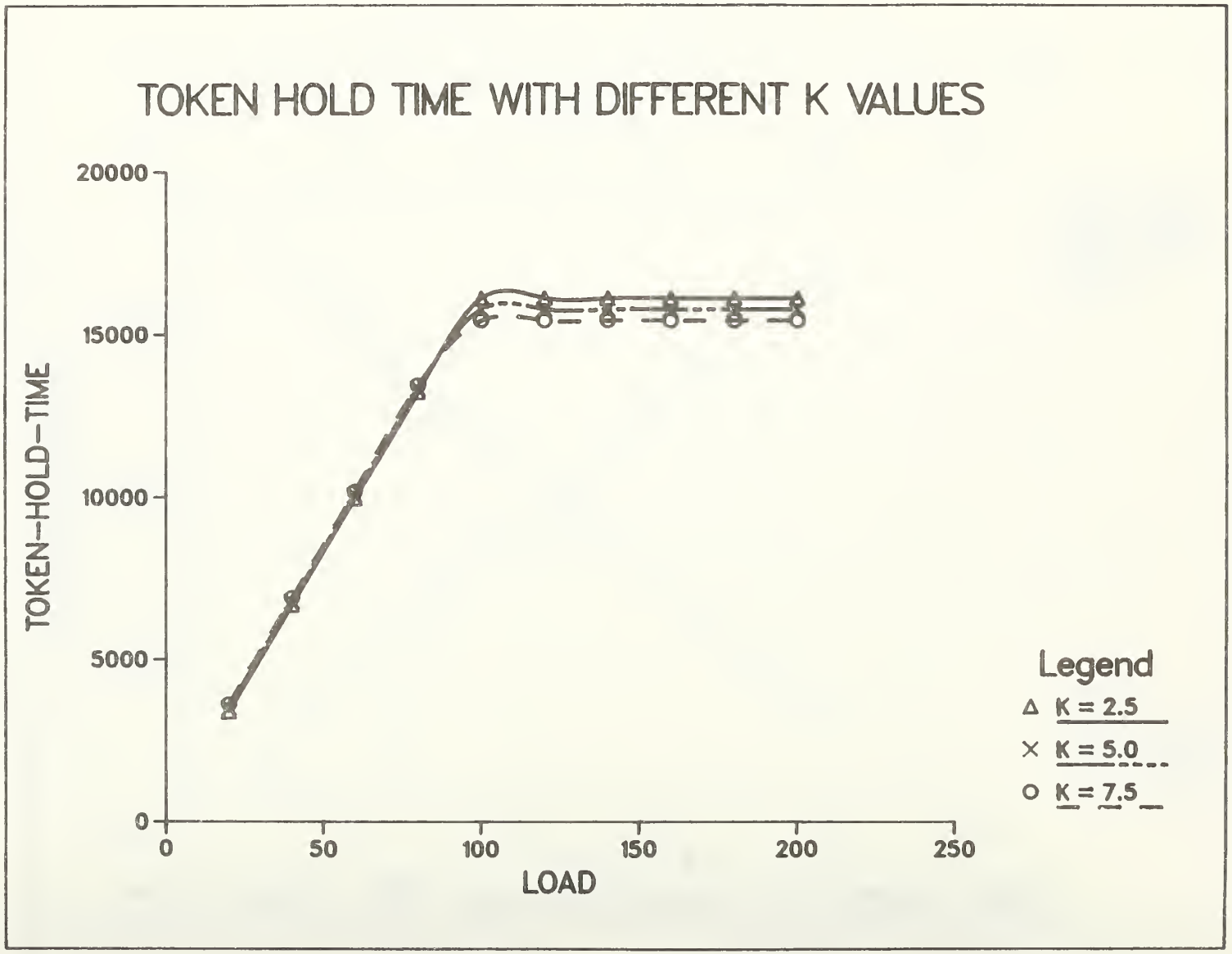


FIGURE 6. STATION 1

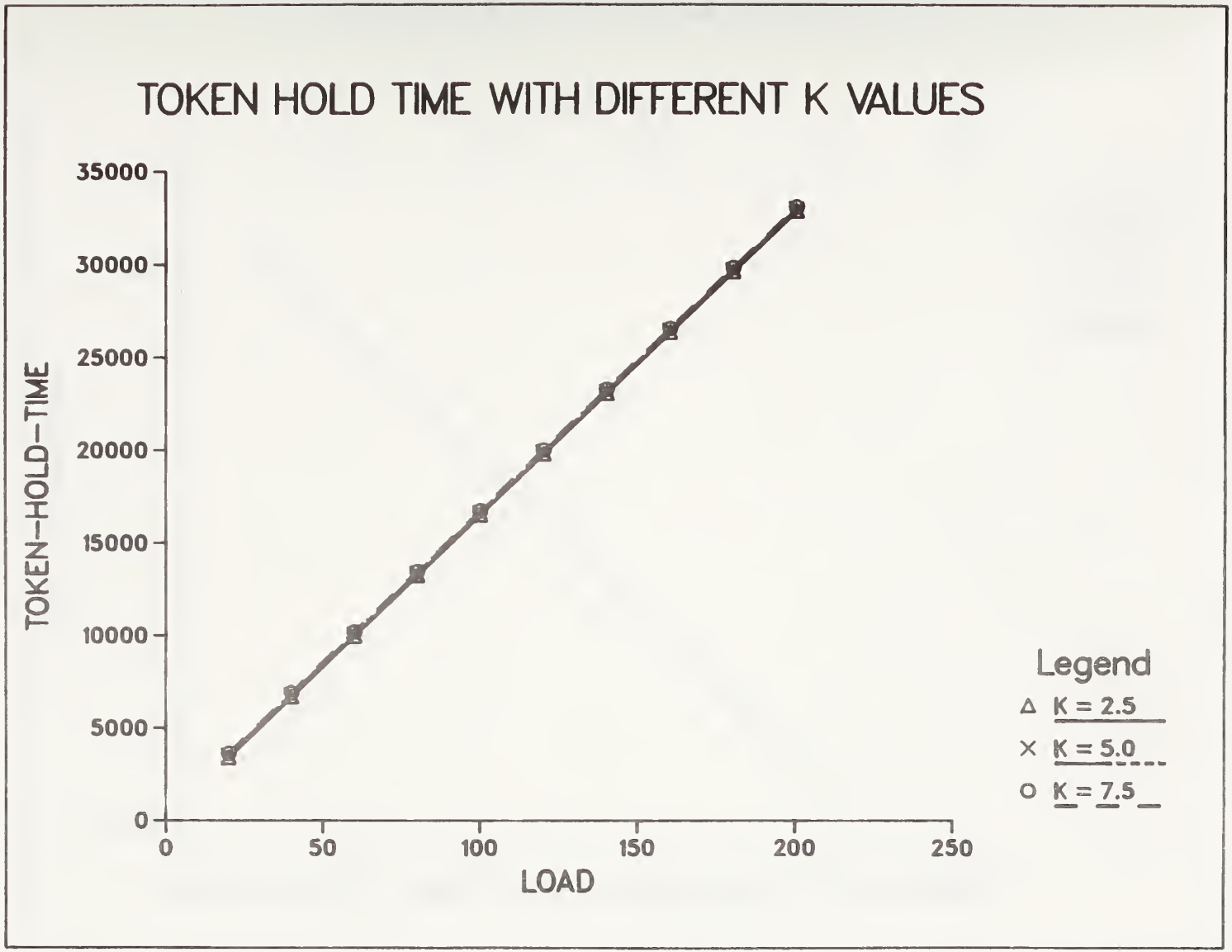


FIGURE 7. STATION 2

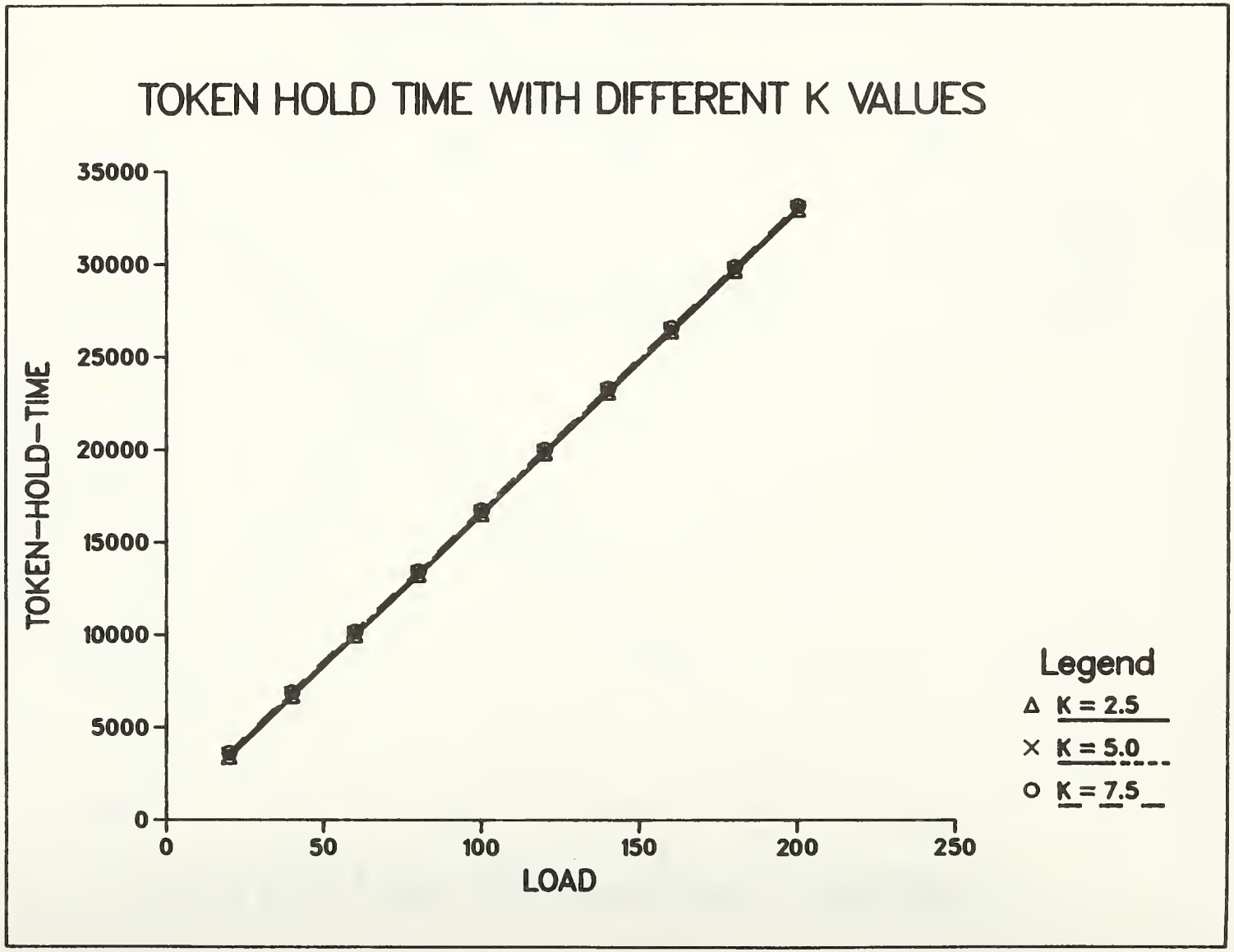


FIGURE 8. STATION 3

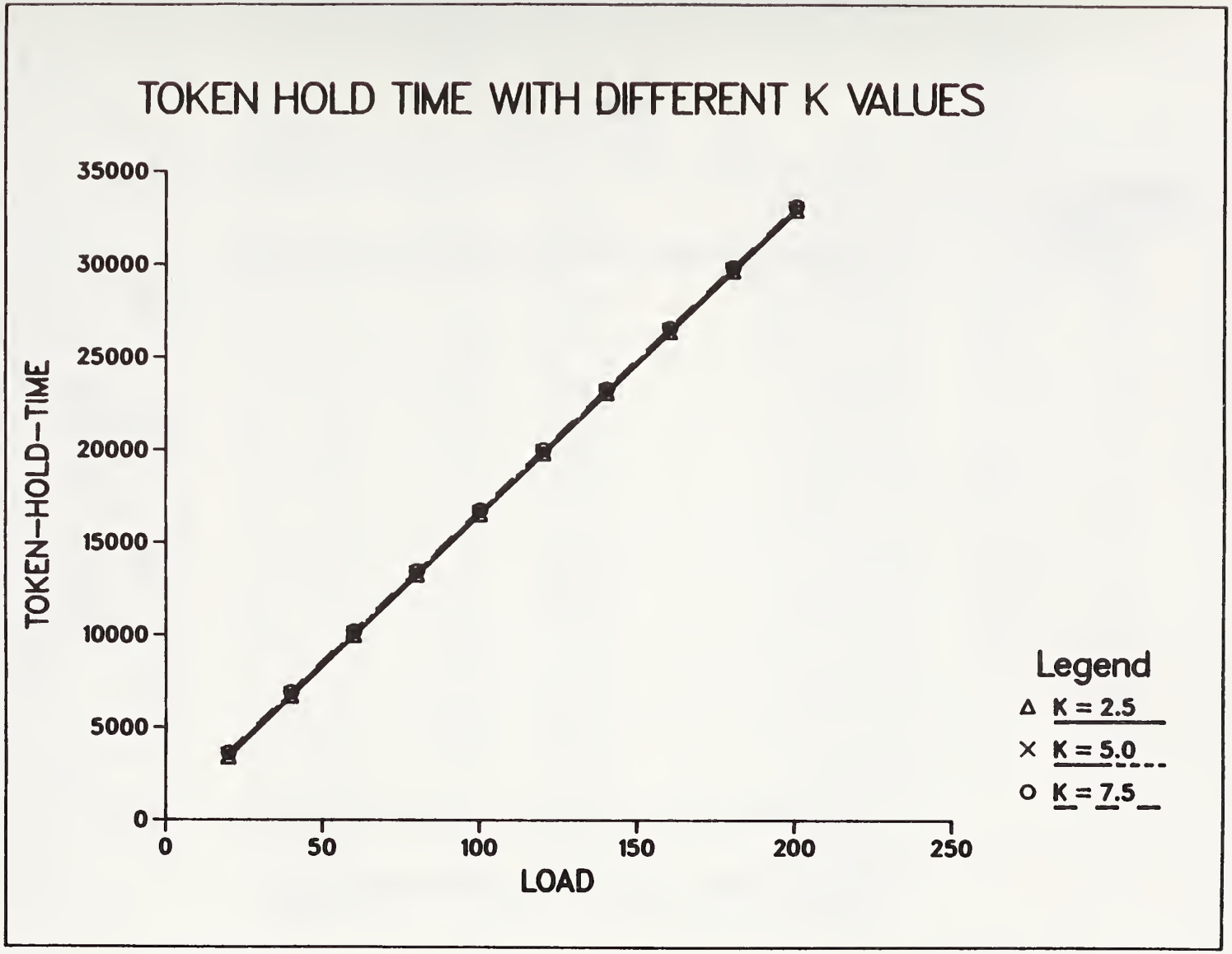


FIGURE 9. STATION 4

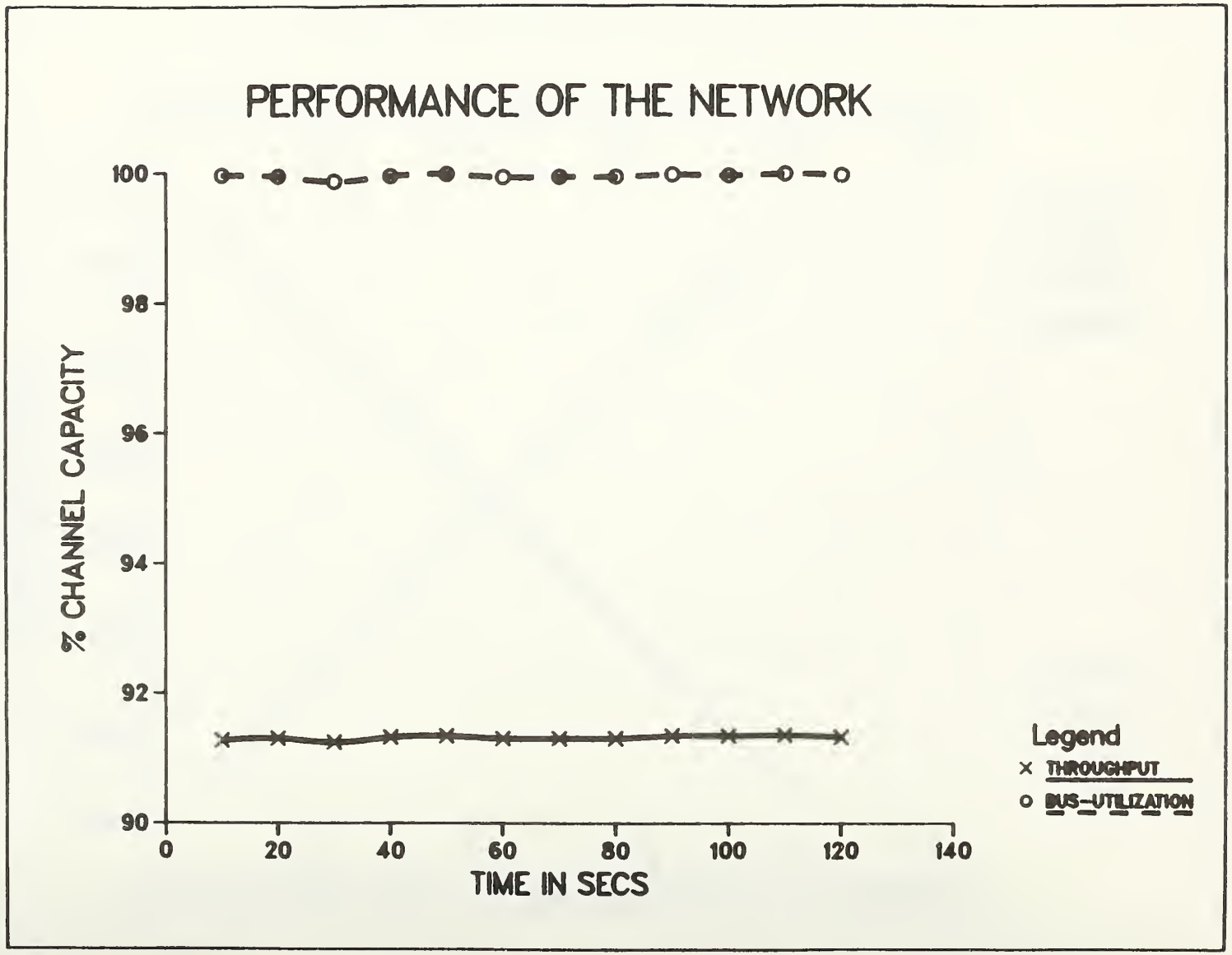


FIGURE 10. PERFORMANCE

On the Stability of a Token Passing Network.

by Anastase Nakassis
Institute for Computer Science and Technology
National Bureau of Standards, Dept of Commerce.
Building 225, room B221.
Wash D.C. 20234.

Abstract: In what follows we will study the stability of token passing networks with a fixed number of queues and we will deduce the average rotation time for the token and the average usage time per queue, under the assumption that the system is stable. These results will then be used to derive system parameters that will make the network stable.

INTRODUCTION.

In what follows we plan to study the performance of a local area network that implements the IEEE 802.4 standard (July 1984, draft F). The questions that we will address and attempt to answer revolve around the following theme:

" Given the arrival and departure rates for every queue in the network, and given the system's parameters (token holding times, Target Rotation Times and token passing times) can we decide if the network is stable?"

As we will see, there are cases in which we can decide if the network is stable or not (a network is stable if and only if the expected length of every queue in the network is finite) on the basis of the information listed above. But, there are cases in which more information is needed, i.e. we need to know the distribution of the interarrival times and, possibly, the distribution of the size of the messages in each queue.

In order to simplify this problem, we will study in what follows networks with a fixed number of nodes/queues/stations and we will study a generalization of the IEEE standard in order to ignore details that are are not relevant to this work. We will assume then that we have k queues named 1,2,3,...,k, and that the token is passed from queue 1 to queue 2, to queue 3,...,to queue k, to queue 1, ... and so on. By "token passing" we understand something more general than what the term usually denotes (passing the token from one station to the next). In this text, token passing means passing the "right of transmission" and it takes place from queue to queue not from station to station. Actually, this study does not distinguish between stations and queues. The queues are either of high priority, in which case they can retain the token for up to h[i] time units per rotation, or of low priority in which case they can hold the token for up to TRT[i]-R time units per rotation (TRT[i] being the queue's target rotation time and R the duration of the previous rotation as seen by queue i). In what follows we assume that we have been given two sequences

- h[i] and TRT[i] , i=1,2,3,4, ..., k - such that:

  1. If queue i is of high priority, then h[i]>0 and TRT[i]=0.
  2. If queue i is not of high priority, then h[i]=0 and TRT[i]>0.

  Evidently, h stands for token holding time and TRT for Target Rotation Time and these two sequences fully describe the queues of our network.

Notice that we do not make any assumption concerning the distribution of high priority queues, although the IEEE 802.4 standard (July 1984, draft F) on which this work is based does imply some distribution.

In what follows, we introduce a set of parameters needed in order to describe the actual behavior of the network and an additional assumption, item (7), that we need in order to describe the exact conditions under which a station is passing the token. Indeed, the following question arises: Assume that queue i has the token and a message to send ; assume also that the queue has not exhausted the time that it was allocated by the protocol, but that if it attempts transmission, then it will exceed it. Will it attempt transmission or not? Whatever the answer, this situation will drive a wedge between the nominal and actual token usage times (i.e. the times the protocol allocates and the actual usage times) and, in some instances, it will greatly complicate the study of the network. For this reason we will introduce item (7) whose purpose is to simplify the network's behavior. Assume therefore that:

1. D is the time needed in order to pass the token from queue to queue during a complete revolution,
2. a[i] is the expected number of arrivals at queue i during a time unit,
3. d[i] is the number of messages sent by queue i within a time unit, (it is implicit in d's definition that all messages in a given station are of the same length. This assumption is not needed but it will be maintained for reasons of simplicity of exposition. We note that when this assumption is lifted, then the distribution of the messages' lengths may, under the right circumstances, affect the stability of the network).
4. f[i] is by definition a[i]/d[i],
5. T is the average time needed for a complete revolution,
6. T[i] is the average time queue i keeps the token during each revolution, and
7. either of the following is true:

   a. the system parameters are such that when the protocol assigns to a queue a certain time t, then the queue will never exceed this time. For most systems, this is tantamount to saying that t is a multiple of d[i].

   b. the queue length is seen as a continuous variable so that each time t is used to its fullest extent.

   c. the system parameters can be altered in such a way that:
      c1. The original and the modified system behave in the same way. For instance: assume that h[1]=2.5, that d[1]=1, and that the system will attempt a transmission if it has messages to send and it has not exhausted its holding time. In that case we can assign to h[1] any value in the interval (2,3] without modifying the

system's behavior. And that
c2. the new system satisfies a. above

The assumptions under item (7) above are not always needed and after each proof we will include remarks as to which expression(s) and/or which part(s) of the proof depend on any of the above assumptions.

MAIN RESULTS.

We start by observing that even if some or all network queues are unstable (the expected length of the corresponding queues tends to infinity as time tends to infinity), all T[i]'s, i=1,2,3,...,k, are well defined. Indeed, by the protocol's very definition, the token usage at any given queue i and during any rotation is bounded by max {h[i],TRT[i]-D}. Furthermore, since

$$T = \sum_i T[i] + D,$$

T is also finite.

Lemma 1. If all queues are stable, then

$$T=D/(1.0-\sum_i f[i]).$$

It ensues that a necessary condition for stability is that the f[i]'s sum to less than one.

Proof: If the system is stable, then for each queue the number of arrivals [messages put in the queue] must equal the number of departures [messages transmitted]. In general, the arrivals are no less than the departures and by averaging we obtain d[i]*T[i]<=a[i]*T, or T[i]<=f[i]*T. If queue i is stable, then T[i]=f[i]*T. We know that stability or not,

T=D+T[1]+T[2]+ ..... +T[k]

Therefore, if all queues are stable, then

T=D+T(f[1]+f[2]+ ... +f[k]) and
T=D/(1.0-f[1]-f[2]- ... -f[k]).
Moreover, T[i]=f[i]T=f[i]*D/(1.0-f[1]-f[2]- ... -f[k]).

(end of Proof)

The problem is to ascertain if the system is stable. The values for T and T[i] were obtained under this very assumption. But it may well be that the parameters of the system are such that a particular queue is not stable. For example, if T[i]>h[i]>0, then the holding time is too small and the queue is unstable. As we will see below, one can derive simple sufficient conditions for the system to be stable. One may also find necessary conditions for system stability. But, it is hard to find conditions that are both sufficient and necessary. Indeed, as we will see shortly, the token usage by the low priority queues depends not only on the average birth and death rates, on the h[i]'s and the TRT[i]'s, but also on the distribution of the token usage times for each queue.

Lemma 2. For a given T, a high priority queue i is stable provided that h[i]›f[i]*T.

Proof: If the queue is a high priority queue, then during each revolution it will keep the token for h[i] time units or for as long as it needs it, whatever is less. But if h[i]›f[i]*T, then it can not consistently hold it for h[i] time units, because then the departures will exceed the arrivals. Thus, whatever the queue's present length, we are assured that sooner or later the queue will be empty and it is quite obvious that the time at which the queue will be empty has a finite expected value.
One can also see that if h[i]‹f[i]*T, then the queue is unstable and that in this case T[i]=h[i]‹f[i]*T.
Finally, it remains to examine the case of h[i]=f[i]*T.
A statement about the queue stability cannot be made unless one has more information on the arrival distribution. We note though that in the Markov case, when the birth and death rates are equal, the queue is unstable. Furthermore, from an engineering point of view, one can not know the precise values of our variables. In other words, we can never know if we have equality and we cannot distinguish between the case of h[i]=f[i]*T and h[i]‹f[i]*T.

(end of proof)

Remark: If condition (7) is not satisfied, then the lemma's sufficient condition holds true, provided that a transmission will always be attempted while messages are present and h is not exhausted. On the other hand, the condition will not be necessary unless a queue refrains from transmissions that will result in token holding in excess of h[i]. This is not only inconsistent with the current protocol, it is a condition that can not be satisfied unless one knows ahead of time the transmission's length. Nevertheless, we know that the token cannot be held for more than h[i]+d[i] time units. Given that our time is measured in increments of a quantum q, and given that the actual holding time cannot exceed h[i]+(last transmission's duration), we can state that if

T[i]›=h[i]+d[i],

then the queue is unstable.

Lemma 3 : Assume that the average token rotation time is T and that a low priority queue, i, is unstable.
Then T+T[i]›=TRT[i].

Proof : If queue i is unstable, then it will use the token as much as it can because its queue will have a tendency to be longer as time passes and the probability that said queue will clear tends to zero as time tends to infinity. Let us assume that queue i is the last queue, and that during rotation j, j=1,2,3,... , queue i held the token for t[j] time units while all the other queues combined held it for s[j] time units. Then we have:

t[j-1]+s[j]+t[j]+D›=TRT[i]

for almost all j (that is for all j, j›1, for which queue i is not empty and its length exceeds some constant length m).

Therefore, if we take the averages we will obtain

$$T[i]+T \geq TRT[i],$$

because $E(t[j-1])=T[i]$ and $E(s[j]+t[j]+D)=T$.
[E(.) is the expected value of whatever happens to be within the parentheses].

(end of proof)

Remark: The assumptions in (7) are not needed provided that the queue will not pass the token before it has exhausted all allocated time, unless it is empty (therefore, the last transmission may extend the actual holding time somewhat beyond the difference between TRT[i] and the time that elapsed between the previous token entry to queue i and the present one).

Remark: Given that the actual token usage of a low priority queue is affected by the token usage of every other queue, it is quite difficult to establish if low prioriry queue i unstable or not. Sometimes one may be able to show that at least one low priority queue is unstable while being unable to determine which ones actually are. What follows are a few fairly straightforward observations:

If t[j-1] and t[j] are the actual token usage by low priority queue i during rotations j-1 and j, then

$$t[j-1]+D+t[j] \leq TRT[i]$$

(if (7) does not hold, then $t[j-1]+D+t[j]<TRT[i]+d[i]$). Therefore, there is an obvious upper bound to the token usage of low priority queue i, namely (TRT[i]-D)/2.0. Clearly, this bound is very crude and does not take in consideration the rotations during which the other queues are active. In any event, in order to have stability, it is necessary that $TRT[i] \geq 2.0*T[i]+D$ for every low priority queue i.

If the TRT and h values are such that it is admissible to have each queue i hold the token for T[i] time units in each rotation, then $TRT[i] \geq T+T[i]$. Therefore, under the conditions outlined above, and if one wishes that the low priority queue be stable ragardless of the arrival/usage patterns of the other queues, then for every low priority queue i, one should choose a value TRT[i] such that: $TRT[i] \geq T+T[i]$.

Finally, let us remark that up to this time we looked at low priority queues in isolation. Given a set of values for a[i], d[i], h[i], and TRT[i], one can conceivably show that for every admissible sequence of token usage times, at least one low priority queue will be unstable while there are sequences that make each specific queue stable. It would therefore be useful to attempt to find inequalities binding not a single queue's TRT, but all TRT's at the same time. In other terms one might come with a function of all the above quantities, p(a,d,h,TRT), such that whenever $p<0$, then any possible sequence of message arrivals (token usage) would leave at least one queue unstable. To give an

example:

Assume that a network consists of two low priority queues with target rotation time R. Then for every possible admissible sequence of token usage times, the average token usage for one of the queues will be (R-D)/3 or less. Thus, unless R›=3.0*min{T[1],T[2]}+D, at least one of the two queues will be unstable.

Lemma 4 : A sufficient condition for a network to be stable is that

a. f[0]+f[1]+...+f[k] be less than one, that
b. For all high priority queues, h[i]›f[i]*T, and that
c. for all low priority queues, TRT[i]› (1+f[i])T

where T=D/(1-f[0]-f[1]-...-f[k]).

Proof : Let P be the average rotation time. If one or more queues are unstable, then their average token usage per revolution is f[i]P or less. Therefore,

P=T[0]+T[1]+...+T[k]+D‹= (f[0]+f[1]+...+f[k])P+D

and if we solve for P, then we see that P‹=T. Since h[i]›f[i]T, h[i]›f[i]*P, and high priority queue i is stable. Similarly, if i is a low priority queue, then P+T[i]‹=T+f[i]P‹=T+f[i]T‹TRT[i].
Therefore, according to our previous lemma, queue i cannot be unstable.

(end of proof)

Remarks: (1) Condition (a) is necessary for stability (see Lemma 1.).
(2) Under most circumstances, condition (b) is also necessary for stability, provided that (7) holds (see Lemma 2.). When neither (b) nor (7) are true, then either we need more information, or we can prove that the network is unstable.
(3) As we will see later, when this condition is not satisfied, then either we need more information, or we can prove that the network is unstable.
(4) The above lemma holds even if (7) is not true.

As noted above, the condition for stability on the low priority queues is sufficient but not necessary. The reason for this is that we have a complex scheme that determines how much token usage each low priority queue gets at each rotation and as the following examples will indicate, the stability or instability may hinge not only on the average usage by the stable queues, but also on the distribution of token usage.

Example 01 :

There are two high priority queues (0 and 2) and two low priority queues (1 and 3). We assume that D=1.0, that TRT=8.0 for both low priority queues, and that both high priority queues are stable with average token usage 2.0 time units per revolution. Then the following token usage sequences are possible:

a. Both low priority queues are saturated:

```
.....................
3    0      3    0
3    1      1    2
1    2      3    0
1    1      1    4
3    0      3    0     etc.
```

with T[0]=T[2]=2.0, T[1]=1.0, T[3]=1.5, and T=7.5

b. Both low priority queues are saturated.

```
.........................
4    0   2   1
0    4   2   0
4    0   2   1     etc.
```

Again T[0]=T[2]=2.0 and T=7.5, but now T[1]=2.0 and T[3]=0.5.

c. Queue 3 is saturated, but 1 is not, and has the token usage pattern shown below:

```
.................................
4   0   2  0
0   3   2  2
4   0   2  0    etc.
```

In this instance T[0]=T[2]=2.0 while T[1]=1.5, T[3]=1.0, and T=7.5.

(Keep in mind that since D=1, we can derive the holding times for the low priority queues by usind their effective TRT's which equal 8.0-1.0=7.0 ; when T is computed, we average the actual token usage times and we then increment this average by D=1.0).

A cursory examination of those examples shows that there are f values for which queue 1 is unstable (example a) while for the same values of f but different token usage queue 1 is stable (example c). Please notice that the token usage patterns we exhibited cannot be sustained unless restrictions are placed upon the mechanism that creates messages to be sent; while the precise token usage sequence that we exhibited cannot be sustained without unrealistic assumptions are made for the message generating mechanism, what we did was to exhibit a specific instance of a stable/unstable system. It is quite possible that less restrictive mechanisms may also produce stable/unstable systems, but in this case it would be much harder to predict an exact token usage.

Example 02:

In the above example we exhibited patterns of token usage such that for the same f[i]'s, h[i]'s, TRT[i]'s, and D, some of the low priority queues may be stable or all may be unstable. It is natural then to consider the following problem:

Assume a set of parameters and a pattern of token usage that makes the system stable. Is there a different pattern that makes some of the queues unstable?
Or, is it the case that if a pattern makes the system unstable, then every other pattern will also make the system unstable but may displace the instability from a set of queues Q to another

set of queues Q*?

The following examples are meant to answer this question.

We assume that f[0]=f[2]=2/7, that TRT[1]=TRT[3]=8.0, that D=1.0, and that h[0]=h[2]=4.0. We will propose two usage patterns based on the assumption that both low priority queues are unstable. From these assumptions we will derive inequalities for f[1] and f[3]. Finally, these inequalities will show that careful choice of f[1] and f[3] will render one system stable, but the other unstable in both low priority queues.

a. Assume the following pattern:

```
 ..........................
 2  0  2  0
 2  3  2  0
 2  0  2  3
 2  0  2  0      etc.
```

We see then that T=(4+1+7+1+7+1)/3=21/3=7.

f[0]=f[2]=2/7.

The instability of queues 1 and 3 implies that f[1] and f[3] equal or exceed 1/7.

b. Assume the following pattern:

```
 ........................
 49/14   0/14   15/14   34/14
 15/14  34/14   49/14    0/14
 49/14   0/14   15/14   34/14     etc.
```

We see then that T=(49/14+15/14+34/14+1+15/14+34/14+49/14+1)/2=
=(98/14+1+98/14+1)/2=(7+1+7+1)/2=8.0, that

f[0]=(49/14+15/14)/16=(64/14)/16=4/14=2/7, that
f[2]=.....=2/7, and that

instability for queues 1 and 3 implies that f[1] and f[3] exceed or equal (34/14)/16=(2/14+32/14)/16=1/112 + 2/14= 1/7 + 1/112.

Therefore, if f[1] and f[3] exceed 1/7 but are less than 1/7+1/112, then the first token usage pattern implies instability, while the second one implies stability.

CONCLUSIONS.

In what preceeded we ivestigated the stability properties of a token passing network. Our investigation showed that given the ratios of arrival/departure rates, it is possible to assign holding times and target rotation times in a way that makes the network stable if and only if the said ratios sum to less than one. Nevertheless, such assignments may make our TRT's quite big and this is an unwelcome feature. It means that if there ever is an activity peak, a low priority queue may grab and hold the token for a long period of time.

Therefore, one is led to the problem of establishing stability, or the lack thereof for a specific network with given holding times and

Target Rotation Times. Our results can, in some cases establish that a given network (all or some queues of the network) is not stable. Evidently, the most interesting case is the one that cannot be covered by our results. As examples 1 and 2 indicate, it is quite unlikely that we may establish stability or the lack thereof without assumptions on the distribution of the token usage. Given the interdependencies between the actual holding times of low priority queues (protocol specification) as well as the fact that the actual holding time at one queue may determine the length of the next, this appears to be a difficult task. A practical approach may consist of running simulations and/or assuming a specific arrival pattern (for instance a Markovian birth process). It also appears that better results may be obtained through analytical models only if we reach a better understanding of the token usage by the low priority queues.

Another subject that is worth examining more closely is the fact that in a stable system the average rotation time is proportional to D. At first the result seems wrong since it implies that if D=0, then T=0. More careful consideration reveals that a stable system will , sooner or later, reach a state when all queues are empty. At that time, if D were zero, the token would perform infinitely many rotations in a small amount of time, thereby driving the mean rotation time to zero. Since T and all T[i]'s are, in a stable system, proportional to D, another research area is to examine why this relation holds and what it implies for the design of networks. Specifically, one would like to know the distribution of the rotation time as a function of D. One would like to know if there is a range of values for D over which the distribution of (rotation time)/D is independent of D or, at least, not sensitive to changes in D. Simulation results imply that such a range does exist but, as D increases, there comes a moment at which the long rotations can not become any longer and therefore small changes in D will have increasingly more severe impact on the short rotations. Finally, a moment comes when the network becomes unstable unless the system parameters are increased whenever D is increased. Therefore, it would be a worthwhile project to study, through simulations if necessary, the distribution of the rotation time when all factors other than D are held constant. We note that the rotation time as seen by queue i and the rotation time as seen by queue j will not necessarily have the same distribution, even though both rotations will have the same expected value, T. Indeed, the simulations in appendix 2 show that the variance of the rotation time as seen by queue i and the variance of the rotation time as seen by queue j are not equal. A fortiori, the rotation time as seen by i and the rotation time as seen by j do not have identical distributions. Thus, even if one manages to derive the distribution of the rotation times as seen by a specific queue of a specific network, it is not obvious that a general model can be extracted from such distributions.

APPENDIX 1

Proof of the statement that if a network consists of two low priority queues with TRT's equal to R, then for every admissible sequence of token usage times the average token usage of at least one queue is (R-D)/3 or less, D being the time spent in token passing during each rotation.

Consider an admissible sequence

```
x[1]   y[1]
x[2]   y[2]
.....
x[i-1] y[i-1]
x[i]   y[i]
x[i+1] y[i+1]
.....
```

Let S=R-D so that we can ignore D in what follows. Clearly, for each i x[i]<=S and y[i]<=S. Given this, x[i]+y[i]<=S and y[i]+x[i+1]<=S. Indeed, if the y[i] (x[i+1]) is zero, this inequality is true while if y[i] (x[i+1]) is positive the protocol guarantees that x[i]+y[i]<=S (y[i]+x[i+1]<=S). By reasoning along the same lines, we see that x[i]+y[i]+x[i+1]<=S, and that y[i]+x[i+1]+y[i+1]<=S for all i=1,2,3,... . It ensues that if X and Y are the correspond averages, then

2*X+Y<=S and X+2*Y<=S.

If X<=Y, then 3*X<=S or X<=S/3.

Notice: The above results were obtained under the assumption that condition (7) holds. If not, the inequalities will have to be modified so that the extra time during which the last message was sent will be properly accounted.

APPENDIX 2

In this appendix we will give the results of some simulations that we carried out in order to investigate the subjects mentioned in the conclusion above.

In order to examine the validity of the results presented above, I ran several simulations of a token passing network. The simulations were run via a rather simple C program, Ratest, which made the same simplifying assumptions that this paper made. What follows are some of the results obtained. The code of Ratest will be made available upon request.

In each simulation we assume that the mean arrival rate at a queue i oscillates between two values, M1[i] and M2[i], not necessarily distinct. The mean arrival rate remains the same for an interval of mean length T, with T being equal to the mean rotation time under the assumption of stability. The actual length of each of these intervals is w*X+(1-w)*T, where X is an exponential random variable of mean value T, and w is the measure of randomness for the determination of the interval's length. At the end of each interval, the mean value will remain the same with probability p (p=p[i]) and will change with probability 1-p. During each interval of constant mean interarrival rate M [M=M1 or M=M2], the time between two successive arrivals is r*Y+(1-r)*M, where Y is exponentially distributed with mean M, and r is the measure of randomness for the message interarrival times. Intervals with the same M value are thought to be logically contiguous so that if the above computation gives an arrival time beyond the end of the current interval, then the arrival will take place in some future interval with the same M value. One way to visualize this situation is the following paradigm:

Each queue is the outlet for a single CPU computer that is executing two processes, P1 and P2. Once the CPU is given to a process, this process will keep it for at least w*X+(1-w)*T time units. At the end of this interval the process will either retain the CPU for another interval of length w*X+(1-w)*T with probability p, or relinquish the CPU to the other process with probability 1-p. The probability p is supposed to be a function of the queue, but not a function of the executing process. Each process is creating a new message every r*Y+(1-r)*Mi time units with i=1 or 2. Clearly, when the CPU is taken away from a process, the process has most likely already done some of the work needed for the creation of the next message and this work will be remembered when the process will get the CPU back. Thus the activity periods of each process form, from the process' point of view, a continuum.
Some of the results of the simulations run are given below. Notice that in all instances we deal with a network of four queues, two of which are of high priority (#1 and #3), while the other two, #2 and #4, are of low priority. The time needed for token passing from queue to queue is 3.5 and the time needed to send a message is 1.0 time units for all queues and messages. The results follow:

Simulation #1.

In this simulation w=r=0.0 and p=0.0 for each queue (arrivals and CPU holding times and CPU switching probabilities are deterministic). For queues 1 and 3 M1=M2=3.5 while for queues 2 and 4 M1=M2=6.8. The holding times are 60.0 and the target rotation times are 130.0. All queues are supposed to be empty at the beginning of the simulation. Theory predicts that this network should be stable and indeed it is. The simulation shows that after 10004 rotations we gathered the following statistics:

The time needed for the 10004 rotations was 1041359.00 time units. The statistical profile of the network for the last 9000 rotations is:

mwait	swait	muse	suse	mqueue	squeue	twait	tuse
104.12	1.56	29.75	0.65	21.39	0.53	104.12	29.75
104.12	1.68	15.31	0.53	13.11	0.42	104.12	15.31
104.12	1.58	29.75	0.66	21.38	0.53	104.12	29.75
104.12	1.66	15.31	0.53	13.12	0.43	104.12	15.31

where each row describes a single station and the explanation of each column is given below:

mwait = average observed rotation time.
swait = observed variance between token rotation times.
muse = average observed token usage.
suse = variation of the observed usage.
mqueue = average observed queue length. This length is observed whenever the token enters the station, even if the station cannot use the token.
squeue = variance of the observed queue lengths.
twait = value of mwait as predicted by theory and under the assumption that the system is stable.
tuse = predicted value for muse. Same assumptions as above.

We observe that the rotation as seen by station 1 and the rotation as seen by station 2 cannot have the same distribution because, as the second column shows, they do not have the same variance.

Simulation #2.

The difference between the first and the second example lies in the TRT values which are both equal to 112. In this case T*(1+f[i]) equals 119.43 and while our results cannot predict if the network is stable, our examples imply that it is probably unstable. Indeed, the token takes 1013080 time units to perform 10004 rotations at the end of which there are over 1867 messages in queue 2 and 1926 in queue 4 while queues 1 and 3 are stable. The statistics obtained from the last 9000 simulated rotations are:

mwait	swait	muse	suse	mqueue	squeue	twait	tuse
101.31	11.20	28.95	5.78	20.71	4.14	104.12	29.75
101.31	20.22	14.71	15.91	1040.15	482.86	104.12	15.31
101.31	11.21	28.95	5.77	20.77	4.11	104.12	29.75
101.31	20.20	14.71	15.89	1077.55	500.39	104.12	15.31

Simulation #3.

In the next simulation we raised M1 and M2 to 6.9 for the low priority queues. While our results do not predict either stability or instability, one might expect instability on the basis of our examples. Nevertheless, the network turned out to be stable taking 1009136.00 time units for 10004 rotations. The statistics obtained for the last 9000 rotations were:

mwait	swait	muse	suse	mqueue	squeue	twait	tuse
100.92	10.78	28.83	5.68	20.70	4.06	100.93	28.84
100.92	19.83	14.63	15.33	29.90	9.72	100.93	14.63
100.93	10.91	28.84	5.83	20.71	4.17	100.93	28.84
100.93	20.18	14.63	15.37	28.82	9.19	100.93	14.63

An explanation is in order. If one were to examine the token usage one would find that although the token arrival times are very regular, the token usage oscillates between high and low values. Our example was predicting instability if the token usage were regular and as it turns out, regular arrivals of messages do not guarantee regular token usage. The above results coupled with the explanation offered, suggest that if we were to lower the holding time, then we could force a more regular token usage and thus create an unstable network. Indeed, this is what we do in the next simulation.

Simulation #4.

In this simulation all parameters are as above except for the token holding times that are set equal to 29.0. The network turns out to be unstable taking 986745.00 time units for 10004 rotations.
By the end of the last rotation queue 2 had over 1572 messages and queue 4 had 1558 messages. The statistics accummulated during the last 9000 rotations are given below:

mwait	swait	muse	suse	mqueue	squeue	twait	tuse
98.67	10.63	28.19	1.25	22.42	2.43	100.93	28.84
98.67	12.63	14.14	11.62	875.39	407.08	100.93	14.63
98.67	10.63	28.19	1.24	22.42	2.45	100.93	28.84
98.67	12.63	14.14	11.62	874.97	407.09	100.93	14.63

Simulation #5.

In this simulation w=r=0 and all p's are zero. The holding times

are 60.0 and the TRT's are 112.0. For the low priority queues M1=M2=6.8 while for the high priority queues M1=3.0 and M2=4.2. Thus, we have the same mean arrival values as in simulation #2 but we try to induce high and low token usage times in the high priority queues by varying the arrival rates. The network turns out to be stable taking 1041236.0 time units to complete 10004 rotations.
The statistics accummulated during the last 9000 rotations are as follows:

mwait	swait	muse	suse	mqueue	squeue	twait	tuse
104.13	3.66	29.75	9.93	22.06	7.30	104.13	29.75
104.13	22.92	15.31	15.37	23.22	6.04	104.13	15.31
104.13	7.98	29.75	12.91	22.27	9.55	104.13	29.75
104.13	33.68	15.31	15.32	21.09	5.25	104.13	15.31

Simulation #6.

In simulations that follow, we attempt to gauge the amount of randomness we may introduce by assigning to w and r non zero valus. Thus, the parameters of this simulation are the same as the parameters of the previous simulation except that r=0.20. Queue 4 is stable, but it is not obvious if queue 2 is stable or not (experimentation with r>0.20 - r=0.25,0.30,0.40 - revealed that queue 2 becomes unstable while queue 4 remains stable). In any event, it took 1039875.0 time units for 10004 rotations at the end of which queue #2 had 163 messages and queue #4 17. The accummulated statistics follow:

mwait	swait	muse	suse	mqueue	squeue	twait	tuse
104.01	5.45	29.73	10.19	21.78	7.14	104.13	29.75
104.01	23.36	15.28	15.44	112.67	44.62	104.13	15.31
104.01	8.29	29.71	12.79	22.02	9.14	104.13	29.75
104.01	33.19	15.29	15.34	21.11	5.58	104.13	15.31

Simulation #7

This simulation is identical to simulation #6 except that w=0.02 and r=0.0. Even such a small randomization of the length of the intervals during which the arrival rate is constant, is sufficient to destroy the stability of the network. Indeed, the network takes 1020703.0 time units to complete 10004 token rotations. At the end, queue #2 holds 1454 messages and queue #4 holds 1338. The rest of the statistics follow:

mwait	swait	muse	suse	mqueue	squeue	twait	tuse
101.91	10.07	29.12	8.67	21.03	6.13	104.13	29.75
101.92	23.70	14.83	15.13	353.37	408.81	104.13	15.31
101.91	9.53	29.12	7.90	20.95	5.44	104.13	29.75
101.91	20.87	14.85	15.13	1082.10	620.03	104.13	15.31

What follows is a trace of the queue changes. Whenever a queue' length changes substantially, in this example by 150 messages or more, we print the rotation number, the time this rotation started, and the number of messages in queues #2 and #4. The current queue lengths are then recorded and future queue lengths will be compared to the current (recorded) queue lengths.

rotation	time	Q#2	Q#4
1806	186348.00	39	157
2166	222774.00	190	152
2463	252938.00	343	107
2847	291989.00	306	259
3017	309245.00	214	411
3198	327603.00	121	567
3379	345937.00	26	730
3914	400406.00	36	888
4205	429994.00	189	844
4416	451529.00	345	726
4579	468054.00	502	651
4836	494125.00	436	804
4968	507491.00	330	959
5151	526106.00	237	1112
5344	545688.00	162	1264
5503	561825.00	63	1418
5822	594360.00	0	1574
6234	636059.00	6	1725
7313	747352.00	16	1877
7622	778613.00	168	1845
7804	797123.00	325	1726
7978	814817.00	477	1637
8189	836148.00	629	1591
8318	849274.00	780	1485
8610	878968.00	934	1429
8820	900274.00	1091	1348
9140	932837.00	1017	1504
9432	962512.00	1170	1468
9657	985387.00	1325	1380

The above data suggest that each queue enters periods of stability and periods of instability but that the periods of instability are longer and deeper.

Conclusion: The data from the above simulations support the results that we derived in the main body of this work. They also suggest that if the stability of some low priority queues is conditioned upon the token usage by the high priority queues, then only stringent conditions upon the message arrival mechanism seem to produce stability. Thus, it appears that as a practical matter one should choose TRT[i] to be close to, if not bigger than, (1.0+f[i])*T.

IEEE 802.4 TOKEN BUS EMULATOR

by

Fred Sylvanus and Tuncay Saydam

University of Delaware
Department of Computer and Information Sciences
Newark, DE 19711

ABSTRACT

A performance evaluation facility which emulates Media Access Control (MAC) portion of the IEEE 802.4 "token bus" standards is presented. The facility consists of an emulator that implements the MAC components of the token bus standards, and a representation of the physical layer of the standards as required to logically interconnect the MAC peer entities. The emulator also includes minimal implementations of the Logical Link Control and Network Management facilities as required to generate and monitor network traffic and initialize the emulator. Experiments intended to measure network delay under several network loading scenarios as a function of MAC parameters are suggested.

Introduction

The media access control standards included in the IEEE 802 standards for Local Area Networks (LAN) specify the physical medium and modulation techniques, the corresponding generalized LAN topologies, and the mechanisms for arbitrating access to the network medium. Some examples of physical networks for which Media Access Control (MAC) mechanisms are specified by the standards are physical bus topologies using coaxial cable as the interconnection medium and physical ring topology using a fiber optic or twisted wire-pair medium. Other components of the IEEE 802 standards (e.g. Logical Link Control (LLC) and network management) are independent of the communication medium and access method.

The IEEE 802.4 portion of the standards defines standard mechanisms for a physical bus topology which employs token passing to arbitrate access to the medium, hereafter referred to as a "token bus". This paper describes research to develop software which will emulate MAC portion of the IEEE 802.4 protocol standards. The emulator is a facility being used for the investigation of token bus performance features and to perform tuning of delay as a function of 802.4 protocol parameters. In particular, the components of network delay, as it relates to the priority access mechanisms of IEEE 802.4 standards, will be examined.

Research Goals

The research being undertaken is directed at understanding the control facilities of the IEEE 802.4 protocols, and to be in a position to predict network performance under a variety of loading circumstances. By understand, we mean understanding the behavior of the protocol to the extent that we can appropriately manipulate tuning parameters made available in the standard. By control, we refer to the manipulation of the boundedness of the protocol with respect to

delay, and the relationship of boundedness to network stability as a function of loading. By prediction, we refer to the ability to accurately portray network performance under circumstances that vary with respect to (1) classes of network loading, (2) level of network loading, and (3) the degree of symmetry of network loading. A sub-goal, with respect to predictive capability, is the ability to compare our results with those of others performing related studies.

Network Loading

Local networks can be used to transport heterogeneous mixes of traffic ranging from continuous or consistent flows of data to intermittent or unpredictable traffic loads. The volume of data transported in any of these mixes can be both large or small, and the distribution of this network loading can vary in the degree of symmetry of the loading among the stations participating in the network.

Digital voice requires that delay be minimized to an extent that the reproduction of a human voice at the destination is of acceptable quality. This entails an relatively consistent, uninterrupted flow of information to the destination, but the volume may be relatively small due to voice compression techniques. Digitized video places a similar set of demands on the local network, although the volume of data may be larger, depending on the time and spatial resolution of the video being transported over the network. In both cases, the continuity of the representation at the destination is sensitive to network delay.

Process control applications may be continuous or intermittent depending on the demands associated with various devices on the network (e.g. mechanical controls, sensors). This application can also require strictly bounded network delays, and delivery to be guaranteed by some supporting protocol level. Furthermore, a wide mix of bounded delays may be associated with the various devices being supported.

Network traffic generated by interactive interfaces places unpredictable demands on the network, and the volume of traffic also varies widely depending on the application (e.g. editors versus high resolution graphics). File transfer may also be unpredictable, and involve larger amounts of data transfer, but the associated traffic loading tends to be more consistent for longer periods of time than that of interactive interfaces. Network delay may be a less critical for these applications, but delay is still a concern of the network designer.

Metrics and Parameters

The traffic loading of local networks, mentioned above, represent what we will call classes of network loading (not to be confused with IEEE 802.4 MAC access classes) over which differing delay constraint are imposed. Several features of the 802.4 standards provide parameters that can be used to tune networks transporting various classes of data while potentially meeting corresponding delay requirements.

The metrics that are used to evaluate the impact of these parameters are:

1. The maximum, mean, and variance of the delay incurred by Logical Link Control (LLC) data units are gathered for each of the four IEEE 802.4 MAC access classes. This delay will include both queue waiting time and transmission time.
2. The maximum, mean, and variance in token rotation times are gathered to examine worst case bounds, and to validate the implementation of the

protocol.

Two other metrics that will be available as a result of our research are throughput (measured on a node by node basis rather than with respect to a particular application), and network utilization, which may serve to further validate the implementation.

The existence of various access classes that correspond to (are mapped from) traffic priority levels (i.e. Logical Link Control qualities of service) provide a means of prioritizing classes of network loading. Thus, different classes of traffic as outlined above can be placed in various MAC access classes during experimentation by manipulating the parameter of LLC quality of service.

Within the MAC access classes, several loading schemes such as exhaustive, non-exhaustive, or one-by-one queue depletion may also be examined by manipulating token holding times for each MAC access class. The sensitivity of the token holding time parameters with respect to delay metrics and token rotation times for given loading patterns may also be established.

Scheduled Investigations

There are three experiments being undertaken related to the goals cited above. An experiment investigating delay as a function of two classes of network loading, digital voice mixed with a heavy load similar to file transfer, using exhaustive MAC access queue depletion is scheduled. The results will be compared with corresponding analytical work being done at the University of Delaware [Saydam&Sethi].

A second experiment involves a single loading class (e.g. digital voice) with varying degrees of symmetry in loading.

The third experiment is used to fine tune a given loading scenario and, thereby, evaluate the sensitivity of the delay metrics with respect to changes in protocol parameters used during the tuning.

Emulation as a Evaluation Tool

Performance evaluation of systems in general can be pursued using any of several tools including analysis, simulation, emulation and direct measurement. Our decision to use emulation is based on our desires relative to the following observations pertaining to the development of an emulator of the IEEE 802.4 token bus protocols:

- Due to the potentially low level of implementation, a more detailed representation of the protocol is possible.
- Less abstraction from the actual protocol is required (i.e. a one-to-one mapping of all MAC components of the protocol specification to its representation is possible).
- Due to the level of implementation detail, emulation allows manipulation of "hardware" parameters which possibly affect system performance.
- Detailed software development is required, thereby accommodating goals related to understanding the protocol.

The IEEE 802.4 Token Bus Emulator

Figure 1 depicts the relationship of the emulator implementation components to the usual representation of IEEE 802 protocol layering. The implementation consists of five components that emulate the activity of the MAC portion of 802.4 (IFM, ACM, RxM, TxM, and Timers), two components that handle

the LLC and NMT interfaces to the MAC (REQ and IND), and one component that emulates the physical medium and interface with the MAC (BUS). The emulator does not include the regenerative repeater component of MAC as defined in the standard.

The IFM, ACM, RxM, and TxM emulate their corresponding part of the MAC as specified in the standard. The Timers are interfaced directly to, and driven by, atomic symbols present on the BUS. The REQ component handles the generation of LLC and NMT requests at each station. The IND component fields indications from the MAC at each station, and handles some emulation halting conditions. The BUS component is the controller of the emulator, providing interactive control of the emulator in addition to emulating the physical medium.

The time resolution of the model is to the level of an atomic bus symbol (i.e. data, non-data symbols). This allows a complete implementation of the token bus receive and transmit state machines. The emulator is synchronized by the bus (i.e. each station sees the same atomic symbol each "cycle" of the bus), and the implementation does not emulate the physical characteristics of propagation delays on the transmission medium.

The emulator incorporates the constraint of finite buffer space limitations within the Interface Machine component of the MAC sublayer. Since this constraint is a function of physical implementations of the protocol, it is parametrically alterable. Other parameters are included in the emulator to control the length of a given emulation, and to establish other halting criteria such as the number or types of LLC and NMT indications, or delay thresholds. Finally, parameters that control traffic generation (e.g. mean interarrival times, minimum, maximum and mean packet lengths) are included.

Statistics pertaining to LLC packet delay times, calculated as the time in queue plus transmission time, are maintained in the implementation of the Interface Machine (IFM) portion of the emulator. A separate set of statistics are maintained for each of the four access class queues maintained by the IFM. Likewise, throughput statistics are also maintained in the IFM. Token rotation time statistics are maintained in the Access Control machine (ACM) portion of the emulator. Statistics pertaining to MAC perceived errors are maintained in the Receive Machine (RxM) implementation.

Emulator Implementation

This section consists of details of the actual implementation of the IEEE 802.4 MAC emulation software. Included are a discussion of the sequential/concurrent aspects of the implementation, control and human interface, module partitioning, and some efficiency considerations.

The time resolution of emulation is at the level of atomic bus symbols and the "physical medium" was chosen as the synchronization mechanism. The implementation is organized in a manner that permits event based synchronization, or even message based synchronization between separate instantations of MAC protocols entities. It was decided that the emulator would be a sequential program because of the relative simplicity of such an implementation. A second consideration was the efficiency that is required using this approach, due to the fine time resolution of the emulator.

Figure 2 depicts the relationship of emulator components to the hierarchy of control upon which the implementation is based. The emulator is controlled from a single point, namely the BUS module, which handles interaction with an emulator user, and controls the operation of the emulator. Operation consists of

(1) dispatching and fielding LLC and NMT requests and indications for each station, (2) maintaining the physical state of the medium (i.e. the current bus symbol), and (3) checking for thresholds related to emulator halting criteria.

Each network station, which consists of the MAC emulator components, is an independent instance of those components (i.e. each station contains a private set of MAC objects). Although the emulator is implemented as a single sequential program, stations are implemented in such a way that, by using alternate means of synchronization and data transfer, each station can be implemented as an independent concurrent process. Because of this potential, the generation of requests (by the REQ module) and the fielding of indications (by the IND module) were included in the control portion of the emulator.

For each station there exists an instance of the REQ object, that controls the generation of requests for a corresponding station. As mentioned above, these requests are dispatched, by the BUS module, to the MAC portion (i.e. the interface machine) of the corresponding station at the time they "arrive". There is one instance of the IND module, which handles logging of indications received from each station, and evaluates those indications in light of some of the halting criteria established by a user of the emulator.

The process of controlling the emulation involves data transfer between the control portion of the emulator and each MAC station at each atomic symbol time or bus "cycle". Thus, during each "cycle" of the bus which is emulated, input is provided to each MAC station in the form of (1) NMT or LLC requests for that station, and (2) the current bus symbol. Outputs that are returned from each MAC station includes (1) that station's contribution to (transmission on) the bus during the next bus "cycle", and (2) any indications that may have been generated by the MAC layer in that station.

Emulator Control Components

As mentioned, the BUS module controls the operation of the emulator by dispatching and handling requests and indications, checking for halting conditions, and maintaining the state of the physical medium during emulation. This module also includes the interactive control portion of the emulator, which is described in the next section. From the interactive state, an emulator user can manipulate or display the state of the emulation (including halting criteria), and can initiate or continue the emulation.

When the emulator is running, it performs bus "cycles" until some halting criterion is met. The emulator is synchronized using a round robin polling technique wherein the bus is the synchronizing component of the model. The following steps are taken during each "cycle" of the bus:

1. For each station, requests that arrive from the NMT or LLC layers are passed to the station MAC.
2. Each station is shown the current atomic symbol state of the bus for that cycle. Likewise, each protocol entity is afforded the opportunity to contribute a bus symbol for the next bus cycle.
3. Indications received from each station, if any, are logged in the audit trail.

At the end of the bus cycle, halting thresholds are checked to determine if another "cycle" is to be emulated. If the emulator is halted, control passes back to the interactive mode wherein an emulator user may again manipulate or display the state of the emulation.

The request module (REQ) handles the generation of LLC data units that are submitted to the MAC components (i.e. the interface machine) at each station. As mentioned, there is a separate instance of this object for each station. Each instance maintains independent information about the state of request generation, along with parameters associated with request generation (e.g. mean interarrival time). Interactive manipulation of these parameters is also encapsulated in this module, as described below and depicted in Figure 4. This module also provides interactive support for the generation of network management (NMT) requests, which can also be queued for delivery to the MAC components of any station.

The IND module fields indications from the MAC layer of the stations, and handles the checking of thresholds for halting criteria related to indications. Indications received from stations are logged in the audit trail during the bus "cycle" they are received. Interactive manipulation of halting criteria based on indications received from the stations is also encapsulated in this module.

MAC Station Components

The MAC station components of the emulator are developed according to the corresponding parts of the IEEE 802.4 standards. The emulator includes five modules in the implementation of the MAC layer: (1) the Interface Machine (IFM), (2) the Access Control Machine (ACM), (3) the Receive Machine (RxM), (4) the Transmit Machine (TxM), and (5) a Timers module as part of the ACM.

Figure 3 depicts the general organization of the implementation of the interface Machine. The implementation supports parametrically bounded queues for each of the four MAC access classes, and provides request-to-indication sequencing for LLC requests queued for transmission on the network. It also includes facilities which calculate LLC packet delay statistics on a station by station basis, and tracks exceptions such as the number of LLC request refusals due to exhaustion of interface machine queue space. The interface machine also maintains a queue of MAC indications to be recorded at each bus "cycle" by the IND module described above.

The access control machine is implemented per the standard, and in a manner similar to that used by [Rahimi&Jelatis] with respect to mapping of the protocol specification to the implementation, and the invocation of that code only at "critical events". The implementation also allows interactive manipulation of the state of the ACM, and includes facilities which calculate token rotation time statistics on a station by station basis.

The receive machine is implemented per the standard, including the start-delimiter, end-delimiter, and sil-act detection state machines. The implementation is parameterized with respect to the degree of hysteresis of the silent/active bus detector state machine. Statistics including frequency of occurrence of detected errors are also maintained in the receive machine (e.g. fcs error, non-data symbol detected on non-octet boundary, buffer overflow).

The transmit machine and the timers are also implemented per the standard. Transmit machine state can be interactively viewed and manipulated which provides a simple means of injecting errors into an emulation. Likewise, timer values can be interactively viewed and manipulated.

Interactive Control

The emulator is controlled through an interactive, menu driven human interface. Input to control the emulator can also be introduced from a command

input contained user designated disk files. This facilitates the set up of an emulation and insures consistency when evaluating the effects of parameter changes over several emulations.

Interactively, a user can:

1. View and/or manipulate the state of any MAC component module (e.g. IFM, ACM, RxM) for any station.
2. Enqueue network management (NMT) requests for any station (e.g. gain ring membership, group addresses).
3. Establish and modify parameters controlling the generation of LLC data requests (e.g. mean interarrival time, mean and maximum data unit length) for any station.
4. Establish halting criteria based on (a) the number and types of indications received from stations, (b) thresholds of statistical information maintained in either the stations or the control module, or (3) a count of bus cycles.
5. Inject a transmission error onto the medium by corrupting the value of the current bus symbol for the next bus "cycle".
6. Initiate or continue the emulation.
7. Display statistical information collected in either the stations or the control module.

Figure 4 depicts a sample of menu selections and the menu hierarchy provided to a user of the emulator.

Software Engineering Considerations

The construction of a detailed implementation of the MAC portion of the standard and the supporting emulation environment was accomplished using some commonly accepted software engineering practices. The actual implementation environment is a Digital Equipment Corporation VAX 11/780 running VMS, and the emulator is coded in Pascal which supports separate compilation, encapsulation, and information hiding. Thus, the support necessary for control and data abstraction is used extensively. Each component of the MAC layer is encapsulated, including the interactive manipulation and display of component internals. Similarly, the request generation and indication handling are also encapsulated.

Module partitioning is based on (1) the ability to map some modules directly to/from the corresponding specifications in the IEEE 802.4 standards, and (2) information hiding pertaining to the representation of the states of MAC components and requests generators, etc. The decision to implement a sequential program was based on efficiency considerations, and the use of "critical events" was used to eliminate unnecessary state machine evaluations as in [Rahimi&Jelatis].

Concluding Remarks

At the time of this writing, the current state of the research as outlined is partially complete. The IEEE 802.4 emulator is operational, pending further testing, and the evaluations as detailed above are being performed.

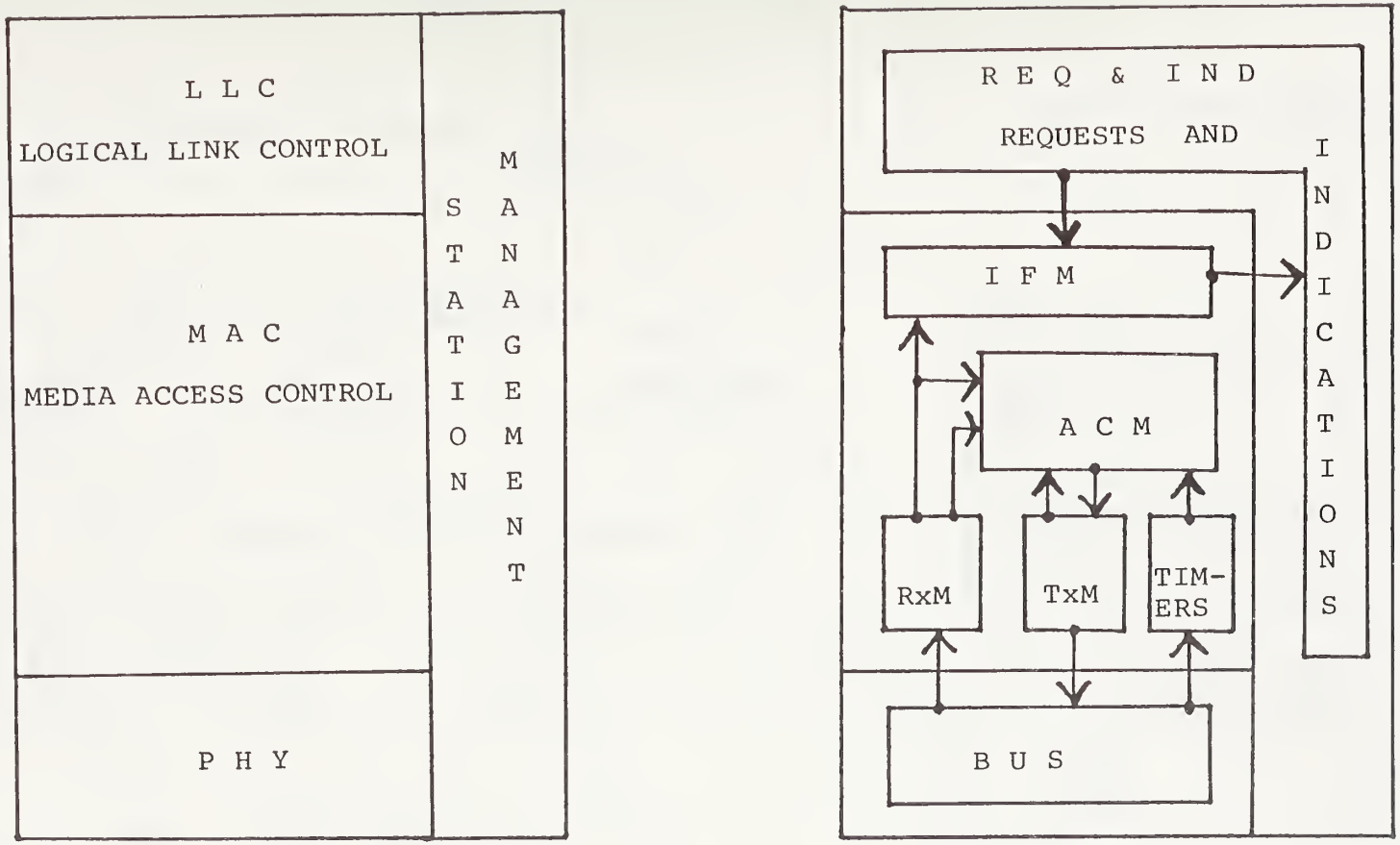
References

[1] Rahimi, S.K. and Jelatis, G.D. "Lan Protocol Validation and Evaluation", IEEE Journal on SAC. Vol 1, No. 5, Nov. 1983.

[2] Phinney, T.L. and Jelatis, G.D. "Error Handling in the IEEE 802 Token Passing Bus LAN", IEEE Journal on SAC. Vol 1, No. 5, Nov. 1983.

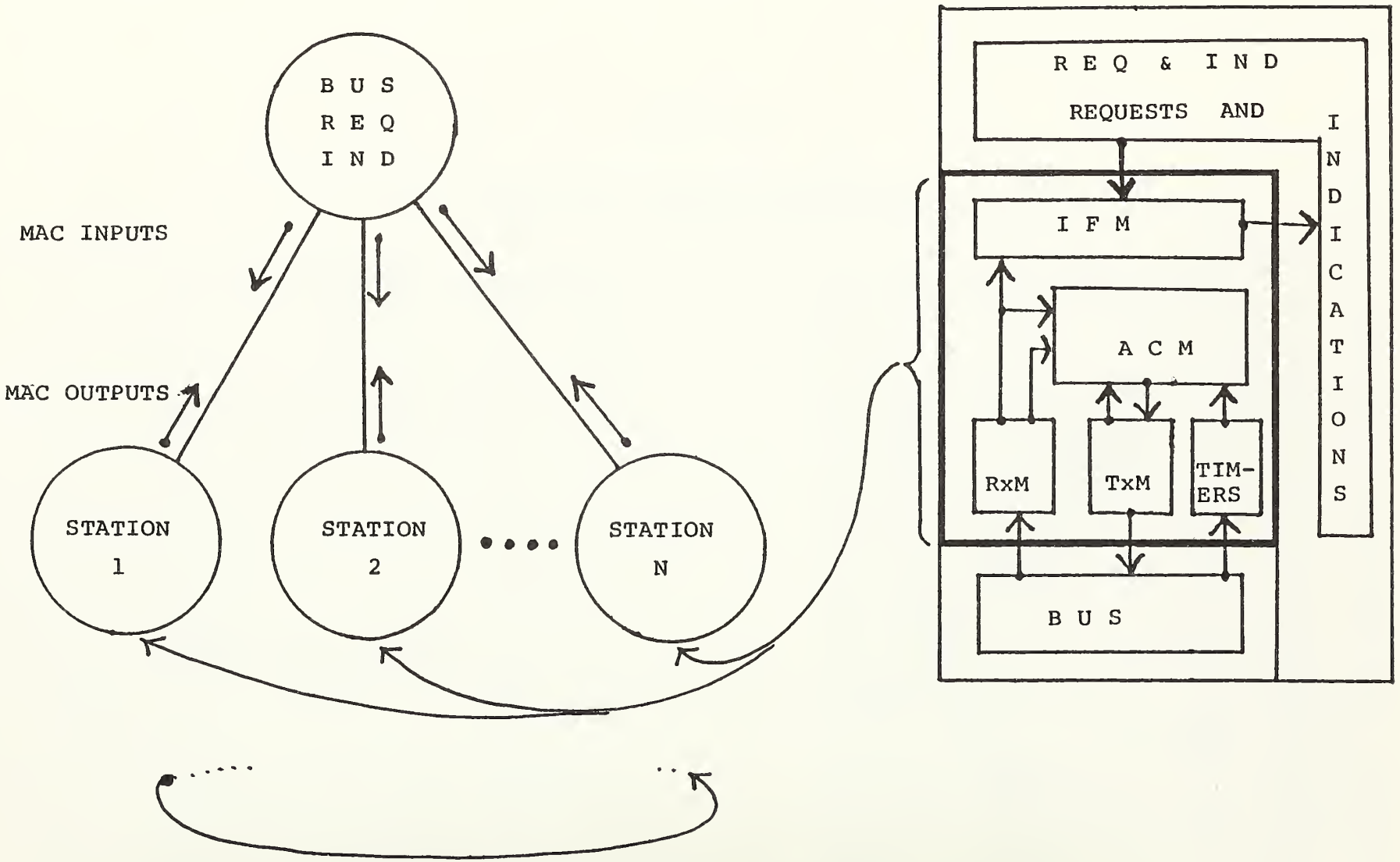
[3] Saydam, T. and Sethi, A.S. "Performance Evaluation of Voice-Data Token Ring LAN's with Random Priorities", Proceedings IEEE InfoCom 85. March, 1985, Washington, D.C.

[4] IEEE Draft Standard 802.4 "Token Passing Bus Access Method and Physical Layer Specification", Dec. 1982



IEEE 802.4 EMULATOR COMPONENTS

Figure 1



EMULATOR IMPLEMENTATION

Figure 2

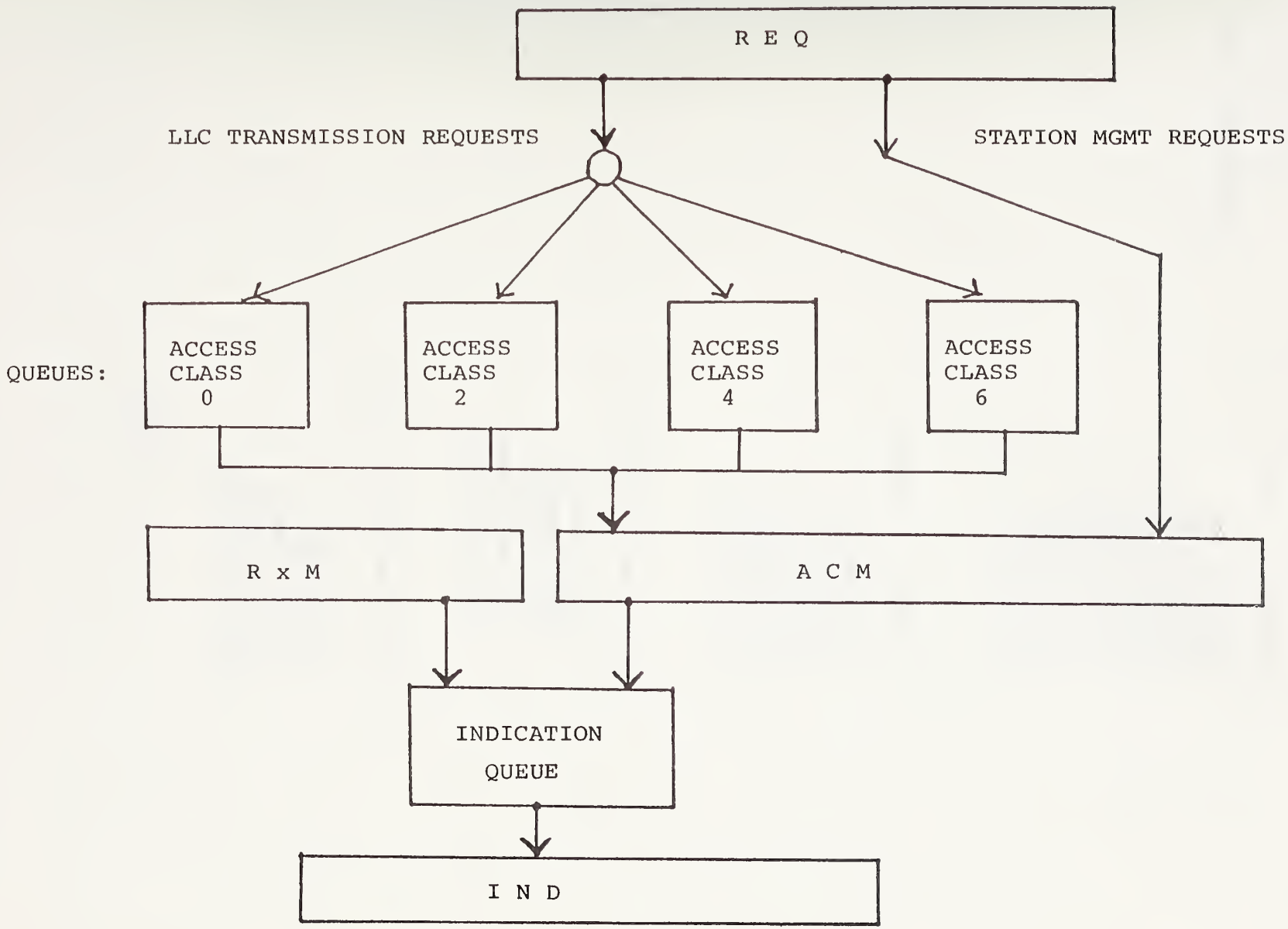

R E Q
LLC TRANSMISSION REQUESTS
STATION MGMT REQUESTS
QUEUES:
ACCESS CLASS 0
ACCESS CLASS 2
ACCESS CLASS 4
ACCESS CLASS 6
R x M
A C M
INDICATION QUEUE
I N D

Figure 3

INTERACTIVE INTERFACE

- Interactive Control Options

  -- flip/flop bus diags
  -- view node states
  -- modify node states
  -- modify halting criteria
  -- add NMT requests
  -- continue emulation
  -- setup from disk file
  -- save node states
  -- corrupt bus signal
  -- stop emulator

- State Modification Selections

  -- modify REQ state
  -- modify TxM state
  -- modify RxM state
  -- modify IFM state
  -- modify ACM state
  -- modify TIMER limits

- Request Generator (REQ) Options

  -- LLC Request Generation
    -- flip/flop diagnostics
    -- min packet length
    -- mean data unit length
    -- mean interarrival time

- NMT Request Generation Options

  -- initialize protocol
  -- set timer limits
  -- set group addresses
  -- gain ring membership
  -- leave ring membership

Figure 4

Discussion Group 1

Factory Automation

Moderator: Gary Workman

NOTES FROM THE FACTORY AUTOMATION APPLICATIONS SESSION

Bureau of Standards - 4/30/85

Attendees: Gary C. Workman, General Motors, Moderator
Anatoly Moldovansky, A/B
Andy Luque, Tektronix
Joseph B. Rickert, Jr., Sytek
Sharon Heatley, NBS
Bruce A. Loyer, Motorola
Detler Leisengang, Siemens
Pete Malpass, Contel, Note-taker

Speaker denoted by two initials:

GM: Let's begin by asking what questions we want to answer.
BL: I'd like to discuss what sorts of network loadings and applications we can expect. Are they huge factories, distributed systems?
AM: What are the workloads? Are there differences in the timeliness of the information required, especially for factory versus office automation?
DL: I'm interested in differences in modems. Are they different than for offices? Are any better for factories since they are more resistent to dust and nasty working conditions?
AM: Another thing is that traffic is typically not characterized by a Poisson distribution of arrivals. It is more often predictable or at least deterministic. Uploads and downloads of information are non-standard.
GW: Another question is the distance that one can generalize from a simulation.
AM: We should think anout how a predictable arrival rate affects token bus.
AL: It's clear that peak loads lend themselves to abnormal conditions. A control system in a factory is not as concerned about maintaining a high utilization, but rather in minimizing an end-to-end response time. We expect to find at most a 20% utilization, but efficient use of the cable plant is not required.
GW: The concern with ETE response is in the realm of connections. A connectionless service is all that has been discussed so far in these (meeting) proceedings. Buffer management, communications interfaces, etc. are all involved with ETE questions.
AL: Yes, and implementation issues above LLC are what's important to the customer, (not just the media access)...
GW: Let's try to divide the issues into 3-4 areas.
AM: There are just two: MAC layer and ETE with MAC layer.
GW: We've found that the excessive overhead is in ISO layer 4 over the LLC. In many applications you don't need the control, but need more timely responses. We are investigating LLC3, trying to get an ack within one token holding time. This drops an orderof magnitude off the ISO4 mechanisms. LLC 1C services are inadequate.
AL: Upper level protocol (HLP) requirements tend to drive which of LLC 1 or LLC 3 get responsibility.
JR: Doesn't the number of nodes also contribute to the delay?
GW: Yes, you must have fewer nodes under token systems: less than 15/channel for real time control and reasonable response times. If, however, one only needs to do monitoring, status and management, then each channel can support over 100 nodes.

JR: Do you need separate cables for single point of failure protection?
GW: That also leads to questions of geographical dispersion of devices.
AL: Aren't they normally bounded areas, say 100 feet square?
GW: In rolling mills, processes can extend over 200 feet and require 30 ms or so response times almost surely. That is, ETE message delay and ack. Thi can be done in LLC3 or lower- ISO4 can't make it.
SH: Do you mean that an application sends a request or notice and gets "acked" in response?
GW: Getting the message delivered on the receiving end is an interface issue there - not communications.
AL: A guy won't take 29 ms and give you 1 ms. Allocation is sharply divided. What is the traffic in this environment? Deterministic?
GW: We hope it's pretty deterministic. Programs relate to each other and they expect certain behavior. It has to be a bounded problem.
AL: Under normal operations, sure, but what's build-able? If we can't meet 30 ms, then redesign communications architecture to make it a different problem which will handle both rotuine and emergency operations.
GW: It would be nice to have both 14 and lower level transmission: I4 allows message segmentation for numbers, etc., and LLC3 handles control messages.
AL: Only two selections: class 3 and ISO 4. Have you done any response time trade-offs?
GW: class 3 is faster, ISO 4 has more services. You actually want both. Time critical must be available, but some applications will need the combined services.
AL: How are you going to assign priority when two or more messages arrive simultaneously?
GW: Is there any concensus? Do we discuss separate or both from here out?
BL: Does every LAN need both?
JR: What's the penalty for losing the services with just class 3? Are you ever sorry for not having ISO 4?
GW: Yes. When you go through a gateway the penalty is like direct dial versus a rural operator. User applications requiring routing generally use the communications services to achieve it. This should not be in the programmer of applications realm of responsibility. Also, segmentation of large messages also requires ISO4.
AL: For geographically distributed applications, does timely service only imply timely for "inside" users? Outside routing should cost more. It may not be so time critical for those users either.
GW: Uploads and downloads sort of need ISO4. Once completed, however, maybe a device can choose what services it needs flexibly.
SH: Would you send transport to all devices?
GW: Let's not be biased by MAP meetings.
PM: You could put multiple boards for multiple services on multiple channels in the same communications interface to a device.
GW: At twice the cost...
AM: As I understand it, you can "Steal Bandwidth" - keep offered load low to assure timely delivery. This is done by keeping token overhead low, say 5%.
AL: You could also use the four priority levels to play a game with arrival times.
GW: That's where the issues of fairness and starvation come into play.
BL: Weren't the examples (in the earlier performance session) paranoid cases for the high priority examples (where starvation occurred)?
GW: In the example (Nakassis) you get 40 tokens, 1 get 5, and he gets 10. It's fair in that at steady state you get far more than equal bandwidth. You

have to be careful in judging examples.
...
GW: Don't worry about the add/delete of stations from the bus: it's very rare.
AL: Are there other constraints to bound the problem of the number of nodes?
SH: Such as more than two trying to talk at once?
GW: Each device knows when he talks. Master/slave relationships are the norm today, but not in the future.
SH: But is it realistic, the case of two nodes talking within 30 ms?
AM: When you talk about two nodes, consider the ETE: most delays are CPU control system. We can conclude that 2 stations have probability p of simultaneous arrivals, and it is not large.
SH: Do robots talk? In high speed machinery, how do you pull out communications from the processor machine, or at least get them coordinated?
AM: Robots have a warning of imminent activity. Device must ack/nak readiness. When a component is against a spindle with too much force, (the normal pickup would be post-production inspection) revealing a deviation, so tool settings would be modified (somewhat later) at manufacturing.
AL: Is there a primary device?
AM: Yes, usually a cell controller, but communications are required far beyond that.
GW/AM: Interactions are point to point today. Braodcast is the most sophisticated.
AL: Is there parity within the cell?
AM: Peer to peer is approaching. It will be accomplished at the task level and the device will execute independently with independent communications.
GW: We are moving to an hierarchy of control with distribution of communications capability.
AL: For example, you have a PC on your desk, but it wasn't economical to equip you with an 800 MByte disk, so you hook up to the mainframe down the hall. You communicate more to the central server than to peer users.
GW: You will get the disk later, however, then you will change to peer-peer communications as principle comm activity. Work expands to use resources. We are moving to distributed environments.
AL": We moved more up and down our branch of the hierarchy before. In organizations, the hierarchy is still there long after its usefulness was reduced. Will the same thing happen in factory automation?
GW: Doubtful.
BL: Not every node will be able to talk to each other. Applications in several places can communicate to the same node. If the node reports off-net status...
GW: You will get an ack
BL: It will use ISO4 and be transparent to the device. The network layer can provide it.
AL: All this discussion has been with respect to traffic control within the segment. With devices that only recognize networt layer, what if there is no operator (Ref: previous allusion to direct dial vs. operator assist).
JR: Connection relays and transport layer will have to be at the device. That's the way it will know what to do.
AL: ISO4 filters the MAC traffic.
GW: Control traffic only consumes a small %.
SH: What about downloading new programs?
GW: Now, the cell will be off while doing downloads. Later it will be possible to send to reserve buffers while the device is working.

SH: At the end of the shift?
GW: No. The cell controller has workstation capabilities (intelligence). If tools break, etc. the cell can be reprogrammed to work around or do alternate work.
SH: Can it be stored then reprogrammed?
AL: Really it's a change of task.
AM: Machine tools run 24 hours in robot factories such as the Japanese.
AL: For non-time critical applications, can you go outside segmented comm?
GW: A master system is available for all cells, so ask/receive works for all units, but there are varying needs.
AL: A lot of semi-intelligent that need a spoon? Feed quick before it starts crying. You still have response times.
GW: You have to look at the capacity of the overall network, which is some function of t-1's and t-2's. What is the upper bound on the segment? it is a function of system comprehensiveness.
AM: If the need for info comes from shop-wide, then you need area-wide sources. You can do 3-4 area nets broadband, then one or so per local. Broadband is required for high connectivity as is hierarchical control.
GW: Areas allow for geographically distributed cells with interconnections.
AL: You need to manage in an integrated fashion. What's the overall level, etc.?
GW: Network management is a much abused term. (1) application management is outside comm.,(2) comm also needs management, (3) configuration and interfaces will need management. LAN management? How do you combine into an NCC (network control center), when that becomes a single point of failure?
AL: A key issue is whether to build up structure. Do you have an overall control or do a distributewd, segment-wide control?
GW: Actually, it evolves from segments with separate controls. You learn how to manage a segment and hopefully the techniques will generalize to a cell of segments.
SH: Do you have a feel for times?
GW: For some devices, you have a 6 mile long reel of instructions.
AM: You can't download all of it. Do chunks to memory. When you want the next chuink, it is fast: you have to emulate a reader now, but when you give the device local storage...
AL: Downloads will become a regular activity.
GW: Tools are single channel now. It's tough to control the big picture.
SH: What about cell centralized storage?
GW: Right now the environment is too nasty for current mass storage devices.
AL: Also, you can buy 256KB chips for $3, but a 20MB hard disk is $500. There is more capability when more requirements demand it. It's a never ending spiral: having more leads to wanting even more.
AM: I suggest we identify key areas for more research: Problems with factory networks.
GW: So far we have:

(1) Performance of transport over LLC 1 (SH: NBS is doing as task)
(2) Extension of MAC to include LLC 3 services (intent is to minimize response times)

DL:(3) Baseband vs. Broadband for control segment. (Criteria may be different, as may the management strategy, even though the MAP strategy is the same - European need is more for baseband: US has more TV cable. For the distances and hierarchy of control, baseband is cheaper. Only 15-20 nodes on the factory floor = baseband with perhaps broadband between floors. Then costs can be separated too)

(4) Are there limitations on 802.4 because of factory automation?
    (a) Transport (a lot of traffic but infrequently) versus
    (b) control (short, predictable, often, a few stations.
    Utilization is kept small by upping data rate. All stuff is based on traffic.
(5) Need traffic analysis for control applications. This is the starter before you vary the other parameters.
(6) What's available? - modems/baseband speeds especially. Currently, Allen-Bradley & GM are seeing 20% utilization about max: what is the throughput at 20% utilization? We can continue to simulate, but we really need an idea - can GM & AB provide "requirements"? - yes but very application-specific. Would like to know how fast, how much for both segment and cell.
(7) Throughput for multiple segments.
    (a) One station between segments leads to network management issues: How much and where?
    (b) bridges and relays and entire net management - can you fine tune to optimize for this capital investment?
    (c) simulate real-time net control too. Is it better to block transmit or fragment long messages?
    (d) For small station counts (15-20) what happens when you divide the load into two nets? What is the impact? - Issue is the delay from going over a bridge or relay.

(At this point we adjourned).

Discussion Group 2

Network Performance and Management

Moderator: Dan Stokesberry

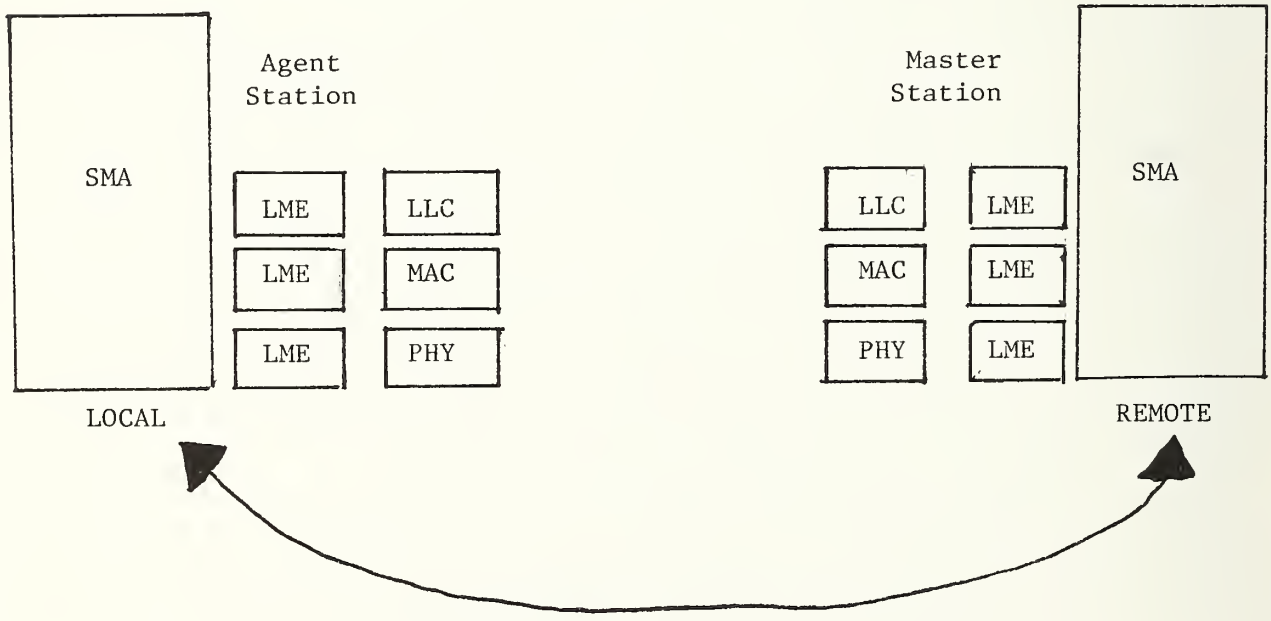
DISCUSSION NOTES OF PERFORMANCE AND NETWORK MANAGEMENT

Three issues were identified at this discussion meeting attended by seventeen (17) participants from ten (10) organizations. (The attendance list is attached.) The three issues were the network management, performance issues and load attributes.

Jade Chien of INI presented the goals, architecture and functions of network management for the physical and link layers based on the current work of IEEE 802.1.

The goals are:
1) start the network
2) make sure it works well
3) correct mistakes.

The architecture of LAN management is depicted below. It consists of local and remote management capabilities. The Local Management Entity (LME) of an agent station reports to the LME of the master station based on events. The master station may set parameters at agent stations based on reports from agent stations and management requirements such as real time response, number of stations and others.



Network Management functions are listed below:

1) to configure
2) to down load
3) to monitor
4) to change parameters based on performance of the network.
5) to respond to events

Jade believes that the performance issues are under the umbrella of network management. In other words, the performance data is mainly used for allowing the performance of management functions.

Karen Hsing of NBS thinks that the effect of management functions on overall protocol performance ought to be of interest to users.

For performance issues, Steven Dimford of Comm Power proposed a set of input and output variable for all simulations of 802.4 specifications. He is interested in comparing the simulation results based on a baseline set of input and environment variables. This idea was well received. Following is a list of the proposed input and output variables.

INPUTS

O/ station 5, 50, 100, 200, 300, 400
Bandwidth 10Mb
Data length 64, 128, 256, 512, 1024, 2048, 4096, 8192
Number of active stations 10, 50. 100, 200, 300, 400
Preamble length 32 bits
Address size 2 or 6 bits
Address allocation Random, Cyclic, Hand
Frame arrival type P, C
Frame arrival rate
Station delay (latency)
Slot time
Token hold time
Max_inter_solict_count
Queue depletion scheme (1, 2, 4, 6, all)
cable length
Amplifier delay
Headend delay
Buffer size
Ring Management
Static or dynamic

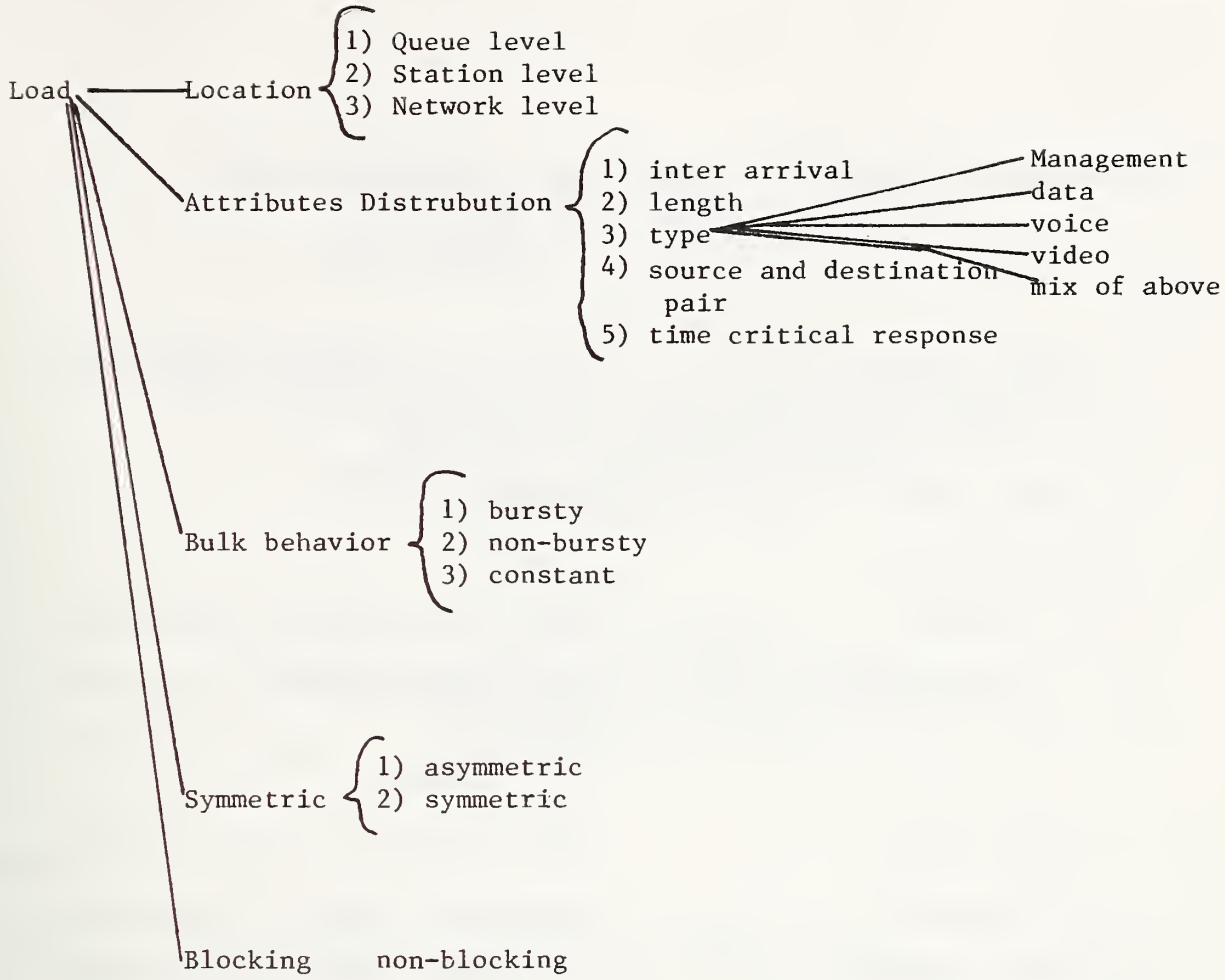
Physical location of station
Criteria for stopping
Error rate of channel
Failure rate of station

OUTPUTS

Worst
Mean
Var
S.D.
Token rotation time
Queue length
Queuing delay
Run of simulation
Difference between two equal time runs
Throughput
Bus utilization
Number of active stations
Priority levels
Stability definition

Steve offered to provide additional inputs to the proceeding entitled, "Terminology Dictionary and Baseline Varibales for IEEE 802.4 Token Bus LAN Simulation."

For load attributes, Professor Tuncay Saydam of The University of Dela presented his view of load as follows.


Load
Location
1) Queue level
2) Station level
3) Network level
Attributes Distrubution
1) inter arrival
2) length
3) type
4) source and destination pair
5) time critical response
Management
data
voice
video
mix of above
Bulk behavior
1) bursty
2) non-bursty
3) constant
Symmetric
1) asymmetric
2) symmetric
Blocking
non-blocking

PERFORMANCE AND NETWORK MANAGEMENT ATTENDANCE LIST

NAME	AFFILIATION
1. Robert Toense	National Bureau of Standards
2. Stephen Dunford	Comm Power
3. Chuck Croft	SYSCON Corp.
4. Tom McGowan	Digital
5. John Meyer	ITI, University of Michigan
6. Curtis Royster	National Bureau of Standards
7. Fred Sylvanus	Autotoe Ltd., and University of Delaware
8. Karen Hsing	National Bureau of Standards
9. Jeff Bader	National Bureau of Standards
10. Steve Ritzman	National Bureau of Standards
11. Tony Capel	SECL, Chalk River
12. Charles Hartmann	National Bureau of Standards
13. Jade Y. Chen	Industrial Networking/ Ungermann-Bass
14. Wayen A. Mack	Vance Systems
15. Tuncay Saydam	University of Delaware
16. Dan Stokesberry	National Bureau of Standards
17. Rob Rosenthal	National Bureau of Standards

TERMINOLOGY DICTIONARY AND BASELINE VARIABLES FOR IEEE802.4 TOKEN BUS LAN SIMULATION

prepared for

THE PERFORMANCE AND NETWORK MANAGEMENT COMMITTEE
OF THE WORKSHOP ON ANALYTICAL MODELING
OF IEEE 802.4 TOKEN BUS -
SESSION III: PERFORMANCE ISSUES
APRIL 29-30, 1985

by

STEPHEN DUNFORD
COMMUNICATIONS & POWER ENGINEERING, INC.
26560 AGOURA ROAD #101
CALABASAS, CA 91302

ABSTRACT

This working paper presents a first draft of a terminology dictionary and a set of baseline variables to be used in simulation modeling of IEEE 802.4 Token Bus so as to create a basis for comparison in future workshops. It will be refined and expanded in the future. Any suggestions and criticisms should be addressed to Stephen Dunford at the above address.

OVERVIEW

As a part of the third session of the NBS workshop on IEEE 802.4 modeling, it was found that there was a need for a terminology dictionary and a set of baseline variables for modelers of the IEEE 802.4 token bus. The need for the data dictionary derives from different interpretations of the vocabulary used in IEEE 802.4 simulation and analysis.

The baseline set of variables is a standard set of inputs for IEEE 802.4 simulations and models so as to provide a means of comparing simulation results. It also identifies a standard set of outputs to be produced. In the future, this standard set of inputs and outputs should be included in papers presented to NBS workshops.

TERMINOLOGY DICTIONARY

THT6 - Priority 6 token hold time

THT4 - Priority 4 token hold time

THT2 - Priority 2 token hold time

THT0 - Priority 0 token hold time

TRT - Token rotation time

Queuing length

- Time from data request to time data actually sent

Simulation time

- The number of seconds of simulated times of the system

Number of stations

- The number of stations that have been granted ring entrance after the logical ring was initialized.

Acquisition Delay

- The acquisition delay measured from the time when a frame arrives at the transmit queue until the first bit is transmitted onto the wire.

Latency

- The average delay in sending a data frame, which includes the queuing delay, the token delay, the station delay, and the transmit delay.

Total Offered Load

- The total number of data bits generated by all the active stations per second (expressed in terms of percentage of channel bandwidth).

Network Data Throughput

- The total number of data bits received at the destinations per second (expressed in terms of percentage of channel bandwidth).

BASELINE SET OF VARIABLES

Inputs

The following are some of the baseline variables to be used in simulations of 802.4, so as to provide a means of comparison of the different models:

Number of stations - 5, 50, 100, 200, 300, 400
Bandwidth
Data length
Number of active stations
Preamble length - 32 bits
Address size - 2 or 6 octets
Address allocation - Hand, Random, Cyclic
Frame arrival type - Constant, Poisson
Frame arrival rate
Station delay
Slot time
Token hold time (for all the different priority queues)
Max_inter_solicit_count
Queue depletion scheme (for all the different priority queues)
Cable length
Amplifier delay
Head end delay
Buffer sizes (for all the different priority queues)
Physical location of stations
Error rate of channel
Failure rate of stations
Criteria for stopping

Outputs

The following information are the standard outputs for an 802.4 simulation using the standard inputs:

Worst case
Mean
Var
S.D.

of

Token Hold Time (for all the different priority queues)
Token rotation timer
Queue lengths (for all the different priority queues)
Queue delays (for all the different priority queues)
Elapsed run of simulation
Machine run time of simulation
Throughput
Bus utilization
Number of active stations

SUMMARY AND CONCLUSIONS

This working paper is a starting point for a common set of terminology and baseline variables to be used in 802.4 simulation. To continue this effort, the author needs input from IEEE 802.4 simulators on the following:

- Terminology
- Types of input/output variables
- Values for the input/output variables
- General criticism

Please mail any remarks to:

STEPHEN DUNFORD
COMMUNICATIONS & POWER ENGINEERING, INC.
26560 AGOURA ROAD #101
CALABASAS, CA 91302

Special thanks must go to Jade Y. Chien of Ungermann-Bass for the use of her paper on "Performance Analysis of the 802.4 Token Bus Media Access Control Protocol."

Discussion Group 3

Compliance Testing

K. H. Muralidhar

Minutes of Special Interest Group Meeting on Conformance Testing

Moderator:
K.H. Muralidhar, Industrial Technology Institute
Members:
David J. Greenstein, GM Technical Center
Heinz Jauch, Western Digital Corporation
Orly Kremien, Motorola Semiconductor Israel Limited
Joseph P. Brazy, Syscon Corporation

In this meeting, three main aspects of conformance testing of IEEE 802.4 protocol were discussed. The aspects discussed were, test architecture, test structure, and types of testing. A brief description of each of the abovementioned items is given below.

Test Architecture: In order to take care of synchronization problems and level of integration of protocols into chips, it was decided to have an application layer interface for testing purposes. This would eliminate the requirement on vendors to provide a separate interface at each layer for testing purpose. In addition, it was thought by providing this interface, design of test management protocol would become easier. However, no attempt was made to specify this application layer interface.

Test Structure: Main topics that were discussed here were, set of scenarios, testing methodology, instrumentation, parameters, and topology.

Set of Scenarios: Two sets of scenarios were identified for conformance testing and they are mandatory test scenarios and optional test scenarios. The mandatory test scenarios must be capable of testing normal and some abnormal behavior of implementation under test. Tests for those abnormal events whose probability of occurence is very low can be included in optional tests set. Main idea in this was to make the set of scenarios reasonable.

Testing Methodology:
Not much was discussed on this topic. Encoder/Decoder approach and Reference Implementation approach were identified as possible methodologies for testing.

Instrumentation: A reasonable set of instruments to analyze physical specifications, group delays, and protocol behavior was the main idea in this topic. No attempt to list instruments that can be used was made.

Parameters: Timers that need to be set, preamble length, and tolerance of signal levels were identified as critical parameters for testing. It was decided to have a capability to adjust these parameters using application layer interface described earlier. Also, it was decided to have a range of values for each parameter as opposed to a value. Further, tolerances in these parameters were also thought to be specified.

Topology: It was decided to have some sort of cable simulator to test implementations in a real world fashion. This should provide a capability to realize distances that are used between stations (to capture the implementation's behavior to frequency drifts, amplitude drifts, group delays etc.) and large number of stations in the network.

Types of Testing: Three different types of testing were identified and they were, acceptance testing, protocol testing, and multi-vendor testing.

Acceptance Testing:

In this type of testing, an implementation was tested against mandatory requirements of a standard to check whether implementation complies to it or not. A GO/NO-GO type of result was expected out of this testing.

Protocol Testing: This testing helps validating a protocol for its behavior. This was aimed at correcting protocol standards to eliminate bugs in protocol. Actual methodology for protocol validation was not discussed.

Multi-Vendor Testing:

Depending on the requirements from customer/vendor, multi-vendor testing of an implementation need to be conducted to ensure interoperability. Details regarding testing of an implementation with which other implementations were not discussed. However, operability of an implementation in an multi-vendor environment only does not imply conformance.

Discussion Group 4

Simulation

Moderator: Juan Pimental

Simulation Subgroup Summary 30APR85

The Simulation subgroup was moderated by J. Pimental(GMI) and attended by about 15 workshop participants. Roughly 1/2 the time was used to outline the main areas of concern for the subgroup, with some discussion mixed in. Following that, each of the points under 1 and 2 below were discussed (time ran out before areas 3 and 4 were reached). Some consensus was reached in the discussion, and the results are summarized on the following pages.

NOTES FOR THE READER: The short time available didn't permit any recommendations by the group, so the following should be taken as information only, not a plan of action at this stage. The last section, "Editorial Notes", includes a few things that weren't properly a part of the subgroup discussion but seem wise to mention in this context.

OUTLINE

1.0 SIMULATION METHODOLOGY
1.1 Intended Simulation User
1.1.1 Protocol Designer
1.1.2 System Designer
1.1.3 Network Designer
1.2 Application Oriented Vs. Procedural Language
1.3 Output Analysis
1.4 Automatic Setup Of Modelling Blocks
2.0 TRAFFIC CHARACTERIZATION
2.1 Number And Type Of Station
2.2 Arrival Rates
2.3 Scope: Factory Automation
2.4 Higher Layer Protocol Effect
2.5 Burst / Peak Traffic
3.0 MODEL COMPARISON
3.1 Common Set Of Inputs
3.2 Common Set Of Outputs
3.3 User Interface
3.4 Validation Against Experiments
4.0 REQUIREMENTS FOR SIMULATION
4.1 Scope Of Simulation
4.2 Parameters
4.3 Controls
5.0 EDITORIAL NOTES
5.1 Standard Terms And Definitions
5.2 Area Of Concern Suggestions
5.3 Subgroup Coordination Needs

1.0 SIMULATION METHODOLOGY
This area includes the structure of and design tools for the simulation itself, separate from the user interface and I/O of a simulator. The following areas were discussed as important determining factors.

1.1 Intended Simulation User
In the discussion, the user of the simulation was emphasized as the main factor. Several user categories were identified - note that one user might be in two (or all?) categories.

1.1.1 Protocol Designer - This is the person who attends IEEE 802 meetings, is interested in a simulation that allows changing the state machine, and wants a simulator that talks the language of the IEEE Standard.

1.1.2 System Designer - Included here are the vendors of 802.4 compatible products, including (but not limited to) turnkey systems, software, boards, and IC's. They want flexible interfaces to other simulators or emulators, few changes to the state machine, and a highly modular simulator (to allow "external" interfaces at many possible points).

1.1.3 Network Designer - Under this category are implementors of a working network, most likely with expertise in the application that the network is used in, but not the network itself. They need to test choices for the network topology and parameters, decide allocation of resources, and interface the simulation to analytical models (we envision the use of analysis when simulation is too slow or investigates too few cases).

1.2 Application Oriented Vs. Procedural Language
This choice is driven on one side by strictly practical considerations (cost, target machine available, existing expertise) and on the other side by the ideal fit for the user. Of the three user groups of the last section, protocol designers might use both language types, system designers mostly procedural languages, and network designers mostly application oriented languages.

1.3 Output Analysis
This area includes statistical analysis of the simulation results and determining confidence intervals (no single method is recommended here).

1.4 Automatic Setup Of Modelling Blocks
This includes automatic (at least as far as possible) translation of protocol specifications and system specifications into the model without hand coding. Issues mentioned were: need for many or fast changes to model (i.e. for system designer), need for high confidence in fidelity of model, and use of ESTEL (sp?) for protocol specification.

2.0 TRAFFIC CHARACTERIZATION

The next main area of concern discussed was getting a realistic traffic model. This centered on abstract vs. detailed methods, and what the target application should be.

2.1 Number And Type Of Station

It was agreed that number of stations (both in ring and transmitting) and number of ports per station should be variable if realistic traffic levels are to be simulated. No specific limits for these numbers were recommended.

2.2 Arrival Rates

A main point here was that the simulation should allow very realistic arrival rates, as contrasted to smooth continuous distributions. A second point brought up was whether actual logs (machine-readable) of network traffic should be usable as input to the simulation.

2.3 Scope: Factory Automation

The group consensus was that the target application examined for traffic data would be factory automation.

2.4 Higher Layer Protocol Effect

The higher layers in the OSI model each generate some network traffic in executing their protocols, and the effect this has on traffic levels at the upper MAC interface should be included in traffic characterization.

2.5 Burst / Peak Traffic

Several comments were made concerning the discontinuous nature of arrival rate distributions in factory and voice applications, for example when a NC machine must be programmed, it requires a much higher data rate through the network than during normal operation and status reporting.

3.0 MODEL COMPARISON

This area is important for synthesis of results from different researchers in the field, and it was apparent from comments that many attendees saw difficulty in comparing assumptions and results presented at this workshop.

3.1 Common Set Of Inputs

A minimum set of conditions and variables used by all simulations.

3.2 Common Set Of Outputs

A minimum set of metrics and statistics (not including presentation).

3.3 User Interface

The presentation of results to the user and the setup of the simulation by the user.

3.4 Validation Against Experiments
The inputs and outputs of the simulation should allow testing of the simulation against experimental results.

4.0 REQUIREMENTS FOR SIMULATION
This area overlaps methodology a little (the ends get mixed with the means!). A good way to distinguish them is: requirements looks at the simulator as a black box of sorts, while methodology jumps right into that box.

4.1 Scope Of Simulation
For instance, how much of the physical layer do we include in the simulation? For some users, a simple delay might work, while others may want much more detail. An issue brought up in Traffic Characterization was how much higher layer effect to include.

4.2 Parameters
These are the variables that the simulation user can "tweak" to investigate a response in the outputs of the simulation.

4.3 Controls
These have to do with the setup, flow and termination of the simulation.

5.0 EDITORIAL NOTES
An appeal: this is just a start, as even the outline above is not a full one. If you have any areas to add to it, or think an existing one doesn't belong, don't hesitate to contribute your ideas. It would be best to have some inputs before the next workshop, since time may be limited again there. I'll leave the method up to the workshop organizers (probably mail direct to other participants or the subgroup moderators), best if there's some way to get one round of comment before the next workshop.

The following are a few comments on areas that bear on this subgroup.

5.1 Standard Terms And Definitions
A great need exists for common language in simulation and in 802.4 modelling as a whole. A facet of this is mentioned above in 3.1 and 3.2. Steve Dunford of Commpower is working on a contribution in this area, and anyone else who can expand the list is encouraged to contribute. As well, some survey of the literature in the area of 802.4 modelling should be done to avoid re-inventing the wheel.

5.2 Area Of Concern Suggestions
Everyone is encouraged to contribute suggestions for areas of concern: an example might be to have as a goal some specific recommendations for tools or a standard methodology, as well as conducting evaluations of the same.

5.3 Subgroup Coordination Needs

There are many common concerns with other subgroups (traffic characterization, for instance) and a coordinated effort is needed to avoid duplicating work. A first area for this coordination may be in writing a tutorial and glossary for 802.4 modelling, especially to smooth the way for newcomers to the field (and let's hope there will be many!).

U.S. GOVERNMENT PRINTING OFFICE: 1985-461-105/20161

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBS/SP-500/127	2. Performing Organ. Report No.	3. Publication Date June 1985

4. TITLE AND SUBTITLE

Computer Science and Technology:
Workshop on Analytic and Simulation Modeling of IEEE 802.4 Token Bus Local Area Networks

5. AUTHOR(S)

Robert Rosenthal, editor

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
U.S. DEPARTMENT OF COMMERCE
GAITHERSBURG, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered

Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

Same as item 6

10. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number: 85-600556

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

Token bus Local area networking technology is anticipated for use by national and international organizations seeking standard solutions for process control and Laboratory and factory automation applications. Several token passing technologies have been described; but, only one emerging standard, the IEEE 802.4 Token Bus currently includes broadband communications utilizing a prioritized, robust and deterministic access method. These workshop proceedings report the deliberations of 39 participants from industry, academia, and the Federal Government who came to NBS to 1) encourage modeling of 802.4, 2) to build competence and confidence in 802.4 technology, 3) to provide public knowledge about the behavior, characteristics and performance of 802.4 and to highlight areas for further study on the NBS 802.4 test bed facility.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

Analytic modeling; discret event simulation; emulation; Local Area Networks; performance; simulation modeling; standards; token bus; testing.

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☒ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

268

15. Price

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SCIENCE & TECHNOLOGY

Superintendent of Documents,
Government Printing Office,
Washington, DC 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Bureau of Standards Special Publication 500-.

Name _____

Company _____

Address _____

City _____ State _____ Zip Code _____

(Notification key N-503)

NBS Technical Publications

Periodical

Journal of Research—The Journal of Research of the National Bureau of Standards reports NBS research and development in those disciplines of the physical and engineering sciences in which the Bureau is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Bureau's technical and scientific programs. Issued six times a year.

Nonperiodicals

Monographs—Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NBS, NBS annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NBS under the authority of the National Standard Data Act (Public Law 90-396).
NOTE: The Journal of Physical and Chemical Reference Data (JPCRD) is published quarterly for NBS by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements are available from ACS, 1155 Sixteenth St., NW, Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Bureau on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NBS under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NBS administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.
Order the **above** *NBS publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.*
Order the **following** *NBS publications—FIPS and NBSIR's—from the National Technical Information Service, Springfield, VA 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NBS pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NBS Interagency Reports (NBSIR)—A special series of interim or final reports on work performed by NBS for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Service, Springfield, VA 22161, in paper copy or microfiche form.

U.S. Department of Commerce
National Bureau of Standards
Gaithersburg, MD 20899