

DTIC FILE COPY

(2)

AD-A229 358

Technical Document 1931
September 1990

Delay-Throughput Performance Evaluator for Distributed Systems

TDMA and Token Ring Schemes
(Version 1)

IRI Corporation

1990
lc

Approved for public release; distribution is unlimited.

The views and conclusions contained in this report are those of the contractors and should not be interpreted as representing the official policies, either expressed or implied, of the Naval Ocean Systems Center or the U.S. Government.

7

NAVAL OCEAN SYSTEMS CENTER
San Diego, California 92152-5000

J. D. FONTANA, CAPT, USN
Commander

R. M. HILLYER
Technical Director

ADMINISTRATIVE INFORMATION

This report was prepared by the IRI Corporation under contract N66001-89-M-BY18, Mod P00001, program element 0602721N.

Released by
R. A. Wasilauski, Head
Computer Systems Software
and Technology Branch

Under authority of
A. G. Justice, Head
Information Processing
and Displaying Division

LH

Table of Contents

1. Introduction
 - 1.1 Background
 - 1.2 The Delay-Throughput Performance Evaluation Programs
2. Delay-Throughput Performance Evaluator for Packet-Switched Time-Division Multiple-Access (PS-TDMA) Distributed Systems
 - 2.1 Introduction
 - 2.2 The Structure of the PS-TDMA Model
 - 2.3 Instructions for Running the PSTDMA Program
 - 2.4 Output of the PSTDMA Program
3. Delay-Throughput Performance Evaluator for Circuit-Switched Time-Division Multiple-Access (CS-TDMA) Distributed Systems
 - 3.1 Introduction
 - 3.2 The Structure of the CS-TDMA Model
 - 3.3 Instructions for Running the CSTDMA Program
 - 3.4 Output of the CSTDMA Program
4. Delay-Throughput Performance Evaluator for Integrated Circuit-Switched and Packet-Switched Time-Division Multiple-Access (ITDMA) Distributed Systems
 - 4.1 Introduction
 - 4.2 The Structure of the ITDMA Model
 - 4.3 Instructions for Running the ITDMA Program
 - 4.4 Output of the PSTDMA Program
5. Analytical Performance Equations for PSTDMA and References for the TDMA Analyses
 - 5.1 References for the TDMA Analyses
 - 5.2 Analytical Performance Evaluation for PSTDMA: The System and Traffic Model
 - 5.3 Analytical Performance Evaluation for PSTDMA: Contiguous Slot Assignment Per station
 - 5.4 Analytical Performance Evaluation for PSTDMA: Single and Uniform Slot Per Frame Assignments

Table of Contents (Cont.)

6. Delay-Throughput Performance Evaluator for Timed Token Protocol (FDDI-I Type)
Token-Ring Systems

- 6.1 Introduction
- 6.2 The Structure of the FDDI Medium Access Control Model
- 6.3 Instructions for Running the FDDI Program
- 6.4 Output of the FDDI Program

- Appendix A. The PS-TDMA Program: Input and Output Examples
- Appendix B. The PS-TDMA Program: Source Code
- Appendix C. The CS-TDMA Program: Input and Output Examples
- Appendix D. The CS-TDMA Program: Source Code
- Appendix E. The ITDMA Program: Input and Output Example
- Appendix F. The ITDMA Program: Source Code
- Appendix G. The FDDI Program: Symmetric Case Example
- Appendix H. The FDDI Program: An Example for the 2 Classes of Stations Case
- Appendix I. The FDDI Program: Asymmetric Case Example
- Appendix J. The FDDI Program: Source Code

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



1. Introduction

1.1 Background

This work involves the development of a delay-throughput performance evaluator for distributed systems. Currently planned and future Navy distributed integrated computer and communications systems involve the extensive use of medium access control procedures for sharing distributed communications, processing and computing resources among distributed stations. This work contributes to the development of methods and tools for carrying out modeling, performance evaluation, analysis and design of such systems.

The principal area of work for this effort involves research and development of delay-throughput performance evaluation tools for distributed communications and computer multiple-access schemes. This development involves the modeling and analysis of generic TDMA and token-ring type medium access control architectures that are of key importance to present and future Navy distributed systems. 400

It has been well demonstrated that the performance features of distributed computer systems and their associated distributed computer communications networks are not separable. In considering distributed systems and networks that are essential to the execution of mission critical tasks, it is essential to be able to evaluate the message/task delay and system throughput performance under a military priority based traffic and messaging environment. The performance evaluation tools to be developed under this effort are to be integrated and used as key elements of the POD (performance oriented design) system. They would provide efficient tools, based upon academically developed analytical and combined analytical/simulation approaches, for the evaluation of key generic medium access control procedures which are critical ingredients of Navy distributed computer and communications systems and networks. They will allow the users and designers of such distributed systems to assess the delay-throughput performance behavior of their systems under various traffic loading and system and channel parameter conditions.

Our works as outlined in this report incorporate the following elements.

a. Development of performance evaluation models and delay-throughput analysis techniques for distributed systems involving TDMA and token-ring multiple-access procedures; with applications to Navy systems, including: MILSTD1553B, SAFENET II and JTIDS.

b. Development of analytical and joint analytical/simulation programs for the multiple-access systems investigated in (a) for the calculation of their delay-throughput performance.

c. The capability to carry out performance evaluation tradeoffs and analyses using the programs developed in (b).

1.2 The Delay-Throughput Performance Evaluation Programs

The delay performance evaluation programs developed are classified as follows.

A. Time-Division Multiple-Access (TDMA) models, which are further categorized into three separate models and programs:

A.1 PS-TDMA (Packet-Switched TDMA) performance evaluation program, for a TDMA station which shares a communications (or processing, computing, buffering, etc.) channel on a PS-TDMA basis, so that it transmits its packets in a packet-switched fashion during its dedicated time slots. We obtain the system throughput and the average, variance and distributions of the message delay and the station queue-size, using both simulation and analytical techniques. This program can also be used to evaluate the performance of a demand-assigned PS-TDMA system.

A.2 CS-TDMA (Circuit-Switched TDMA) performance evaluation program, for a TDMA station which shares a communications (or processing, computing, buffering, etc.) channel on a CS-TDMA basis, so that it transmits its established sessions (connections) in a circuit-switched fashion during its dedicated time slots. We obtain the system throughput and the average, variance and distributions of the message delay and the station queue-size, using both simulation and analytical techniques. This program can also be used to evaluate the performance of a demand-assigned CS-TDMA system.

A.3 I-TDMA (Integrated CS/PS TDMA) performance evaluation program, for a TDMA station which shares a communications (or processing, computing, buffering, etc.) channel on a multiplexed integrated PS-TDMA and CS-TDMA basis, so that it transmits its established sessions (connections) in a circuit-switched fashion during its dedicated CS time slots and transmits its packets on a PS-TDMA during its dedicated PS slots, as well as during the CS frame slots which remain unused. We obtain the system throughput for the PS and CS components, the session blocking probability for the CS connections, and the average, variance and distributions of the packet delay and the station queue-size for the PS subsystem, using both simulation and analytical techniques. This model also represents the delay-throughput performance of a demand-assigned integrated CS/PS services TDMA system.

B. Timed Token Rotation Protocol (FDDI-I Type) model, as applied to the operation of a token ring network. This protocol is of the same type employed by FDDI and the Navy SAFENET II networks. It also allows performance analysis for numerous distributed computer and communications systems that employ polling access protocols of this type to share access to system resources (such as system buses, communications channels, processors, memory, etc.)

An extensive simulation model has been set-up which allows the investigation of the performance of such systems under various network traffic loading conditions and system configurations. Performance results are obtained for the system throughput and for message delays and buffer queue-sizes.

The presented programs constitute the Version 1 of the corresponding developments. Further extensions and expansions are being planned.

2. Delay-Throughput Performance Evaluator for Packet-Switched Time-Division Multiple-Access (PS-TDMA) Distributed Systems

2.1 Introduction

In this Chapter, we describe the structure and provide the operation directions for the Packet-Switched Time Division Multiple-Access (PS-TDMA) program. In Section 2.2, the structures of the service, channel, message and traffic models are described, and the performance measures are defined. In Section 2.3, we provide detailed instructions for running the PS-TDMA program. Examples and the source codes are given in the appendices. The output is discussed in Section 2.4.

2.2 The Structure of the PS-TDMA Model

The Channel Structure

A Time Division Multiple Access (TDMA) channel structure is assumed. The (service, processing or communications) channel is shared among multiple user stations, on a synchronous basis. The channel time is divided into time slots (or, slots) so that

τ = duration of a slot [sec.] = transmission time of a single packet.

Time frames (or simply frames, or cycles) are consecutively identified across the channel, so that

A time frame consists m slots.

T_F = duration of a time frame = $m\tau$.

Each station is dedicated a number of slots within each time frame, during which it can transmit its packets. Since each station is then operating independently of each other station, and its performance is independent of that of any other station, we need to examine the performance of only an arbitrary single station, which is the station subsequently identified here (as "the station").

We set

The number of slots per frame dedicated to the station = n ;

where: $n \leq m$.

The channel structure is illustrated in Fig. 1.

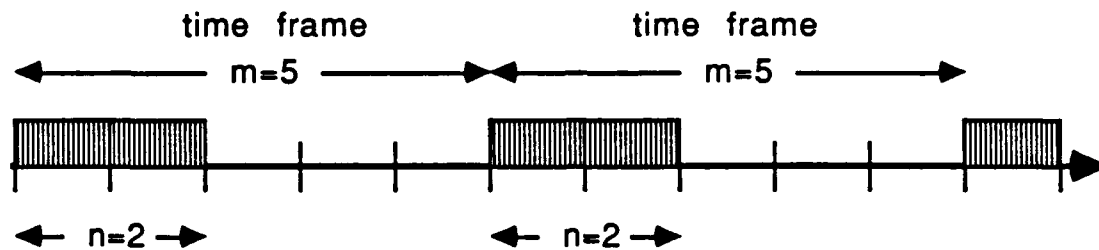


Fig. 1 Illustration of a TDMA channel structure whereby a station is allocated $n=2$ slots during each time frame of duration $m=5$ slots.

The Message Structure

Messages arrive at random at the station's buffer (whose capacity is not limited) and are served on a packet-switched store-and-forward FCFS (First come First Served) basis.

A message contains a random number (say P) of packets with an average given by:

Average number of packets per message = $1/q$,
where $0 < q \leq 1$.

Each packet is assumed to be a segment of fixed length, where the packet transmission time is equal to a single slot.

Since the transmission time of a single packet is equal to a single slot, if the message would have been transmitted continuously across the channel, its transmission rate would be

message transmission rate = q [messages/slot].

In the PSTDMA program it is further assumed that the number of packets per message (P) is governed by a Geometric distribution, so that q also designates the probability that a packet transmission is the last transmission of the underlying packet (i.e., with probability q this transmitted packet is the last packet and with probability $1-q$ the message contains more packets). The probability that a message contains k packets is given by the Geometric distribution:

$$\text{Prob}\{P=k\}=(1-q)^{k-1}q, \quad k=1,2,3,\dots$$

Example: A message contains an average of 5 packets. Then, $q=1/5=0.2$, and the transmission rate is equal to 5 [mess./slot]. If the transmission rate across the channel is equal to 10 Kbps and each packet (of fixed size) contains 1000 bits, then the packet transmission time=slot duration= $\tau=1000/10K=0.1$ sec.

The message average transmission time of 5 slots would take 0.5 sec if 5 consecutive slots are allocated to the transmission of this message. Otherwise, a longer transmission time would be required.

The Message Arrival Process

Messages are assumed to arrive into the station in accordance with a Geometric-Batch arrival point process described as follows. Message arrivals are recorded at the end of the arrival slot.

A batch of messages will arrive in a slot independently of arrival into other slots. We set:

$$P\{\text{a batch of messages arrives in a slot}\}=p;$$

$$P\{\text{no messages arrive in a slot}\}=1-p;$$

where $0 < p < 1$.

The number of messages in an arrival batch (say B) is a random variable, with an average that is set to be given by

$$\text{The average size of the arrival messages batch} = b \text{ [mess./batch]}.$$

The PSTDMA program developed by IRI allows the user to select one of three distributions for the number of messages in the batch:

1. A Deterministic distribution; under which the number of messages in the arrival batch is always fixed, equal to b.
2. A Geometric distribution with mean set equal to b.
3. A uniform distribution, whereby the number of messages in a batch is uniformly distributed between a specified lower value U_1 and upper level U_2 ; note the mean value is now equal to $(U_1 + U_2)/2$.

Performance Measures

Throughput

The station's throughput (TH_S) is equal to the number of messages per unit

time (or per slot) transmitted by the station across the channel. Under the PS-TDMA model the station throughput rate is equal to the message arrival rate (no messages are blocked) so that we have

$$TH_S = bp \text{ [mess./slot]}.$$

The packet throughput is given by

$$\text{Packet throughput} = bp/q \text{ [packets/slot]}.$$

Note that since each slot is equal to τ [sec.], where

$$\tau[\text{sec}] = (\text{average \#bits/packet}) / (\text{channel transmission rate in bps}),$$

we have, per station,

$$\text{Station Message Throughput} = bp/\tau \text{ [mess/sec]}$$

$$\text{Station Packet Throughput} = bp/q\tau \text{ [packets/sec]}.$$

The station's normalized throughput, or traffic intensity, is equal to the ratio of the traffic rate of arriving packets at the station and the channel's service rate dedicated to the station, and is thus given by

$$\text{Normalized Throughput} = \rho = (bp)/(nq/m).$$

For system stability (i.e, to ensure finite steady state system delays and buffer queue sizes) it is necessary to ensure that

$$\rho < 1.$$

Message Delay and Queue-Size

The message waiting-time is defined as:

W = Message Waiting Time = Total time elapsed since the message arrival slot to the time that the first packet of the message starts transmission

The message effective transmission time is

T = message effective transmission time = Time elapsed since the start of transmission of the session's first packet to the time that the last

packet ends transmission

The message delay time (D) is defined as:

D=Message Delay=Total time elapsed since the message arrival slot to the time that the last packet of the message is transmitted

Thus, we have

$$D=W+T.$$

The Message queue-size is:

X=Message queue-size=number of messages resident in the station's buffer (including the message in the process of transmission, if any)

The PSTDMA program provides results for the mean and standard-deviation of X and D as well as for the distributions

$$P(j)=P\{X=j\}; \quad D(j)=P\{D=j\}.$$

These results are obtained by us through the use of two methods:

a. We develop a simulation program that employs analytic recurrence relationships expressing the evolution of the system states. Random generators are used to generate the traffic loading. The sample mean, variance and distributions of the queue size and delay variables are then obtained through statistics collections and computations. The user is able to select the number of simulation runs (slots) for the stop time as well as the start time for the collection of statistics to incorporate in the calculation of the X and D performance measures.

b. We use methods developed by us and presented in the References which provide for analytical derivations of exact and/or lower and upper bound formulas for the steady state mean queue sizes and message delays.

Note that the analytical results are steady state results and thus describe performance results after the system has been running for a long time. Hence, the analytical based output results can differ from simulation results which are based on shorter run times.

2.3 Instructions for Running the PSTDMA Program

In the following we provide instructions for running the PSTDMA program (Version 1). It is noted that the specific name of the program used can be PSTDMA x where x is a number designating a version of the program compiled to incorporate certain limits on the program size. For example, PSTDMA2 involves the limit $m \leq 500$, and printed values for the queue-size (x) and for the delay (d) which are in the range $x \leq 100$, $d \leq 100$.

Step by Step Instructions (See Appendix A for examples)

1. Enter the name of the program (such as PSTDMA2)
Subsequent inputs are entered in response to program prompts.
 2. Enter the name of an output file; no more than 12 characters; in PC, quote the 'file name'
 3. Enter a number to designate the run method. Select
 - 1 - for simulation only
 - 2 - for analysis only
 - 3 - for simulation and analysis
 4. Enter the start time for simulation collection start
 5. Enter the stop time for the simulation length duration
 6. Enter the frame duration (m , an integer, where $1 \leq m \leq$ specified upper bound such as 500); then also type (following a space key entry) the number of slots allocated to the station (n , an integer, where $1 \leq n \leq m$)
 7. Enter the batch arrival rate ($p = p_1$, so that $0 < p < 1$, where p designates the probability that a batch arrives in a slot)
 8. Enter the message transmission rate ($q = q_1$, $0 < q \leq 1$, where q is the probability that the transmitted packet is the last one belonging to the underlying message, and $1/q$ is the average number of packets belonging to a message, or the average number of slots required to transmit a message)
 9. Enter the batch size distribution index; select
 - 1 - for deterministic (fixed) batch size
 - 2 - for Geometric batch size
 - 3 - for uniform batch size
 10. If selected deterministic or Geometric batch distributions, enter next the mean batch size b .
If selected a uniform distribution, select the lower and upper levels for the batch sizes u_1 and u_2 ; the program then calculates the mean batch size as $b = (u_1 + u_2)/2$.
- To ensure system stability, so that the message arrival rate is lower than

the channel message service rate, it is necessary to ensure that the normalized throughput ($\rho = \rho$) is less than one, or that

$$bp < (nq/m).$$

If the parameters selected violate this condition, the program indicates so and requests the user to re-select the system parameters.

11. When simulation is used, enter the levels x and d for the program to explicitly provide at the output the probabilities

$$P\{X > x\}, \quad P\{D > d\}.$$

12. The program then responds with 'please wait' and proceeds with its run. When done, the output is shown on the screen as well as provided in the specified output file.

Extension: An extended version PSTDMA5 includes in addition to the above described batch message arrival model also a Poisson arrival model. The structure of the program is similar. The user is asked to select which arrival model is desired. If the batch message arrival model (selection 1) is chosen, the remainder of the input is as described above. If a Poisson arrival process is chosen (selection 2), the only arrival parameter that must be selected is the message arrival rate λ expressed in [mess/slot] units. Note that $\lambda = bp$ represent the average message arrival rate per slot. The message length is Geometrically distributed with mean $1/q$. as for the batch arrival model. The program proceeds otherwise as described above and in the following.

2.4 Output of the PSTDMA Program

Appendix A provides two examples of the inputs and outputs for the PSTDMA program.

The output of the program contains the following information:

1. Statement of the input parameters.
2. The normalized throughput (ρ) for the station under consideration, in relation to the transmission time provided for this station
3. When simulation used, a table presenting the queue-size and message delay distributions, $P(j)$ and $D(j)$, respectively.
4. For simulation: the means and standard deviations for the station queue size and for the message delay; the probabilities $P(X > x)$, $P(D > d)$, for the selected x and d values
5. For analysis: the means or the station queue size and for the message delay, exact and upper and lower bound results.

As indicated in the References, the analytical method developed for the calculation of the upper and lower bounds is much more computationally efficient than the method developed for the analytical calculation of the exact result.

Further note that the analytical steady state mean delay $E(D)$ and mean queue size $E(X)$ are related through Little's Theorem so that

$$E(X) = \rho E(D).$$

3. Delay-Throughput Performance Evaluator for Circuit-Switched Time-Division Multiple-Access (CS-TDMA) Distributed Systems

3.1 Introduction

In this Chapter, we describe the structure and provide the operation directions for the Cacket-Switched Time Division Multiple-Access (CS-TDMA) program. In Section 3.2, the structures of the service, channel, message and traffic models are described, and the performance measures are defined. In Section 3.2, we provide detailed instructions for running the PS-TDMA program. Examples and the source codes are given in the appendices. The format of the output is discussed in Section 3.4.

3.2 The Structure of the CS-TDMA Model

The Channel Structure

A Time Division Multiple Access (TDMA) channel structure is assumed. The (service, processing or communications) channel is shared among multiple user stations, on a synchronous basis. The channel time is divided into time slots (or, slots) so that

τ = duration of a slot [sec.] = transmission time of a single segment

Time frames (or simply frames, or cycles) are consecutively identified across the channel, so that

A time frame consists m slots.

T_F = duration of a time frame = $m\tau$.

Each station is dedicated a number of slots within each time frame, during which it can transmit its established session's segments. Since each station is then operating independently of each other station, and its performance is independent of that of any other station, we need to examine the performance of only an arbitrary single station, which is the station subsequently identified here (as "the station").

We set

The number of slots per frame dedicated to the station = n ;

where: $n \leq m$.

The channel structure is illustrated in Fig. 2.

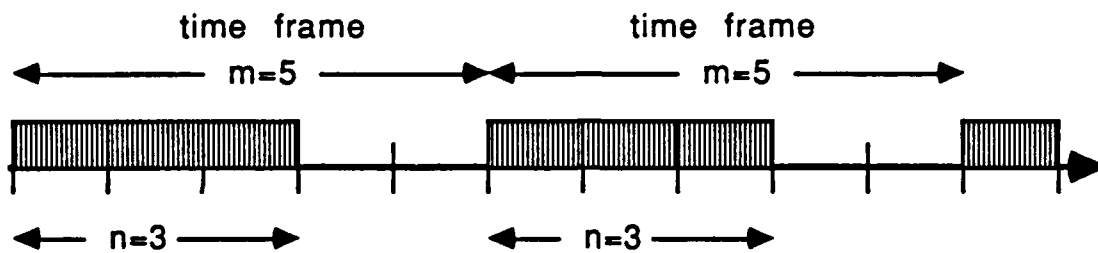


Fig. 2 Illustration of a TDMA channel structure whereby a station is allocated $n=3$ slots during each time frame of duration $m=5$ slots.

The Session Structure

Sessions (or connection requests) arrive at random at the station's buffer (whose capacity is not limited) and are served on a circuit-switched store-and-forward FCFS (First come First Served) basis.

Each session (connection) requires for its support across the channel the allocation of a single slot during each frame, for as long as the session lasts (i.e., for the session's holding time). The station transmits a single segment of information from the session's output during each session's designated slot.

Since the station is allocated only n slots per frame, it can simultaneously support at most n sessions. If a larger number of session requests for support arrive, they will be queued on a FCFS basis, and served when other sessions are terminated. Under the underlying CSTDMA model, a delayed rather than blocking service is provided, so that sessions arriving at a station when all of its dedicated slots are occupied would not be blocked (as in the ITDMA CS component model), but rather queued until served. The CSTDMA program provides performance analysis for determining this session delay.

A session lasts for a random number (say C) of frames with an average given by:

Average session holding time
= average number of frames per session connection = $1/q$,
where $0 < q \leq 1$.

The session transmission rate is thus also denoted as

session transmission rate = q [sessions/frame].

In the CSTDMA program it is further assumed that the session holding time (number of frames per session) is governed by a Geometric distribution, so that q also designates the probability that a segment frame transmission is the last transmission of the underlying session (i.e., with probability q this transmitted segment is the last segment and with probability $1-q$ the session contains more segments). The probability that a session lasts for k frames is given by the Geometric distribution:

$$\text{Prob}\{C=k\}=(1-q)^{k-1}q, \quad k=1,2,3,\dots$$

Example: A session average holding time is equal to 500 frames. Then, $q=1/500=0.002$, and the transmission rate is equal to $q=500$ [frames/session]. If the transmission rate across the channel is equal to $R=640$ Kbps and each segment (of fixed size) contains 100 bits, then

the segment transmission time=slot duration= $\tau=100/640K=0.15625$ msec.

Assume that each session requires a channel circuit operating at an average rate of 16 Kbps. Then, the number of frame slots must be set equal to

$$m=\text{channel rate}/\text{circuit rate}=640K/16K=40.$$

The frame thus contains $m=40$ slots, and the frame duration is then equal to

$$T_F=m\tau=6.25 \text{ msec.}$$

The session's average holding time is equal to 50 frames, so that the average session holding time is equal to $50 \times 6.25 \text{ msec}=3.125 \text{ sec.}$

The Session Arrival Process

Sessions (i.e., requests for allocation of a circuit to establish a session) are assumed to arrive into the station in accordance with a Geometric-Batch arrival point process described as follows. Session arrivals are recorded at the end of the arrival slot.

A batch of sessions will arrive in a slot independently of arrivals into other slots. We set:

$$P\{\text{a batch of sessions arrives in a slot}\}=p;$$

$$P\{\text{no sessions arrive in a slot}\}=1-p;$$

where $0 < p < 1$.

The number of sessions in an arrival batch (say B) is a random variable, with an average that is set to be given by

$$\text{The average size of the arrival sessions batch} = b \text{ [sessions./batch].}$$

The CSTDMA program developed by IRI allows the user to select one of three distributions for the number of sessions in the batch:

1. A Deterministic distribution; under which the number of sessions in the arrival batch is always fixed, equal to b .
2. A Geometric distribution with mean set equal to b .
3. A uniform distribution, whereby the number of sessions in a batch is uniformly distributed between a specified lower value U_1 and upper level U_2 ; note the mean value is now equal to $(U_1 + U_2)/2$.

Performance Measures

Throughput

The station's throughput (TH_S) is equal to the number of sessions per unit time (or per slot) transmitted by the station across the channel. Under the CS-TDMA model the station throughput rate is equal to the session arrival rate (no sessions are blocked) so that we have

$$TH_S = bp \text{ [sess./slot]}.$$

Note that since each slot is equal to τ [sec.], where

τ [sec] = (average #bits/segment)/(channel transmission rate in bps),
we have, per station,

$$\text{Station Message Throughput} = bp/\tau \text{ [sess/sec]}.$$

The station's normalized throughput, or traffic intensity, is equal to the ratio of the traffic rate of arriving sessions at the station and the channel's session service rate dedicated to the station, and is thus given by

$$\text{Normalized Throughput} = \rho = (bp)/(nq/m).$$

For system stability (i.e, to ensure finite steady state system delays and buffer queue sizes) it is necessary to ensure that

$$\rho < 1.$$

Message Delay and Queue-Size

The session waiting-time is defined as:

$W = \text{Session Waiting Time} = \text{Total time elapsed since the session's arrival slot to the time that the first segment of the session starts transmission}$

The session transmission time is

$T = \text{Session Transmission Time} = \text{Time elapsed since the start of transmission of the session's first segment to the time that the last segment ends transmission}$

The session delay time (D) is defined as:

$D = \text{Session Delay} = \text{Total time elapsed since the session's arrival slot to the time that the last segment of the session ends transmission}$

Thus, we have

$$D = W + T.$$

The station's queue-size is:

$X = \text{station queue-size} = \text{number of sessions (or session requests) resident in the station's buffer (including sessions to which a circuit is currently allocated, if any)}$

The CSTDMA program provides results for the mean and standard-deviation of X and D as well as for the distributions

$$P(j) = P\{X=j\}; \quad D(j) = P\{D=j\}.$$

These results are obtained by us through the use of two methods:

a. We develop a simulation program that employs analytic recurrence relationships expressing the evolution of the system states. Random generators are used to generate the traffic loading. The sample mean, variance and distributions of the queue size and delay variables are then obtained through statistics collections and computations. The user is able to select the number of simulation runs (slots) for the stop time as well as the start time for the collection of statistics to incorporate in

the calculation of the X and D performance measures.

b. We use methods developed by us and presented in the References which provide for analytical derivations of exact and/or lower and upper bound formulas for the steady state mean queue sizes and message delays.

Note that the analytical results are steady state results and thus describe performance results after the system has been running for a long time. Hence, the analytical based output results can differ from simulation results which are based on shorter run times.

3.3 Instructions for Running the CSTDMA Program

In the following we provide instructions for running the CSTDMA program (Version 1). It is noted that the specific name of the program used can be CSTDMA x where x is a number designating a version of the program compiled to incorporate certain limits on the program size. For example, CSTDMA2 involves the limit $m \leq 500$, and printed values for the queue-size (x) and for the delay (d) which are in the range $x \leq 100$, $d \leq 100$.

Step by Step Instructions (See Appendix C for examples)

1. Enter the name of the program (such as CSTDMA2)
Subsequent inputs are entered in response to program prompts.
2. Enter the name of an output file; no more than 12 characters; in PC, quote the 'file name'
3. Enter a number to designate the run method. Select
 - 1 - for simulation only
 - 2 - for analysis only
 - 3 - for simulation and analysis
4. Enter the start time for simulation collection start
5. Enter the stop time for the simulation length duration
6. Enter the frame duration (m , an integer, where $1 \leq m \leq$ specified upper bound such as 500); then also type (following a space key entry) the number of slots allocated to the station (n , an integer, where $1 \leq n \leq m$)
7. Enter the batch arrival rate ($p = p_1$, so that $0 < p < 1$, where p designates the probability that a batch arrives in a slot)
8. Enter the message transmission rate ($q = q_1$, $0 < q \leq 1$, where q is the probability that the transmitted segment is the last one belonging to the underlying message, and $1/q$ is the average session holding time, measured in frames)
9. Enter the batch size distribution (for the number of sessions arriving in a batch) index; select
 - 1 - for deterministic (fixed) batch size
 - 2 - for Geometric batch size
 - 3 - for uniform batch size
10. If selected deterministic or Geometric batch distributions, enter next the mean batch size b .
If selected a uniform distribution, select the lower and upper levels for the batch sizes u_1 and u_2 ; the program then calculates the mean batch size as $b = (u_1 + u_2)/2$.

To ensure system stability, so that the session arrival rate is lower than

the channel's session service rate for the underlying station, it is necessary to ensure that the normalized throughput ($\rho = \rho$) is less than one, or that

$$bp < (nq/m).$$

If the parameters selected violate this condition, the program indicates so and requests the user to re-select the system parameters.

11. When simulation is used, enter the levels x and d for the program to explicitly provide at the output the probabilities

$$P\{X > x\}, \quad P\{D > d\}.$$

12. The program then responds with 'please wait' and proceeds with its run. When done, the output is shown on the screen as well as provided in the specified output file.

Extension: An extended version CSTDMA5 includes in addition to the above described batch session arrival model also a Poisson arrival model. The structure of the program is similar. The user is asked to select which arrival model is desired. If the batch session arrival model (selection 1) is chosen, the remainder of the input is as described above. If a Poisson arrival process is chosen (selection 2), the only arrival parameter that must be selected is the session arrival rate λ expressed in [mess/slot] units. Note that $\lambda = bp$ represent the average session arrival rate per slot. The session length is Geometrically distributed with mean $1/q$. as for the batch arrival model. The program proceeds otherwise as described above and in the following.

3.4 Output of the CSTDMA Program

Appendix C provides two examples of the inputs and outputs for the CSTDMA program.

The output of the program contains the following information:

1. Statement of the input parameters.
2. The normalized throughput (ρ) for the station under consideration, in relation to the transmission time provided for this station
3. When simulation used, a table presenting the queue-size and session delay distributions, $P(j)$ and $D(j)$, respectively.
4. For simulation: the means and standard deviations for the station queue size and for the session delay; the probabilities $P(X > x)$, $P(D > d)$, for the selected x and d values
5. For analysis: upper and lower bounds for the means of the station queue size and for the session delay

As indicated in the References, the analytical method developed for the calculation of the upper and lower bounds is much more computationally efficient than the method developed for the analytical calculation of the exact result.

Further note that the analytical steady-state mean delay $E(D)$ and mean queue size $E(X)$ are related through Little's Theorem so that

$$E(X) = \rho E(D).$$

4. Delay-Throughput Performance Evaluator for Integrated Circuit-Switched and Packet-Switched Time-Division Multiple-Access (ITDMA) Distributed Systems

4.1 Introduction

In this Chapter, we describe the structure and provide the operation directions for the Integrated Circuit-Switched and Packet-Switched Time Division Multiple-Access (ITDMA) program. In Section 4.2, the structures of the service, channel, message and traffic models are described, and the performance measures are defined. In Section 4.3, we provide detailed instructions for running the ITDMA program. Examples and the source codes are given in the appendices. The output is discussed in Section 4.4.

4.2 The Structure of the ITDMA Model

The Channel Structure

A Time Division multiplexing channel structure is assumed. The (service, processing or communications) channel is shared among the multiple user Cs sessions and PS packets, on a synchronous basis.

A single station is considered. This station is allocated the global channel bandwidth and time for the transmission of its information. This station can serve as the network control station of an integrated CS/PS demand-assignment/TDMA (DA/TDMA), which is the basis for many military and commercial system and network operations.

The station accommodates two traffic types:

1. Sessions which are accommodated on a circuit-switched (CS) basis; a supported CS session (connection) requires the allocation of a single slot per frame for the duration of the session; during a slot the session transmits a segment
2. Messages which are served on a packet-switched (PS) FCFS basis. Each message is assumed here to consist of a single packet. The packet transmission time is equal to the slot duration.

The channel time is divided into time slots (or, slots) so that

τ = duration of a slot [sec.] = transmission time of a single PS-packet.

Time frames (or simply frames, or cycles) are consecutively identified across the channel, so that

A time frame consists m slots.

$$T_F = \text{duration of a time frame} = m\tau.$$

The station provides constrained higher priority to the support of CS sessions, by allocating within each frame a maximum of n slots which are used for the support of CS sessions, so that

the maximum number of slots per frame dedicated
to the support of CS sessions = n ;
where: $n \leq m$.

The remainder of the slots in the frame, $m-n$ slots, are always dedicated to the transmission of PS-packets. However, in addition, any slots within the frame which are not utilized for the support of CS-sessions are also made available for the transmission of PS-packets.

The channel structure is illustrated in Fig. 3.

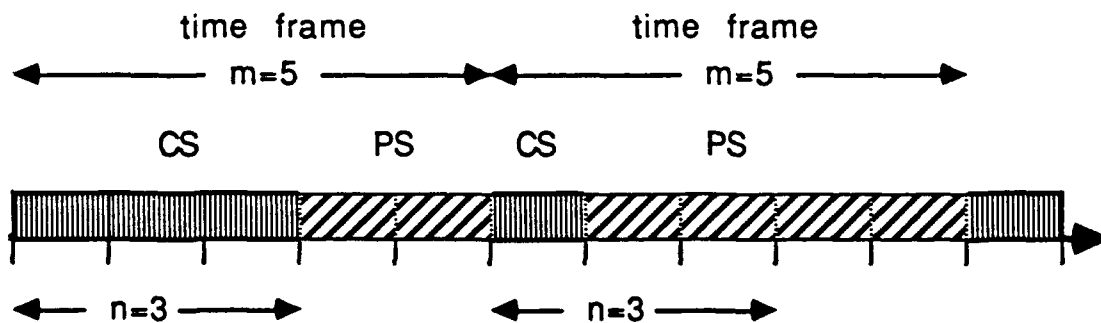


Fig. 3 Illustration of a Integrated CS-PS TDMA channel structure where a maximum of $n=3$ slots during each frame of duration $m=5$ slots are allocated to CS traffic; during the first frame all 3 CS slots are occupied by 3 CS sessions; during the second frame, only 1 CS slot is occupied, and the remaining 4 frame slots are used for PS-packet transmissions

The Session Structure

Sessions (or connection requests) arrive at random at the station's buffer (whose capacity is not limited) and are served on a circuit-switched store-and-

forward FCFS (First come First Served) basis using the n slots per frame allocated to the support of CS session traffic.

Each session (connection) requires for its support across the channel the allocation of a single slot during each frame, for as long as the session lasts (i.e., for the session's holding time). The station transmits a single segment of information from the session's output during each session's designated slot.

Since the station is allocated only n slots per frame, it can simultaneously support at most n sessions. A **blocking** CS service is assumed, so that a session arriving when all the CS slots are allocated will be blocked and assumed lost. The ITDMA program uses a Poisson/Geom/ n/n queueing system model to carry out the analysis of the CS session support subsystem. In particular, the following functions and probabilities are computed:

$$v_i = P\{i \text{ CS slots per frame are occupied by sessions}\}, 0 \leq i \leq n;$$

$$V = \text{average number of slots per frame occupied by CS session connections};$$

$$P_B = \text{blocking probability} = \text{probability that a session is blocked}.$$

A session lasts for a random number (say C) of frames with an average given by:

$$\begin{aligned} \text{Average session holding time} \\ = \text{average number of frames per session connection} = 1/q, \end{aligned}$$

where $0 < q \leq 1$.

The session transmission rate is thus also denoted as

$$\text{session transmission rate} = q \text{ [sessions/frame]}.$$

In the ITDMA program it is further assumed that the session holding time (number of frames per session) is governed by a Geometric distribution, so that q also designates the probability that a segment frame transmission is the last transmission of the underlying session (i.e., with probability q this transmitted segment is the last segment and with probability $1-q$ the session contains more segments). The probability that a session lasts for k frames is given by the Geometric distribution:

$$\text{Prob}\{C=k\} = (1-q)^{k-1}q, \quad k=1,2,3,\dots$$

The Session Arrival Process

Sessions (i.e., requests for allocation of a circuit to establish a session) are assumed to arrive into the station in accordance with a Poisson arrival point process characterized by the arrival rate λ

λ = Session arrival rate = Average number of session arrivals per slot.

The PS Message and Packet Structure

Messages arrive at random at the station's buffer (whose capacity is not limited) and are served on a packet-switched store-and-forward FCFS (First come First Served) basis.

A message is assumed here to consist of a single packet.

Each packet is assumed to be a segment of fixed length, where the packet transmission time is equal to a single slot.

The PS Packet Arrival Process

Packets (or messages) are assumed to arrive into the station in accordance with a Geometric-Batch arrival point process described as follows. Packet arrivals are recorded at the end of the arrival slot.

A batch of packets will arrive in a slot independently of arrival into other slots. We set:

$P\{\text{a batch of messages arrives in a slot}\} = p;$

$P\{\text{no messages arrive in a slot}\} = 1-p;$

where $0 < p < 1$.

The number of packets in an arrival batch (say B) is a random variable, with an average that is set to be given by

The average size of the arrival messages batch = b [mess./batch].

The ITDMA program developed by IRI allows the user to select one of three distributions for the number of packets in the batch:

1. A Deterministic distribution; under which the number of packets in the arrival batch is always fixed, equal to b .
2. A Geometric distribution with mean set equal to b .
3. A uniform distribution, whereby the number of packets in a batch is uniformly distributed between a specified lower value U_1 and upper level U_2 ; note

the mean value is now equal to $(U_1 + U_2)/2$.

Performance Measures

CS Session Throughput

The station's **CS session throughput** (TH_{CS}) is equal to the number of sessions per unit time (or per slot) transmitted by the station across the channel. Under the ITDMA model the station throughput rate is equal to the fraction of session arrival rate corresponding to unblocked sessions, so that we have

$$TH_{CS} = \lambda(1-P_B) \text{ [sess./slot]}.$$

Note that since each slot is equal to τ [sec]., we have

$$\text{Station Message Throughput} = \lambda(1-P_B)/\tau \text{ [sess/sec]}.$$

The station's **normalized CS session throughput**, or **CS session traffic intensity**, is equal to the ratio of the traffic rate of arriving sessions at the station and the channel's session service rate dedicated to the station, and is thus given by

$$\text{Normalized Throughput} = \rho_{CS} = [\lambda(1-P_B)]/(nq/m).$$

PS Packet Throughput

The **PS packet throughput** (TH_{PS}) is equal to the number of packets per unit time (or per slot) transmitted by the station across the channel. Under the PS-TDMA model the station throughput rate is equal to the message arrival rate (no messages are blocked) so that we have

$$TH_{PS} = bp \text{ [mess./slot]}.$$

Note that since each slot is equal to τ [sec.], where

τ [sec] = (average #bits/packet)/(channel transmission rate in bps),
we have, per station,

$$\text{Station Packet Throughput} = bp/\tau \text{ [packets/sec]}$$

The station's normalized PS packet throughput, or packet traffic intensity, is equal to the ratio of the traffic rate of arriving packets at the station and the channel's PS packet service rate allocated to the station. Since an average of V slots per frame are used for CS support, the number of slots per frame available for PS packet transmissions is equal to $m-n$. Hence we have

$$\text{Normalized PS packetThroughput} = \rho_{PS} = (\text{mbp}) / [m - V].$$

For system stability (i.e, to ensure finite steady state system delays and buffer queue sizes for PS traffic) it is necessary to ensure that

$$\rho_{PS} < 1.$$

Packet Delay and Queue-Size

The performance of the CS subsystem is expressed by the CS throughput presented above and by the session blocking probability obtained by the ITDMA program. The performance of the PS subsystem is expressed in terms of the station packet queue-size (also termed as system-size) and the packet delay.

The message waiting-time is defined as:

$W = \text{Message Waiting Time} = \text{Total time elapsed since the message arrival slot to the time that the first packet of the message starts transmission}$

The message effective transmission time is

$T = \text{message effective transmission time} = \text{Time elapsed since the start of transmission of the session's first packet to the time that the last packet ends transmission}$

The message delay time (D) is defined as:

$D = \text{Message Delay} = \text{Total time elapsed since the message arrival slot to the time that the last packet of the message is transmitted}$

Thus, we have

$$D=W+T.$$

The Message queue-size is:

X =Message queue-size=number of messages resident in the station's buffer
(including the message in the process of transmission, if any)

Note that for this ITDMA program a message consists of a single packet.

The ITDMA program provides results for the mean and standard-deviation of the PS X and D variables as well as for the distributions

$$P(j)=P\{X=j\}; \quad D(j)=P\{D=j\}.$$

These results are obtained by us through the use of two methods:

a. We develop a simulation program that employs analytic recurrence relationships expressing the evolution of the system states for the PS subsystem. Random generators are used to generate the traffic loading. The sample mean, variance and distributions of the queue size and delay variables are then obtained through statistics collections and computations. The user is able to select the number of simulation runs (slots) for the stop time as well as the start time for the collection of statistics to incorporate in the calculation of the X and D performance measures.

b. We use methods developed by us and presented in the References which provide for analytical derivations of lower and upper bound formulas for the steady state mean queue sizes and message delays for the PS subsystem. The Cs subsystem is analyzed by using a M/Geom/n/n queueing analysis.

Note that the analytical results are steady state results and thus describe performance results after the system has been running for a long time. Hence, the analytical based output results can differ from simulation results which are based on shorter run times.

4.3 Instructions for Running the ITDMA Program

In the following we provide instructions for running the ITDMA program (Version 1). It is noted that the specific name of the program used can be PSTDMA x where x is a number designating a version of the program compiled to incorporate certain limits on the program size. For example, ITDMA2 involves the limit $m \leq 50$, and printed values for the queue-size (x) and for the delay (d) which are in the range $x \leq 20$, $d \leq 20$.

Step by Step Instructions (See Appendix E for examples)

1. Enter the name of the program (such as ITDMA2)
Subsequent inputs are entered in response to program prompts.
2. Enter the name of an output file; no more than 12 characters; in PC, quote the 'file name'
3. Enter a number to designate the run method. Select
 - 1 - for simulation only
 - 2 - for analysis only
 - 3 - for simulation and analysis
4. Enter the start time for simulation collection start
5. Enter the stop time for the simulation length duration
6. Enter the frame duration (m , an integer, where $1 \leq m \leq$ specified upper bound such as 50); then also type (following a space key entry) the number of maximum slots allocated for CS support (n , an integer, where $n \leq m$)
7. Enter the CS session arrival rate ($\lambda > 0$ [sess./slot])
8. Enter the CS session transmission rate ($q = q_1$ [frames/session], where the average session holding time is equal to $1/q$ [frames/session])
9. Enter the batch arrival rate of PS packets ($p = p_1$, so that $0 < p < 1$, where p designates the probability that a batch of packets arrives in a slot); each message is assumed to contain a single packet
10. The program computes and exhibits the value of V , which expresses the average number of slots per frame used by the CS session traffic; this value must be used to ensure that the PS subsystem is stable, as noted in the following
11. Enter the PS packet arrival batch size distribution index; select
 - 1 - for deterministic (fixed) batch size
 - 2 - for Geometric batch size
 - 3 - for uniform batch size
12. If selected deterministic or Geometric batch distributions, enter next the mean batch size b .
If selected a uniform distribution, select the lower and upper levels for

the batch sizes u_1 and u_2 ; the program then calculates the mean batch size as $b=(u_1+u_2)/2$.

To ensure system stability, so that the PS packet arrival rate is lower than the channel average packet service rate, it is necessary to ensure that the normalized throughput ($\rho=\rho_{PS}$) is less than one, or that

$$mbp < m - V.$$

If the parameters selected violate this condition, the program indicates so and requests the user to re-select the PS subsystem parameters.

13. When simulation is used, enter the levels x and d for the program to explicitly provide at the output the probabilities

$$P\{X > x\}, \quad P\{D > d\}$$

for the PS subsystem queue size (X) and delay (V) measures.

14. The program then responds with 'please wait' and proceeds with its run. When done, the output is shown on the screen as well as provided in the specified output file.

4.4 Output of the ITDMA Program

Appendix E provides an example of the input and output for the ITDMA program.

The output of the program contains the following information:

1. Statement of the input parameters.
2. The normalized throughput ($\rho = \rho_{PS}$) for the PS subsystem
3. When simulation used, a table presenting the queue-size and message delay distributions, $P(j)$ and $D(j)$, respectively.
4. For simulation: the means and standard deviations for the station PS queue size and for the PS packet delay; the probabilities $P(X > x)$, $P(D > d)$, for the selected x and d values
5. The blocking probability for CS sessions
6. For analysis: upper and lower bounds for the means of the station queue size and for the message delay

As indicated in the References, the analytical method developed for the calculation of the upper and lower bounds is much more computationally efficient than the method developed for the analytical calculation of the exact result.

Further note that the analytical steady-state mean delay $E(D)$ and mean queue size $E(X)$ are related through Little's Theorem so that

$$E(X) = \rho E(D).$$

5. Analytical Performance Equations for PSTDMA and References for the TDMA Analyses

5.1 References for the TDMA Analyses

The TDMA analyses used in the PSTDMA, CSTDMA and ITDMA programs are based upon the analytical models, derivations and performance evaluations presented in the following references based upon research investigations carried out by Professor Izhak Rubin and his research group at the Electrical Engineering department of UCLA.

References

1. Rubin, I., and Z. Zhang, "Message Delay Analysis for TDMA schemes using Contiguous-Slot Assignments," Proceedings IEEE ICC'88, Philadelphia, PA. , June 1988
2. Rubin, I., and Z. Zhang, "Message Delay Analysis for TDMA schemes using Contiguous-Slot Assignments," UCLA Technical Report
3. Rubin, I. and Z. Zhang, "Message Delay and Queue Size Analysis for Circuit-Switched TDMA Systems," to be published in IEEE Transactions on Communications.
4. Z. Zhang and I. Rubin, "Bounds on the Mean System-Size and Delay for a Movable Boundary Integrated Circuit-Switched and Packet-Switched Communications Channel, UCLA Technical Report, 1989.

5.2 Analytical Performance Evaluation for PSTDMA: The System and Traffic Model

Under the PSTDMA program, the following model has been assumed.

1. A station is considered. The station is allocated n (or N) Slots per frame. The frame (cycle) consists of m (or M) slots.
2. Messages arrive at the station at random according to an independent (Geometric Batch) arrival process, so that if

A_n =number of messages arriving during the n-th slot
then $\{A_n, n \geq 1\}$ is a sequence of i.i.d. random variables. The moments of A are defined as:

$$a(i) = P\{A=i\}, i \geq 0; \quad (1)$$

with moments:

$$a = E(A) = \text{mean number of message arrivals per slot} = \lambda \quad (2)$$

$$a_2 = E(A^2) \quad (3)$$

$$a_3 = E(A^3). \quad (4)$$

Equivalently, the arrival process can be represented as a Geometric batch arrival process, characterized as follows.

2.1 A batch of messages arrives in a slot with probability p, $0 < p < 1$; no messages arrive in a slot with probability $1-p$.

2.2 The message batches are independent identically distributed (i.i.d.) random variables $\{G_n, n \geq 1\}$, with distribution

$$P(G_n=i) = g_i, i \geq 1 \quad (5)$$

and moments

$$b = E(G) = \text{Average batch size}; b_2 = E(G^2); b_3 = E(G^3) \quad (6)$$

In relating the two above-mentioned presentations of the independent arrival process, the following relationships hold:

$$g_i = a(i) / [1 - a(0)], i \geq 1; \quad (7)$$

$$p = 1 - a(0); \quad (8)$$

$$a = \lambda = pb \quad (9)$$

$$a_2 = pb_2 \quad (10)$$

$$a_3 = pb_3. \quad (11)$$

In the following, we present examples of batch distributions.

Geometrically Distributed Batches

For a Geometric batch size distribution with mean batch size set equal to b , we have:

$$g_i = (1-b^{-1})^{i-1}(b^{-1}), i \geq 1; \quad (12)$$

$$E(G) = b \quad (13)$$

$$b_2 = E(G^2) = (2-b^{-1})b^2 \quad (14)$$

$$b_3 = E(G^3) = b^3(6-6b^{-1}+b^{-2}) \quad (15)$$

Deterministic (Fixed) Batch Sizes

For batch sizes which are assumed to be of fixed (deterministic) size equal to b , we have

$$b_2 = b_3 = b \quad (16)$$

Uniformly Distributed Batch Sizes

For a batch size which is assumed continuously uniformly distributed in (u_1, u_2) , we have:

$$b = (u_1 + u_2)/2 \quad (17)$$

$$b_2 = (u_2^3 - u_1^3)/[3(u_2 - u_1)] \quad (18)$$

$$b_3 = (u_2^4 - u_1^4)/[4(u_2 - u_1)] \quad (19)$$

Poisson Arrival Process

Under a Poisson arrival process with intensity λ [mess/slot], we have

$$a(i) = P(A=i) = \exp(-\lambda)\lambda^i/i!, \quad i \geq 0; \quad (20)$$

$$a = E(A) = \lambda \quad (21)$$

$$a_2 = \lambda + \lambda^2 \quad (22)$$

$$a_3 = \lambda + 3\lambda^2 + \lambda^3. \quad (23)$$

3. A message contains a random number of packets. The packet transmission time is equal to a single slot duration.

Let B_n denote the number of packets contained in the n -th message. The packet length sequence $\{B_n, n \geq 1\}$ consists of i.i.d. random variables, characterized by the distribution and moments:

$$\beta(i) = P(B=i), i \geq 1; \quad (24)$$

$$\beta = \beta_1 = E(B); \beta_2 = E(B^2); \beta_3 = E(B^3). \quad (25)$$

The following special message length distributions are noted.

Geometrically Distributed Message Length

For a Geometrically distributed message length with parameter q and mean message length $1/q$, we have:

$$\beta(i) = P(B=i) = (1-q)^{i-1}q, i \geq 1; \quad (26)$$

$$\beta = \beta_1 = E(B) = 1/q; \quad (27)$$

$$\beta_2 = E(B^2) = (2-q)/q^2; \quad (28)$$

$$\beta_3 = E(B^3) = (6-6q+q^2)/q^3. \quad (29)$$

Fixed (Deterministic) Message Length

If the message contains a fixed number of packets, set equal to β , we have:

$$B = \beta; \beta_1 = \beta; \beta_2 = \beta^2; \beta_3 = \beta^3. \quad (30)$$

Uniformly Distributed Message Length

For a message length which is assumed continuously uniformly distributed in (u_1, u_2) , we have:

$$\beta = (u_1 + u_2)/2 \quad (31)$$

$$\beta_2 = (u_2^3 - u_1^3)/[3(u_2 - u_1)] \quad (32)$$

$$\beta_3 = (u_2^4 - u_1^4)/[4(u_2 - u_1)]. \quad (33)$$

The Frame Arrival Process

The number of arrivals during the n -th frame is set equal to $N_n, n \geq 1$. The frame arrival sequence $\{N_n, n \geq 1\}$ consists of i.i.d. random variables, with distribution and moments given by:

$$n(i) = P(N=i), i \geq 0; \quad (34)$$

$$n_1 = E(N); n_2 = E(N^2); n_3 = E(N^3). \quad (35)$$

When the slot arrival process $\{A_n, n \geq 1\}$ is assumed to be an independent process (to consist of i.i.d. random variables), as assumed here, we have:

$$N_n = \sum_{i=1}^M A_i, \quad (36)$$

where M denotes the number of slots per frame. Hence,

$$n_1 = E(N) = Ma = M\lambda; \quad (37)$$

$$n_2 = E(N^2) = (M^2 - M)a^2 + Ma_2; \quad (38)$$

$$n_3 = E(N^3) = M(M-1)(M-2)a^3 + 3M(M-1)aa_2 + Ma_3. \quad (39)$$

5.3 Analytical Performance Evaluation for PSTDMA: Contiguous Slot Assignment Per station

The PSTDMA program assumes contiguous assignment of station slots in each frame. The station is thus allocated N (or n) contiguous slots during each frame. The frame contains M (or m) slots.

Under such a slot allocation, our analysis yields the following performance results. Assumed here is a Geometric batch arrival process with batch arrival intensity p , $0 < p < 1$, batch-size mean b and moments b_2 and b_3 , and message length which is Geometrically distributed with parameter q and mean $1/q$; see Section 5.2 for details of the traffic load definitions and relationships.

The steady-state mean queue-size expressing the average number of messages resident in the station buffer, $E(X)$, is obtained to be given by the following equations.

$$E(X)=E(X_1)+E(X_2)+E(X_3), \quad (1)$$

where

$$E(X_1)=[2pb(1-pb)+pb_{22}]/[2(q-pb)]; \quad (2)$$

$$E(X_2)=\{q^2[b_{22}-pb^2+pb(2-q)]/[2(q-pb)]\}\{(M-N)/(Nq-Mpb)\}; \quad (3)$$

$$E(X_3)=\{(M-N)q^2/(2M)\}\left\{\sum_{j=1}^{N-1} [A(w_j)Q(w_j)+w_j][A(w_j)Q(w_j)-w_j]\right\} \quad (4)$$

and

$$A(z)=1-p+pG(z) \quad (5)$$

$$Q(z)=q+(1-q)z \quad (6)$$

where $B(z)$ is the generating function of the batch size distribution

$$G(z)=\sum_{i=1}^{\infty} z^i P(G=i) \quad (7)$$

and

$$b=E(G), \quad b_{22}=E\{G(G-1)\}=b_2-b. \quad (8)$$

The numbers $\{w_j, j=1, 2, \dots, N\}$ are the $N-1$ roots of

$$z^N - A(z)^M Q(z)^N \quad \text{inside } |z| < 1. \quad (9)$$

The mean message delay is then obtained by Little's formula to be:

$$E(D)=E(X)/pb. \quad (10)$$

The station's normalized traffic intensity is

$$\rho = (Mpb)/(Nq). \quad (11)$$

The condition, for system stability, so that a finite message delay is attained, is

$$\rho < 1, \text{ or } Mpb < Nq. \quad (12)$$

Upper and Lower Bounds for the Mean Queue-Size and Message Delay

We have developed equations for upper and lower bounds for the mean queue-size and mean message delays, which are easy to compute. These approximations are given as follows. They are used in the PSTDMA program. We have:

$$Q \leq E(X) \leq Q + [(N-1)/2][(M-N)q/M], \quad (13)$$

where

$$Q = [2a(1-a) + a_{22}]/[2(q-a)] + \{Lq[a_{22} - a^2 + a(2-q)]/[2(q-a)(Nq-Ma)]\}. \quad (14)$$

and

$$a = E(A) = pb; \quad a_{22} = E(A^2) - E(A) = a_2 - a = p(b_2 - b). \quad (15)$$

We note that the upper and lower bounds for the mean queue-size differ by at most a constant value of $(N-1)/2$. For $N=1$, a station is allocated a single slot per cycle and the upper and lower bounds coincide, yielding $E(X)=Q$.

The mean message delay upper and lower bounds are then computed using the relationship:

$$E(D) = E(X)/pb = E(X)/a. \quad (16)$$

5.4 Analytical Performance Evaluation for PSTDMA: Single and Uniform Slot Per Frame Assignments

In this section we make the assumption that the station is assigned a single slot ($N=1$) during a frame containing M slots.

If the station is assigned slots within a frame in a uniformly distributed manner, so that the station is allocated N slots in a cycle of duration L slots and

$$L/N = M \quad (1)$$

where M is an integer, the same channel configuration conditions occur so that the station is in fact assigned 1 slot every M slots, and the results of this section can be applied.

Additionally, the results of this section can also be used as an approximation if the ratio L/N is not an integer, whereby we use M to denote

$$M = \text{average number of frame slots per each allocated station slot.} \quad (2)$$

The key advantage of assuming a uniform distribution of station slots per frame (or, equivalently a single slot per cycle allocation) is that it induces a much simpler analytical model that we employ to compute exact results for the mean and the variance of the message delay, $E(D)$ and $\text{Var}(D) = \sigma^2(D)$, resp. The ability to analytically compute the standard-deviation $\sigma(D)$ is of prime importance in many applications in which the tail probability of the delay must be computed and constrained. Thus, for example, we can use the analytically computed value

$$D_{.99} \approx E(D) + 3\sigma(D), \quad (3)$$

to provide an estimate of the 99-th percentile delay, for which we guarantee

$$P(D \leq D_{.99}) \geq 0.99 \quad (4)$$

so that the message delay D can be guaranteed to be lower than $D_{.99}$ for 99% of the served messages. (Note that for a Gaussian distribution estimate 3 yields a 99.9% probability, while for an exponential distribution, for which $\sigma(D) = E(D)$, this estimate represents a 98% probability.)

In deriving the results for the PSTDMA system, we assume:

1. An independence arrival process with parameters as described in Section 5.2. Thus, the arrival process parameters involve the moments $a = \lambda$, a_2 and a_3 . Equivalently, the Geometric batch arrival model description induces the batch arrival probability p and the batch moment parameters b , b_2 and b_3 . We also use the frame arrival moments n , n_2 and n_3 . (See Section 5.2.)
2. The message can contain a random number of packets. The message length distribution has the moments β , β_2 , β_3 . (See Section 5.2.)
3. The station is assigned a single slot every M slots, as described above. The packet transmission time is equal to the duration of a single slot.

The mean message delay $E(D)$ is given by the following formulas.

$$E(D) = ME(W_D) + [(M-1)/2] + (M\beta + 1 - M), \quad (5)$$

where

$$E(W_D) = \{\rho/[2(1-\rho)]\} \{(\beta_2/\beta) + [\rho(n_2 - n_1)/n_1^2] - 1\} + (\beta/2)[(n_2/n_1) - 1], \quad (6)$$

and the normalized throughput (traffic intensity) is

$$\rho = n_1\beta = M\lambda\beta < 1. \quad (7)$$

For example, for the special case of a Poisson arrival process with intensity λ [mess/slot], we obtain

$$E(D) = \{M\rho/[2(1-\rho)]\}(\beta_2/\beta) + M\beta + 1 - M/2. \quad (8)$$

In deriving Eq. (8), we have added 1/2 slot to the delay to account for the continuous-time random nature of the Poisson arrival process, noting that a message may arrive anywhere within the slot, but according to the discrete model its arrival is recorded at the end of the slot.

The mean delay expression can also be written as:

$$E(D) = \{M\rho/[2(1-\rho)]\} \{(\beta_2/\beta) - 1\} + \{\rho^2/[2\lambda(1-\rho)]\} \{M\lambda - \lambda - 1 + a_2/\lambda\} \\ + (M\beta/2)[(M-1)\lambda + a_2/\lambda] + (M\beta + 1 - M)/2. \quad (9)$$

The variance of the message delay, $\text{Var}(D)$, is given by the following formulas.

$$\text{Var}(D) = \text{Var}(W_1) + \text{Var}(W_2) + \text{Var}(F) + \text{Var}(T), \quad (10)$$

where

$$\text{Var}(F) = (M^2 - 1)/12 \quad (11)$$

$$\text{Var}(T) = M^2(\beta_2 - \beta^2). \quad (12)$$

We have:

$$\text{Var}(W_1) = M^2 \{W_{12} + W_{11} - W_{11}^2\}, \quad (13)$$

$$\text{Var}(W_2) = M^2 \{W_{22} + W_{21} - W_{21}^2\}, \quad (14)$$

where

$$W_{11} = \{n_2 \beta_1^2 - n_1(\beta_1^2 + \beta_1 - \beta_2)\} / [2(1-\rho)]; \quad (15)$$

$$W_{21} = (n_2 - n_1) \beta_1 / 2n_1. \quad (16)$$

Note that $E(W_D) = W_{11} + W_{21}$. For the computation of W_{12} and W_{22} , we present the following formulas.

$$\begin{aligned} W_{12} = & \{1/[6(1-\rho)^2]\} \cdot \{3(n_2 - n_1)\beta_1^2 + 3n_1(\beta_2 - \beta_1) + 2(1-\rho)[(n_3 - 3n_2 + 2n_1)\beta_1^3 \\ & + 3(n_2 - n_1)\beta_1(\beta_2 - \beta_1) + n_1(\beta_3 - 3\beta_2 + 2\beta_1)]\}; \end{aligned} \quad (17)$$

$$\begin{aligned} W_{22} = & \{1/[6\rho\beta_1^2]\} \cdot \{3\beta_1(\beta_2 - \beta_1)[(n_2 - n_1)\beta_1^2 + n_1(\beta_2 - \beta_1)] + 2\rho\beta_1(\beta_3 - 3\beta_2 + 2\beta_1) \\ & - 2\beta_1^2[(n_3 - 3n_2 + 2n_1)\beta_1^3 + 3(n_2 - n_1)\beta_1(\beta_2 - \beta_1) + n_1(\beta_3 - 3\beta_2 + 2\beta_1)] \\ & - 3\rho(\beta_2 - \beta_1)^2\}. \end{aligned} \quad (18)$$

6. Delay-Throughput Performance Evaluator for Timed Token Protocol (FDDI-I Type) Token-Ring Systems

6.1 Introduction

In this Chapter, we describe the structure and provide the operation directions for the FDDI program, which provides a simulation based delay-throughput analysis of token ring networks employing the timed token protocol (FDDI-I type) medium access control (MAC) scheme, as used by SAFENET II.. In Section 6.2, the structures of the service, channel, message and traffic models are described, and the performance measures are defined. In Section 6.3, we provide detailed instructions for running the FDDI program. Examples and the source codes are given in the appendices. The output is discussed in Section 6.4.

6.2 The Structure of the FDDI Medium Access Control Model

The Network Topology

The network is configured as a token ring, where each station is actively inserted into the ring through its ring interface unit. Each pair of neighboring stations is connected by a unidirectional fiberoptic link. (A contra revolving ring is also used to increase the network reliability; however, its existence does not affect the delay-throughput performance of the network, which is based on the regular use of a single link.)

The network topology is illustrated in Fig. 4.

We set:

N =Number of Stations.

The access protocol is based on a token-based distributed polling procedure. A single token (packet of a determined pattern) exists across the ring. A station is allowed to transmit some of its packets (in accordance with the timed token protocol described below) only when it acquires the token. Upon the termination of transmission of its messages, the station immediately (early token release) generates a new token and transmits it across the unidirectional link to its neighboring station.

A critical parameter determining the delay-throughput efficiency of the token ring system is the underlying walk time. The ring walk time represents the time it takes for the token to walk around the ring when no station wishes to transmit. It involves, as components, the time it takes for the station to generate the token, transmit it to the neighboring station and the time it takes for the latter station to fully

receive the token, recognize it and act properly (i.e., immediately release it if the station does not wish to transmit across the channel, or remove the token from the ring if the station wishes to transmit its own messages across the ring).

We set:

$r(i)$ = walk time for the token from station i to station $i+1$.

Key components included in the walk time $r(i)$ are:

- the token transmission time (which is equal to the token length divided by the transmission rate across the channel)
- the token propagation delay from station i to station $i+1$, which depends upon the length of the fiber between these two stations, and is computed by using the medium propagation rate of

Propagation rate = $5\mu\text{sec/km}$.

Example: For example, for a system which uses a token which is 100 bits long, a channel transmission rate of 10 Mbps and an average interstation fiber length of 5 km, we have:

Token transmission time = $100/100\text{M} = 1\mu\text{sec}$;

Interstation propagation time $\approx 5\mu\text{sec/km} \times 5\text{km} = 25\mu\text{sec}$;

so that (when other delay are neglected) we have:

$r(i) = 26\mu\text{sec}$.

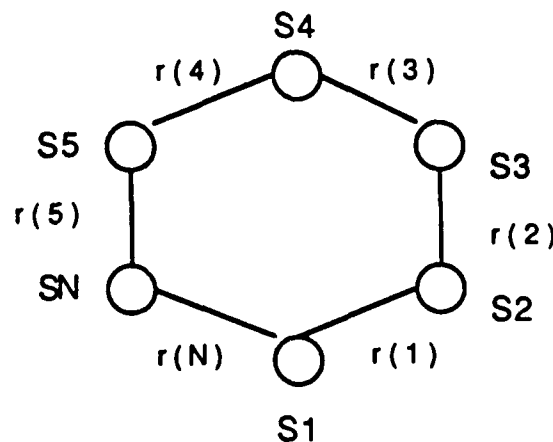


Fig. 4 The Token Ring Topology

The Message Structure

Messages arrive at random at the station's buffer (whose capacity is not limited) and are served on a packet-switched store-and-forward priority basis, in accordance with the timed token protocol.

Messages arriving at a station are categorized into the following priority classes:

- Synchronous messages, which are granted the highest priority
- Asynchronous messages which are further classified into multiple priority classes, so that priority-j asynchronous messages are granted higher priority than priority-(j+1) asynchronous messages. Thus, priority -1 messages are provided higher priority for accessing the ring than priority-2 messages.

We set

$N_p(i)$ = number of message priority classes for station i.

In our current (Version 1) FDDI program, each message is assumed to consist of a single packet. The packet is assumed to require a transmission time across the link which is:

Packet transmission time = Uniformly distributed, with a specified mean.

In the FDDI program, we set:

$\text{plens}(i)$ = average transmission time for synchronous packet at station i [msec]

$\text{plena}(i,j)$ = average transmission time for an asynchronous packet at station i, considering a packet of priority class j.

The Message Arrival Process

Messages are assumed to arrive into the station in accordance with a Poisson process, with the arrival rates given as follows:

$\text{as}(i)$ = arrival rate of synchronous messages (packets) at station i
[packets/msec]

$\text{aa}(i,j)$ = arrival rate of priority-j asynchronous packets at station i
[packets/msec]

The Timed Token Rotation access protocol and its parameters

The Timed Token Rotation access protocol operates as described in the following. We also present here the associated system and program parameters that must be selected.

During initialization, a target time is selected for token rotation, which is set in our FDDI program to be:

Target Token Rotation Time = TTRT [msec].

Upon the receipt of a token, the station can transmit its synchronous packets, provided the total time it spends for this transmission of synchronous packets during this visit of the token is not longer than a specified time limit called the bandwidth time. We thus set in the FDDI program:

Bandwidth time = BWT [msec].

To ensure that if all stations transmit synchronous packets for the maximum duration BWT the token rotation time will not exceed TTRT, we must select BWT so that

$$N \times BWT + \sum r(i) + \text{Max[asynchronous packet length]} \leq \text{TTRT}$$

where $\sum r(i)$ represents the total walk time around the ring (which is equal to Nr , when $r(i)=r$, each i , for symmetrically located stations around the ring).

Each station keeps a record of the token rotation time (TRT) representing the total time elapsed from the instant that it was visited by the token for the last time to the instant of current visit of the token.

Following the receipt of a token, assume the station to have transmitted synchronous packets for a total transmission time of $TT(0)$. Assume the station holds in its buffer priority-1 asynchronous packets which it wishes to transmit. to determine if the latter can be transmitted the following time threshold is selected:

$T_{\text{pri}(1)}$ = time threshold for priority-1 packet transmissions [msec].

The station then compares the time duration $TRT+TT(0)$ with this threshold. If

$$TRT+TT(0) < T_{pri}(1),$$

the station can transmit a priority-1 packet. Another priority 1 packet can be transmitted provided

$$TRT+TT(0)+TT(1) < T_{pri}(1),$$

where $TT(1)$ represents the total transmission time (up to this time) of priority-1 packets during this token visit.

Upon the termination of transmission of priority-1 packets, the station is allowed to transmit its priority-2 packets (if it has any) provided

$$TRT+TT(0)+TT(1)+TT(2) < T_{pri}(2),$$

where $TT(2)$ represents the total transmission time of priority-2 packets heretofore during this token visit, and

$$T_{pri}(2) = \text{time threshold for priority-2 packet transmissions [msec]}.$$

Similarly, for the priority based access control for asynchronous packets of lower priority levels.

Our FDDI program provides for a controlled exhaustive service at a station, so that synchronous packet transmissions can take place after the termination of an asynchronous packet transmission, provided their underlying transmission threshold has not been crossed yet. Similarly, for asynchronous packet transmissions.

Performance Measures

Throughput

The station's **throughput** (TH_S) is equal to the number of messages per unit time transmitted by the station across the channel. The station i priority- j throughput rate is then equal to

$$TH_S(i,j) = aa(i,j) \text{ [packets/msec]}.$$

while the normalized throughput for station i and priority-j packets is given by

$$\text{Normalized Throughput } (i,j)=\rho(i,j)=aa(i,j)plena(i,j) \text{ [Erlangs]}.$$

For synchronous traffic (which is also denoted here as priority-0 traffic class) we have:

$$TH_S(i,0)=as(i) \text{ [packets/msec]}.$$

$$\text{Normalized Throughput } (i,0)=\rho(i,0)=as(i)plens(i) \text{ [Erlangs]}.$$

The network normalized throughput for priority-j traffic is equal to

$$\rho(j) = \sum_i \rho(i,j), \quad j \geq 0.$$

For system stability (i.e, to ensure finite steady state system delays and buffer queue sizes) it is necessary to ensure that

$$\rho = \sum_j \rho(j) < 1, \quad j \geq 0.$$

Under the current version of the FDDI program, the station buffer is limited (to 200 packets), leading to a blocking probability $P_B(i,j)$ for station-i and priority-j packets. Unless the traffic intensity levels are very high, the blocking probabilities are kept very low. In turn, when very high traffic intensities (such as $\rho > 0.95$) are assumed, buffer overflows can occur and distinctive blocking probabilities result. In this case, the above formulas represent the offered traffic loads. The carried traffic loads are computed by multiplying the offered traffic loads by $(1-P_B)$, for the proper blocking probability factor (P_B).

Message Delay and Queue-Size

The message waiting-time is defined as:

W=Packet Waiting Time=Total time elapsed since the packet arrival instant to the instant that the packet starts transmission

The packet transmission time is:

T =packet transmission time

The packet delay time (D) is defined as:

D =Packet Delay=Total time elapsed since the message arrival time to the time that the packet is transmitted

Thus, we have

$$D=W+T.$$

ThePacket queue-size is:

X =Packet queue-size=number of packets resident in the station's buffer at the time of a token arrival to the station

The FDDI program measures also other queue-sizes:

X_a =Packet queue-size at packet arrival times to the station

X_d =Packet queue-size at packet departure times from the station

X_t =Packet queue-size at a station at an arbitrary time

The FDDI program provides results for the mean and standard-deviation of X , W and D as well as for the distributions

$$P(j)=P\{X=j\}; W(w)=P\{W\leq w\}, D(d)=P\{D\leq d\}.$$

In addition, the FDDI program provides statistical information about other system variables:

V_j =dwell time for priority- j messages at a station
= total transmission time of priority- j packets during a station's dwell time (token visit time)

V =dwell time of a station = total time used by a station to transmit packets

across the ring during a single token visit

C = cycle duration (relative to a station) = time elapsed between successive visits of the token to a station

In addition to obtaining performance results for each station and for each message priority class, we compute global performance measures that represent the means and standard deviations of X , W and D over all network stations, for each priority class.

To obtain these results, we have developed a simulation program that employs analytic recurrence relationships expressing the evolution of the system states. Random generators are used to generate the traffic loading. The sample mean, variance and distributions of the queue size, packet delay and other system variables are then obtained through statistics collections and computations. The user is able to select the simulation stop time as well as the start time for the collection of statistics to incorporate in the calculation of the performance measures.

6.3 Instructions for Running the FDDI Program

In the following we provide instructions for running the FDDI program (Version 1). It is noted that the specific name of the program used can be FDDI x where x is a number designating a version of the program compiled to incorporate certain limits on the program size.

Step by Step Instructions (See Appendices G-I for examples)

1. Enter the name of the program (such as FDDI)
Subsequent inputs can be entered by a data file or in response to program prompts. Appendices G-H describe both.
2. Enter the name of an output file
3. Enter a number to designate the network feature. Select
 - 1 - symmetric system
 - 2 - a system with 2 classes of stations
 - 3 - a system for which the traffic loading can change from station to station

Under a symmetric system feature, the stations are assumed to be statistically identical, so that the same traffic loads are offered to each station. All interstation walk times are assumed identical. Appendix G illustrates the input and output formats of this case.

Under system feature 2, the network consists of two classes of stations. Stations belonging to the same class are assumed statistically identical. This model allows the evaluation of networks where class 1 stations are characterized with certain service types and traffic loads while class 2 stations experience different service mixtures and/or higher traffic loads, or serve as gateways to other networks and thus involve different traffic characteristics. Appendix H illustrates the input and output formats of this case.

Under system feature 3, an asymmetric network configuration can be defined, where each station can have its own uniquely defined traffic loading features. Appendix I illustrates the input and output formats of this case.

In the following, we illustrate the inputs for a symmetric system; they are similar for the other cases.

4. Enter the start time for simulation collection start [msec]
5. Enter the stop time for the simulation length duration [msec]
6. Enter w, d, x for computing $P(W > w)$, $P(D > d)$, $P(X > x)$.
7. Enter the number of stations (N)
8. Enter the walk time (r [msec])

9. Enter the value selected for TTRT [msec]
10. For synchronous traffic, one then enters:
 - 10.1 packet arrival rate (as, packets/msec per station)
 - 10.2 mean packet transmission time (plens, msec)
 - 10.3 Bandwidth time (BWT, msec)
11. For asynchronous traffic, for each priority class j (and for each station type for feature cases other than symmetric) one then enters:
 - 11.1 packet arrival rate (aa(i,j), packets/msec per station)
 - 11.2 mean packet transmission time (plena(i,j), msec)
 - 11.3 Threshold time (T_pri(j), msec)

6.4 Output of the FDDI Program

Appendices G-I provide three examples of the inputs and outputs for the FDDI program.

The output of the program contains the following information:

1. Statement of the input parameters.
2. The normalized throughput load offered and realized. Also computed is the theoretical maximum normalized throughput.
3. Total number of cycles realized during the simulation run is presented.
4. We then present the means and standard deviations for the station queue-size (X), for message wait-time (W) and for message delay (D). These values are first averages over all the stations, for each message priority class (and for each station class for configuration feature 2). These values are then presented for each station.
5. For each station and each message priority class, performance results are then exhibited for:
 - 5.1 mean and standard deviation of v_j , the station dwell time for the transmission of priority j messages
 - 5.2 the probabilities $P(X > x)$, $P(W > w)$, $P(D > d)$, for the selected values of x , w and d
6. For each station, presented are results for
 - 6.1 mean and standard deviation of V , the station's dwell time
 - 6.2 mean and standard deviation of C , the station's cycle time
 - 6.3 mean and standard deviation of $C-V$, the station's vacation time
7. For each station and for each message priority class served by each station, performance results are presented
 - 7.1 for the queue size distributions: $P(X=j)$, $P(X_g=j)$, $P(X_d=j)$, $P(X_t=j)$
 - 7.2 for the message wait-time distribution $P(W \leq w)$
 - 7.3 for the message delay-time distribution $P(D \leq d)$
8. Also noted is the realized normalized throughput per station.

Delay-Throughput Evaluator, IRI Corp.

Appendix A. The PS-TDMA Program: Input and Output Examples

```

pstdma2
Enter the file_name for data:
(no more than 12 characters)
(in PC, QUOTE the 'file_name')
'pout1'
What would you like to do ?
  1: Simulation only
  2: Analysis only
  3: Simulation and analysis
Please enter a number:
3
Enter the start time (when statistic starts)
10
Enter the stop time (total simulation time)
1000
Enter the frame duration (m) ( $\leq 500$ ) and number
of slots (n) allocated to a station ( $n \leq m$ )
m and n should be integer
10 5
Enter the batch arrival rate [batch./slot] p1
0.1
Enter the transmission rate [mess./slot] q1
0.5
Enter the batch size distribution index
  1: deterministic
  2: geometric
  3: uniform

2
Enter the mean batch size, b ( $b \cdot p1 < n \cdot q1 / m$ ):
2
To find the probabilities  $P(X > x)$  and  $P(D > d)$ , please
enter x and d ( $x \leq 100$ ,  $d \leq 100$ )
x and d should be integer
10 10
please wait

```

PS-TDMA EXAMPLE 1

** Simulation and Analysis **

PS-TDMA System Parameters:

m, n = 10 5
the mean batch arrival rate p1 1.000000000000000E-001
the mean service rate q1 5.000000000000000E-001
system throughput rho 8.000000000000000E-001
the batch size is ; geometric
the mean batch size is : 2.000000000000000
The queue-size probabilities are:

j	P(j)	D(j)
0	.200101010	.000000000
1	.091616162	.329718739
2	.072525253	.015249068
3	.083030303	.011860386
4	.062828283	.017282277
5	.071919192	.016265673
6	.068989899	.019654354
7	.052020202	.020332091
8	.037676768	.018976618
9	.042929293	.013893595
10	.027272727	.019315486
11	.025252525	.014910200
12	.020404040	.013554727
13	.017777778	.014571332
14	.015858586	.010504914
15	.015757576	.011521518
16	.012828283	.014232464
17	.013737374	.013893595
18	.007272727	.014571332
19	.006363636	.015926804
20	.002323232	.010504914
21	.003232323	.012876991
22	.003232323	.012538123
23	.002424242	.012538123
24	.001313131	.013215859
25	.002929293	.010843782
26	.002525253	.012876991
27	.002222222	.010504914
28	.003737374	.012876991
29	.001010101	.010504914
30	.002424242	.008471705
31	.001010101	.012199254
32	.001212121	.012538123
33	.002525253	.005083023
34	.004040404	.010166045
35	.001515152	.006777364
36	.002121212	.006438495
37	.002121212	.005421891
38	.001818182	.007116232

39	.003434343	.005083023
40	.001212121	.005421891
41	.002424242	.006099627
42	.001919192	.007455100
43	.000808081	.005421891
44	.000202020	.007455100
45	.000101010	.006099627
46	.000000000	.004405286
47	.000000000	.003727550
48	.000000000	.004744155
49	.000000000	.005421891
50	.000000000	.006438495
51	.000000000	.006099627
52	.000000000	.006438495
53	.000000000	.003727550
54	.000000000	.003049814
55	.000000000	.002710945
56	.000000000	.004066418
57	.000000000	.004066418
58	.000000000	.007116232
59	.000000000	.004066418
60	.000000000	.002372077
61	.000000000	.002372077
62	.000000000	.003388682
63	.000000000	.003049814
64	.000000000	.002033209
65	.000000000	.002372077
66	.000000000	.002033209
67	.000000000	.002033209
68	.000000000	.001355473
69	.000000000	.003388682
70	.000000000	.002033209
71	.000000000	.003388682
72	.000000000	.000677736
73	.000000000	.000677736
74	.000000000	.002372077
75	.000000000	.001016605
76	.000000000	.001355473
77	.000000000	.001694341
78	.000000000	.000000000
79	.000000000	.000677736
80	.000000000	.001694341
81	.000000000	.000338868
82	.000000000	.001355473
83	.000000000	.000677736
84	.000000000	.000338868
85	.000000000	.000677736
86	.000000000	.000677736
87	.000000000	.001016605
88	.000000000	.000338868
89	.000000000	.001016605
90	.000000000	.000000000

91	.0000000000	.000338868
92	.0000000000	.000338868
93	.0000000000	.000338868
94	.0000000000	.000338868
95	.0000000000	.000338868
96	.0000000000	.000677736
97	.0000000000	.000677736
98	.0000000000	.001016605
99	.0000000000	.001355473
>=100	.0000000000	.039308709

P(X> 10)= .189090909

P(D> 10)= .517451711

mean queue-size (simulation) is : 6.394646464646464

standard deviation of queue-size (simu.) is: 7.603460993452924

mean delay (simulation) is: 22.430362588952900

standard deviation of delay (simulation) is: 28.780133446363980

mean queue-size (analysis) is : 3.376208812762674

mean queue-size (analysis, lower bound) is: 3.3500000000000001

mean queue-size (analysis, upper bound) is: 4.3500000000000001

mean delay (analysis) is: 16.881044063813370

mean delay (analysis, lower bound) is: 16.750000000000000

mean queue-size (analysis, upper bound) is: 21.750000000000010

```

C>pstdma2
Enter the file_name for data:
(no more than 12 characters)
(in PC, QUOTE the 'file_name')
'pout2'
What would you like to do ?
  1: Simulation only
  2: Analysis only
  3: Simulation and analysis
Please enter a number:
3
Enter the start time (when statistic starts)
10
Enter the stop time (total simulation time)
1000
Enter the frame duration (m) ( $\leq 500$ ) and number
of slots (n) allocated to a station ( $n \leq m$ )
m and n should be integer
10 1
Enter the batch arrival rate [batch./slot] p1
0.01
Enter the transmission rate [mess./slot] q1
0.2
Enter the batch size distribution index
  1: deterministic
  2: geometric
  3: uniform
3
Enter the range of the batch size:
(e.g., "1.0, 3.0")  $(u1+u2)*p1/2 < n < q1/m$ 
1.0,2.6
To find the probabilities  $P(X>x)$  and  $P(D>d)$ , please
enter x and d ( $x \leq 100$ ,  $d \leq 100$ )
x and d should be integer
20 10
please wait

```

PS-TDMA EXAMPLE 2

** Simulation and Analysis **

PS-TDMA System Parameters:

m, n = 10 1
the mean batch arrival rate p1 1.000000000000000E-002
the mean service rate q1 2.000000000000000E-001
system throughput rho 9.000000000000000E-001
the batch size is : uniform
the range of the batch size is: [1.000000000000000:
2.600000000000000]

The queue-size probabilities are:

j	P(j)	D(j)
0	.049898990	.000000000
1	.033838384	.202479339
2	.091414141	.000000000
3	.064848485	.000000000
4	.076969697	.000000000
5	.060909091	.004132231
6	.096464646	.000000000
7	.084242424	.000000000
8	.059191919	.000000000
9	.108888889	.000000000
10	.068484848	.004132231
11	.065959596	.000000000
12	.074949495	.000000000
13	.028888889	.000000000
14	.019494949	.000000000
15	.007474747	.000000000
16	.006060606	.000000000
17	.002020202	.000000000
18	.000000000	.000000000
19	.000000000	.000000000
20	.000000000	.000000000
21	.000000000	.000000000
22	.000000000	.000000000
23	.000000000	.000000000
24	.000000000	.000000000
25	.000000000	.000000000
26	.000000000	.000000000
27	.000000000	.000000000
28	.000000000	.000000000
29	.000000000	.000000000
30	.000000000	.000000000
31	.000000000	.000000000
32	.000000000	.000000000
33	.000000000	.000000000
34	.000000000	.000000000

35	.0000000000	.008264463
36	.0000000000	.0000000000
37	.0000000000	.0000000000
38	.0000000000	.0000000000
39	.0000000000	.0000000000
40	.0000000000	.0000000000
41	.0000000000	.0000000000
42	.0000000000	.0000000000
43	.0000000000	.0000000000
44	.0000000000	.0000000000
45	.0000000000	.0000000000
46	.0000000000	.0000000000
47	.0000000000	.004132231
48	.0000000000	.0000000000
49	.0000000000	.0000000000
50	.0000000000	.008264463
51	.0000000000	.0000000000
52	.0000000000	.0000000000
53	.0000000000	.0000000000
54	.0000000000	.0000000000
55	.0000000000	.0000000000
56	.0000000000	.0000000000
57	.0000000000	.0000000000
58	.0000000000	.0000000000
59	.0000000000	.0000000000
60	.0000000000	.004132231
61	.0000000000	.0000000000
62	.0000000000	.004132231
63	.0000000000	.0000000000
64	.0000000000	.0000000000
65	.0000000000	.0000000000
66	.0000000000	.0000000000
67	.0000000000	.0000000000
68	.0000000000	.0000000000
69	.0000000000	.0000000000
70	.0000000000	.0000000000
71	.0000000000	.0000000000
72	.0000000000	.004132231
73	.0000000000	.0000000000
74	.0000000000	.0000000000
75	.0000000000	.0000000000
76	.0000000000	.004132231
77	.0000000000	.004132231
78	.0000000000	.0000000000
79	.0000000000	.0000000000
80	.0000000000	.0000000000
81	.0000000000	.0000000000
82	.0000000000	.0000000000
83	.0000000000	.0000000000
84	.0000000000	.004132231
85	.0000000000	.0000000000
86	.0000000000	.0000000000
87	.0000000000	.0000000000

88	.000000000	.000000000
89	.000000000	.000000000
90	.000000000	.000000000
91	.000000000	.000000000
92	.000000000	.000000000
93	.000000000	.000000000
94	.000000000	.000000000
95	.000000000	.000000000
96	.000000000	.000000000
97	.000000000	.000000000
98	.000000000	.000000000
99	.000000000	.000000000
>=100	.000000000	.743801653

$P(X > 20) = .000000000$
 $P(D > 10) = .789256198$

mean queue-size (simulation) is : 6.847373737373737
 standard deviation of queue-size (simu.) is: 3.899230508336784

mean delay (simulation) is: 267.053719008264500
 standard deviation of delay (simulation) is: 202.557489423080300

mean queue-size (analysis) is : 2.5656000000000002
 mean queue-size (analysis, lower bound) is: 2.5656000000000002
 mean queue-size (analysis, upper bound) is: 2.5656000000000002

mean delay (analysis) is: 142.533333333333400
 mean delay (analysis, lower bound) is: 142.533333333333400
 mean queue-size (analysis, upper bound) is: 142.533333333333400

Delay-Throughput Evaluator, IRI Corp.

Appendix B. The PS-TDMA Program: Source Code

```

C:          Packet-Switched TDMA System
C
C          *****
C          *   Version 1   Sept. 20, 1989   *
C          *   Professor Izhak Rubin   *
C          *   Department of Electrical Engineering   *
C          *   Univeristy of California   *
C          *   Los Angels, CA 90024   *
C          *****
C
C: Descriptions of the variables (partial):
C:   pl   : mean message batch arrival rate
C:   b    : mean number of message arrivals per slot
C:   ql   : mean message service rate
C:   types : the type of batch size distribution
C:   u    : deterministic message btach size
C:   ul   : lower limit of the uniformly dist. message batch size
C:   u2   : uuper limit of the uniformly dist. message batch size
C:   seedt : the seed to generate the number of batch arrival
C:   seeds : the seed to generate the number of messages served
C:   ic(j) : total # of messages in the system at the start of jth slot
C:           (j=0,1,2,...,100)
C:   p(j)  : the prob. that there are j messages in the system at the
C:           start of the jth slot
C:   qmean : mean queue-size calculated from simulation
C:   wmean : mean message delay calculated from little's formula
C: External function: rand(seed)
C:           return a value that is uniformly distributed over (0, 1)

```

```

implicit real*8(a-h,o-z)
complex*16 x(500)
real*8 mu
integer xn(500),seedt,seeds,types,
+ batch,bs,icd(0:100,1:500),id(0:100),
+ ict(0:100,1:500),ic(0:100),types1,seedt1,dn(500)
common mu,u,ul,u2,seeds
character*12 filen
mm=100
print *, 'Enter the file_name for data: '
print *, '(no more than 12 characters)'
print *, '(in PC, QUOTE the ''file_name'')'
read *, filen
open(10,file=filen)
print *, 'What would you like to do ? '
print *, ' 1: Simulation only '
print *, ' 2: Analysis only '
print *, ' 3: Simulation and analysis '
print *, 'Please enter a number:'
read *, types1
if(types1.eq.1) then
  write(10,*) ' '
  write(10,*) '** Simulation **'
  print *, 'Enter the start time (when statistic starts)'
  read *, istart
  print *, 'Enter the stop time (total simulation time)'
  read *, istop
elseif(types1.eq.3) then
  write(10,*) ' '
  write(10,*) '** Simulation and Analysis **'
  print *, 'Enter the start time (when statistic starts)'
  read *, istart
  print *, 'Enter the stop time (total simulation time)'
  read *, istop
elseif (types1.eq.2) then
  write(10,*) ' '
  write(10,*) '**** Analysis ****'

```

```

endif
c
88 print *, 'Enter the frame duration (m) (<=500) and number'
print *, ' of slots (n) allocated to a station (n<=m)'
print *, ' m and n should be integer '
read *, m,n
print *, 'Enter the batch arrival rate [batch./slot] p1 '
read *, p1
print *, 'Enter the transmission rate [mess./slot] q1 '
read *, q1
1 print *, 'Enter the batch size distribution index'
print *, ' 1: deterministic'
print *, ' 2: geometric'
print *, ' 3: uniform'
print *, ' '
read *, types
if(types.eq.1) then
    print *, 'Enter the fixed batch size u (u*p1<n*q1/m):'
    read *, u
    rho=u*p1*m/(n*q1)
    if (rho.ge.1.d0) then
        print *, ' '
        print *, 'system not stable, please reenter the parameters '
        go to 88
    endif
    b=u*p1
    a2=p1*u*(u-1)
elseif(types.eq.2) then
    print *, 'Enter the mean batch size, b (b*p1<n*q1/m):'
    read *, b
    rho=b*p1*m/(n*q1)
    if (rho.ge.1.d0) then
        print *, ' '
        print *, 'system not stable, please reenter the parameters '
        go to 88
    endif
    mu=1.0d0/b
    b=b*p1
    a2=p1*2.d0*(1-mu)/mu**2
elseif(types.eq.3) then
    print *, 'Enter the range of the batch size:'
    print *, ' (e.g., "1.0, 3.0") (u1+u2)*p1/2<n*q1/m '
    read *, u1,u2
    rho=(u1+u2)*p1*m/(2*n*q1)
    if (rho.ge.1.d0) then
        print *, ' '
        print *, 'system not stable, please reenter the parameters '
        go to 88
    endif
    b=p1*(u1+u2)/2.d0
    a2=p1*(u2*(u2+1)*(2*u2+1)/6-u2*(u2+1)/2-(u1-1)*u1*(2*u1-1)
    * /6+u1*(u1-1)/2)/(u2-u1+1)
else
    go to 1
endif

```

```

c
write(10,*) ' PS-TDMA System Parameters: '
write(10,*) 'm, n = ',m,n
write(10,*) 'the mean batch arrival rate p1 ',p1
write(10,*) 'the mean service rate q1 ',q1
write(10,*) 'system throughput rho ',rho
if(types.eq.1) then
    write(10,*) 'the batch size is : deterministic '
    write(10,*) 'the fixed batch size is : ',u
elseif(types.eq.2) then
    write(10,*) 'the batch size is ; geometric '

```

```

        write(10,*) 'the mean batch size is : ',1/mu
elseif(types.eq.3) then
        write(10,*) 'the batch size is : uniform '
        write(10,*) 'the range of the batch size is: [',u1,':',u2,']'
endif
c
if(types1.eq.2) goto 125
print *, 'To find the probabilities P(X>x) and P(D>d), please '
print *, 'enter x and d (x<=100, d<=100)'
print *, 'x and d should be integer'
read(5,*) igx,igd
print *, 'please wait '
seedt=1099
seedt1=1099
seeds=99999
do 14 i=0,m
do 15 j=1,m
        ict(i,j)=0
        icd(i,j)=0
15 continue
ic(i)=0
id(i)=0
14 continue
c
totalx=0.0d0
tolx2=0.0d0
ktd=0
tdd=0
tdd2=0
do 16 i=1,m
16 xn(i)=0
c
do 117 i=1,500
117 dn(i)=0
npl=n+1
c
c ** simulation starts here
c
do 20 i=1,istop
mn=nb(seedt,p1)
if(xn(m).eq.0) then
        if(mn.eq.0) then
                xn(1)=0
        else
                bs=batch(types)
                xn(1)=mn*bs
        endif
else
        ndn=nb(seedt1,q1)
        if(mn.eq.0) then
                xn(1)=xn(m)-ndn
        else
                bs=batch(types)
                xn(1)=xn(m)-ndn+bs*mn
        endif
endif
l1=xn(1)+ndn
minx=min(500,l1)
do 149 kd=1,minx
149 dn(kd)=dn(kd)+1
c
if(ndn.eq.1) then
        kdd=dn(1)
        minx1=minx-1
        do 119 kd=1,minx1
119 dn(kd)=dn(kd)+1

```

```

        dn(minx1+1)=0
    endif
c
    if(i.gt.istart) then
        if(xn(1).gt.mm) then
            ict(mm,1)=ict(mm,1)+1
        else
            ict(xn(1),1)=ict(xn(1),1)+1
        endif
        totalx=totalx+xn(1)
        tolx2=tolx2+xn(1)**2
        if(ndn.eq.1) then
            ktd=ktd+1
            tdd=tdd+kdd
            tdd2=tdd2+kdd**2
            if(kdd.gt.mm) then
                icd(mm,1)=icd(mm,1)+1
            else
                icd(kdd,1)=icd(kdd,1)+1
            endif
        endif
    endif
c
    do 17 j=2,n
        mn=nb(seedt,p1)
        if(xn(j-1).eq.0) then
            if(mn.eq.0) then
                xn(j)=0
            else
                bs=batch(types)
                xn(j)=mn*bs
            endif
        else
            ndn=nb(seedt1,q1)
            if(mn.eq.0) then
                xn(j)=xn(j-1)-ndn
            else
                bs=batch(types)
                xn(j)= xn(j-1)-ndn+bs*mn
            endif
        endif
        ll=xn(j)+ndn
        minx=min(500,ll)
        do 139 kd=1,minx
139         dn(kd)=dn(kd)+1
c
        if(ndn.eq.1) then
            kdd=dn(1)
            minx1=minx-1
            do 129 kd=1,minx1
129         dn(kd)=dn(kd)+1
            dn(minx1+1)=0
        endif
c
        if(i.gt.istart) then
            if(xn(j).gt.mm) then
                ict(mm,j)=ict(mm,j)+1
            else
                ict(xn(j),j)=ict(xn(j),j)+1
            endif
            totalx=totalx+xn(j)
            tolx2=tolx2+xn(j)**2
            if(ndn.eq.1) then
                ktd=ktd+1
                tdd=tdd+kdd
                tdd2=tdd2+kdd**2

```

```

        if(kdd.gt.mm) then
            icd(mm,j)=icd(mm,j)+1
        else
            icd(kdd,j)=icd(kdd,j)+1
        endif
    endif
endif
17 continue
do 18 j=np1,m
    mn=nb(seedt,p1)
    bs=batch(types)
    xn(j)=xn(j-1)+mn*bs
C
    minx=min(500,xn(j))
    do 211 kd=1,minx
211 dn(kd)=dn(kd)+1
C
        if(i.gt.istart) then
            if(xn(j).gt.mm) then
                ict(mm,j)=ict(mm,j)+1
            else
                ict(xn(j),j)=ict(xn(j),j)+1
            endif
            totalx=totalx+xn(j)
            tolx2=tolx2+xn(j)**2
        endif
18 continue
C
20 continue
do 22 j=0,mm
    do 21 k=1,m
        ic(j)=ic(j)+ict(j,k)
        id(j)=id(j)+icd(j,k)
21 continue
22 continue
C
    pd=0
    px=0
    last=istop-istart
    print *, 'The queue-size probabilities are as follows:'
    write(6,*) 'j          P(j)          D(j)          '
    print *, '-----'
    write(10,*) 'The queue-size probabilities are:'
    write(10,*) 'j          P(j)          D(j)          '
    write(10,*) '-----'
    qmean=totalx/(last*m)
    var=dsqrt((tolx2-m*last*qmean**2)/(last*m-1))
    dmean=tdd/(ktd*1.0d0)
    vard=dsqrt((tdd2-ktd*dmean**2)/(ktd-1.0d0))
    do 110 i=0,mm
        if(i.lt.mm) then
            write(6,200) i,ic(i)*1.d0/(m*last),id(i)*1.0d0/ktd
            write(10,200) i,ic(i)*1.d0/(m*last),id(i)*1.0d0/ktd
        else
            write(6,210) i,ic(i)*1.d0/(m*last),id(i)*1.0d0/ktd
            write(10,210) i,ic(i)*1.d0/(last*m),id(i)*1.0d0/ktd
        endif
        if(i.gt.igx) px=px+ic(i)*1.d0/(m*last)
        if(i.gt.igd) pd=pd+id(i)*1.d0/ktd
110 continue
200 format(1x,i3,2x,2f14.9)
210 format(1x,'>=',i3,2x,2f14.9)
    write(6,*) 'mean queue-size (simulation) is : ',qmean
    write(6,*) 'standard deviation of queue-size (simu.) is:',var
    write(6,*) 'mean delay (simulation) is: ',dmean
    write(6,*) 'standard deviation of delay (simulation) is: ',vard

```

```

        write(6,300) igx,px
        write(6,301) igd,pd
        write(10,300) igx,px
        write(10,301) igd,pd
300    format(1x,'P(X>',i3,')= ',f14.9)
301    format(1x,'P(D>',i3,')= ',f14.9)
        write(10,*) 'mean queue-size (simulation) is : ',qmean
        write(10,*) 'standard deviation of queue-size (simu.) is: ',var
        write(10,*) 'mean delay (simulation) is: ',dmean
        write(10,*) 'standard deviation of delay (simulation) is:',vard
        if(types1.eq.1) then
            goto 126
        else
            goto 127
        endif
125    print *, 'please wait'
127    call tdma(x,n,m,p1,q1,sum1,types)
C *****
        d=q1*(2*b*(1-b)+a2)/(2*(q1-b))+q1**2*(a2-b**2+b*(2-q1))*(m-n)
        * / (2*(q1-b)*(n*q1-m*b))
        amean=d+(m-n)*q1**2/(2*m)*sum1
C *****
        write(6,*) 'mean queue-size (analysis) is : ',amean
        write(6,*) 'mean queue-size (analysis, lower bound) is: ',d
        dl=d+(m-n)*(n-1)*1.0d0/2/m
        write(6,*) 'mean queue-size (analysis,upper bound) is: ',dl
        write(6,*) 'mean delay (analysis) is: ',amean/b
        write(6,*) 'mean delay (analysis, lower bound) is: ',d/b
        write(6,*) 'mean queue-size (analysis,upper bound) is: ',dl/b
        write(10,*) 'mean queue-size (analysis) is : ',amean
        write(10,*) 'mean queue-size (analysis, lower bound) is: ',d
        write(10,*) 'mean queue-size (analysis,upper bound) is: ',dl
        write(10,*) 'mean delay (analysis) is: ',amean/b
        write(10,*) 'mean delay (analysis, lower bound) is: ',d/b
        write(10,*) 'mean queue-size (analysis,upper bound) is: ',dl/b
126    close(10)
        stop
        end
C
        subroutine tdma(x,n,m,p1,q1,sum1,types)
        implicit complex*16 (a-h,o-z)
        real*8 p1,q,q1,sum1,mu,u,u1,u2
        dimension x(1)
        integer seeds,types
        common mu,u,u1,u2,seeds
        x0=.2
        l=m-n
        call root(x0,l,n,p1,q1,x,types)
        n1=n-1
        sum=0.0d0
        do 15 i=1,n1
            q=q1+(1-q1)*x(i)
            if (types.eq.1) then
                a=1.0d0-p1+p1*x(i)**int(u*1.001)
            elseif(types.eq.2) then
                a=1-p1+p1*mu*x(i)/(1-(1-mu)*x(i))
            else
                a=1-p1+p1*(x(i)**int(u1*1.01)-x(i)**int(u2+1.01))/(1-x(i))
            *
                / (u2-u1+1)
            endif
            sum=sum+(a*q+x(i))/(a*q-x(i))
15    continue
C
        sum1=real(sum)
        return
        end

```

c

c

```

subroutine root(x0,l,n,p,q,x,types)
implicit complex*16 (a-h,o-z)
real*8 p,q,mu,u,u1,u2
integer seeds,types
dimension x(1)
common mu,u,u1,u2,seeds
n1=n-1
do 15 k=1,n1
  x1=x0
  call fix(k,l,n,p,q,x1,y,types)
  x(k)=y
15 continue
return
end

```

```

subroutine fix(k,l,n,p,q,x0,y,types)
implicit complex*16 (a-h,o-z)
real*8 p,delt,q,mu,u,u1,u2
integer types,seeds
common mu,u,u1,u2,seeds
itmax=1000
delt=1.0d-12
i=1
10 y=g(x0,k,l,n,p,q,types)
  if(i.eq.itmax) goto 20
  if(cabs(y-x0).le.delt) goto 15
  x0=y
  i=i+1
  go to 10
20 write(6,*) 'maximum number reached'
  stop
c write(6,*) ' x = ',y
c write(6,*) ' f(x) = ',x0-y
15 return
end

```

c

c

```

function g(x,k,l,n,p,q,types)
implicit complex*16 (a-h,o-z)
common mu,u,u1,u2,seeds
real*8 p,p1,q,q1,t,t1,mu,u,u1,u2
integer seeds,types
  q1=q+(1-q)*x
  if (types.eq.1) then
    a=1.0d0-p+p*x**int(u*1.001)
  elseif(types.eq.2) then
    a=1-p+p*mu*x/(1-(1-mu)*x)
  else
    a=1-p+p*(x**int(u1*1.01)-x**int(u2+1.01))/(1-x)
  *   / (u2-u1+1)
  endif
p1=2.0d0*3.14159d0*k/n
t=dcos(p1)
t1=-dsin(p1)
g=dcmplx(t,t1)*q1*a**(1.0d0+1*1.0d0/n)
return
end

```

c

c

```

integer function batch(types)
integer seeds,types
real*8 mu,u,u1,u2,rand
common mu,u,u1,u2,seeds
if(types.eq.1) then

```

```

        batch=int(u*1.001)
    elseif(types.eq.2) then
        batch=igeom(1.0d0/mu,seeds)
    else
        batch=ul+int( (u2-ul+1)*rand(seeds))
    endif
c    write(6,*) 'batch = func re ',batch
    return
end

```

C -----Random Generator Function-----

```

C This function will take the argument "seed" as an "input" seed,
C and return the value which is uniformly distributed over [0, 1],
C and the output seed to be used as the next input seed.
function rand(seed)
integer seed
double precision pp,i7,i2,rand
i7=7**5
i2=2**31-1
pp=i7*seed
seed=dmod(pp,i2)
rand=seed/(i2+1)
return
end

```

C ***** FUNCTION IGEOM *****

C FUNCTION IGEOM PRODUCES A GEOMETRICALLY DISTRIBUTED PSEUDO-RANDOM
C OBSERVATION WITH AVERAGE AVG , FROM R.N. STREAM ISTRM

```

FUNCTION IGEOM(AVG,ISTRM)
REAL*8 THETA,GI,FI,avg,uu,rand
Uu= RAND(istrm)
THETA = 1.0d0/AVG
NN=INT(10.0d0*AVG)
GI=THETA
FI=THETA
I=1
IF(Uu.LE.FI) GOTO 20
DO 10 I=2,NN
GI = (1.-THETA)*GI
FI = FI + GI
IF(Uu.LE.FI) GOTO 20
10 CONTINUE
c    WRITE(*,600) TNOW,AVG,Uu
c600  FORMAT(1X,'!!!! ERROR IN % IGEOM %, TIME = ',F12.4,
c    */2X,' AVG = ',F12.4,' Uu = ',F8.5)
20  IGEOM = I
    RETURN
END

```

C

```

c    This function will take the value 1 with proba. p and 0 with prob. 1-p
function nb(seed,p)
real*8 t,p,rand
integer seed
t=rand(seed)
if(t.le.p) then
    nb=1
else
    nb=0
endif
return
end

```

Delay-Throughput Evaluator, IRI Corp.

Appendix C. The CS-TDMA Program: Input and Output Examples

```

C>
C>cstdma2
Enter the file_name for data:
(no more than 12 characters)
(in PC, QUOTE the 'file_name')
'cout1'
What would you like to do ?
  1: Simulation only
  2: Analysis only
  3: Simulation and analysis
Please enter a number:
3
Enter the start time (when statistic starts)
10
Enter the stop time (total simulation time)
1000
Enter the frame duration (m) ( $\leq 500$ ) and number
of slots allocated to a station (n)
 $n \leq m$ , m and n should be integers
10 5
Enter the batch arrival rate p1
0.01
Enter the transmission rate q1
0.1
Enter the batch size distribution index
  1: deterministic
  2: geometric
  3: uniform
2
Enter the mean batch size, b ( $b \cdot p1 < n \cdot q1 / m$ ):
4
To find the probabilities  $P(X > x)$  and  $P(D > d)$ , please
enter x and d ( x,  $d \leq 100$ , both should be integers)
10 10
please wait

```

CS-TDMA EXAMPLE 1

** Simulation and Analysis **

CS-TDMA System Parameters:
 m, n= 10 5
 the mean batch arrival rate p1 1.000000000000000E-002
 the mean service rate q1 1.000000000000000E-001
 system throughput rho = 7.999999999999999E-001
 the batch size is : geometric
 the mean batch size is : 4.000000000000000

The queue-size probabilities are:

J	P(J)	D(J)
0	.062424242	.000000000
1	.067777778	.357392317
2	.068787879	.112922002
3	.085353535	.011641444
4	.066262626	.002328289
5	.047575758	.000000000
6	.058686869	.000000000
7	.030606061	.022118743
8	.029696970	.005820722
9	.028484848	.002328289
10	.034343434	.001164144
11	.030000000	.001164144
12	.014949495	.002328289
13	.028787879	.002328289
14	.017070707	.000000000
15	.027676768	.000000000
16	.021414141	.000000000
17	.043333333	.001164144
18	.029898990	.002328289
19	.026464646	.004656577
20	.026767677	.000000000
21	.011010101	.001164144
22	.012121212	.002328289
23	.026969697	.000000000
24	.010808081	.000000000
25	.008282828	.001164144
26	.003131313	.000000000
27	.005151515	.001164144
28	.015454545	.005820722
29	.023232323	.000000000
30	.010303030	.002328289
31	.013838384	.001164144
32	.004040404	.001164144
33	.005757576	.000000000
34	.000909091	.001164144
35	.000202020	.000000000

36	.002424242	.000000000
37	.000000000	.000000000
38	.000000000	.005820722
39	.000000000	.002328289
40	.000000000	.001164144
41	.000000000	.001164144
42	.000000000	.001164144
43	.000000000	.000000000
44	.000000000	.000000000
45	.000000000	.001164144
46	.000000000	.002328289
47	.000000000	.001164144
48	.000000000	.002328289
49	.000000000	.002328289
50	.000000000	.000000000
51	.000000000	.000000000
52	.000000000	.001164144
53	.000000000	.000000000
54	.000000000	.000000000
55	.000000000	.000000000
56	.000000000	.002328289
57	.000000000	.002328289
58	.000000000	.001164144
59	.000000000	.002328289
60	.000000000	.002328289
61	.000000000	.003492433
62	.000000000	.000000000
63	.000000000	.002328289
64	.000000000	.001164144
65	.000000000	.001164144
66	.000000000	.001164144
67	.000000000	.002328289
68	.000000000	.001164144
69	.000000000	.003492433
70	.000000000	.001164144
71	.000000000	.003492433
72	.000000000	.000000000
73	.000000000	.000000000
74	.000000000	.002328289
75	.000000000	.000000000
76	.000000000	.001164144
77	.000000000	.000000000
78	.000000000	.001164144
79	.000000000	.002328289
80	.000000000	.000000000
81	.000000000	.001164144
82	.000000000	.001164144
83	.000000000	.001164144
84	.000000000	.000000000
85	.000000000	.002328289
86	.000000000	.000000000
87	.000000000	.001164144
88	.000000000	.000000000

89	.000000000	.001164144
90	.000000000	.000000000
91	.000000000	.001164144
92	.000000000	.001164144
93	.000000000	.002328289
94	.000000000	.001164144
95	.000000000	.000000000
96	.000000000	.001164144
97	.000000000	.003492433
98	.000000000	.008149010
99	.000000000	.001164144
>=100	.000000000	.364377183

mean queue-size (simulation) is : 10.655757575757580
 standart deviation (queue-size, simu.) is: 8.976742237234827

mean delay (simulation) is: 113.440046565774200
 standard deviation (delay, simulation) is: 195.965372555913300

P(X> 10)= .420000000000
 P(D> 10)= .48428405122

mean queue-size (analysis, lower bound) is : 15.740000000000000
 mean queue-size (analysis, upper bound) is : 19.700000000000000

mean delay (analysis, lower bound) is: 393.499999999999900
 mean delay (analysis, upper bound) is: 492.499999999999900

cstdma2

Enter the file_name for data:
(no more than 12 characters)
(in PC, QUOTE the 'file_name')
'cout2'

What would you like to do ?

- 1: Simulation only
- 2: Analysis only
- 3: Simulation and analysis

Please enter a number:

3

Enter the start time (when statistic starts)

10

Enter the stop time (total simulation time)

1000

Enter the frame duration (m) (≤ 500) and number
of slots allocated to a station (n)
 $n \leq m$, m and n should be integers

10 1

Enter the batch arrival rate p1

0.01

Enter the transmission rate q1

0.2

Enter the batch size distribution index

- 1: deterministic
- 2: geometric
- 3: uniform

2

Enter the mean batch size, b ($b \cdot p1 < n \cdot q1 / m$):

1.8

To find the probabilities $P(X > x)$ and $P(D > d)$, please
enter x and d (x, $d \leq 100$, both should be integers)

10 10

please wait

CS-TDMA EXAMPLE 2

** Simulation and Analysis **

CS-TDMA System Parameters:
m, n= 10 1
the mean batch arrival rate p1 1.000000000000000E-002
the mean service rate q1 2.000000000000000E-001
system throughput rho = 9.000000000000000E-001
the batch size is : geometric
the mean batch size is : 1.800000000000000

The queue-size probabilities are:

J	P(J)	D(J)
0	.140707071	.000000000
1	.091717172	.419871795
2	.133434343	.000000000
3	.109292929	.000000000
4	.111717172	.003205128
5	.096060606	.000000000
6	.060505051	.000000000
7	.057575758	.000000000
8	.036262626	.000000000
9	.032828283	.000000000
10	.014040404	.000000000
11	.017373737	.000000000
12	.010707071	.000000000
13	.013535354	.000000000
14	.015353535	.006410256
15	.014141414	.000000000
16	.011414141	.000000000
17	.007676768	.000000000
18	.002020202	.000000000
19	.003737374	.000000000
20	.012121212	.000000000
21	.005353535	.006410256
22	.002424242	.003205128
23	.000000000	.006410256
24	.000000000	.000000000
25	.000000000	.003205128
26	.000000000	.000000000
27	.000000000	.000000000
28	.000000000	.000000000
29	.000000000	.000000000
30	.000000000	.000000000
31	.000000000	.000000000
32	.000000000	.000000000
33	.000000000	.003205128
34	.000000000	.000000000
35	.000000000	.003205128

36	.0000000000	.003205128
37	.0000000000	.0000000000
38	.0000000000	.0000000000
39	.0000000000	.0000000000
40	.0000000000	.003205128
41	.0000000000	.0000000000
42	.0000000000	.0000000000
43	.0000000000	.0000000000
44	.0000000000	.0000000000
45	.0000000000	.0000000000
46	.0000000000	.003205128
47	.0000000000	.003205128
48	.0000000000	.003205128
49	.0000000000	.0000000000
50	.0000000000	.003205128
51	.0000000000	.0000000000
52	.0000000000	.0000000000
53	.0000000000	.003205128
54	.0000000000	.0000000000
55	.0000000000	.0000000000
56	.0000000000	.003205128
57	.0000000000	.003205128
58	.0000000000	.003205128
59	.0000000000	.0000000000
60	.0000000000	.003205128
61	.0000000000	.0000000000
62	.0000000000	.003205128
63	.0000000000	.003205128
64	.0000000000	.0000000000
65	.0000000000	.003205128
66	.0000000000	.0000000000
67	.0000000000	.006410256
68	.0000000000	.012820513
69	.0000000000	.0000000000
70	.0000000000	.003205128
71	.0000000000	.0000000000
72	.0000000000	.0000000000
73	.0000000000	.003205128
74	.0000000000	.003205128
75	.0000000000	.0000000000
76	.0000000000	.0000000000
77	.0000000000	.003205128
78	.0000000000	.003205128
79	.0000000000	.003205128
80	.0000000000	.0000000000
81	.0000000000	.0000000000
82	.0000000000	.0000000000
83	.0000000000	.0000000000
84	.0000000000	.0000000000
85	.0000000000	.006410256
86	.0000000000	.0000000000
87	.0000000000	.003205128
88	.0000000000	.0000000000

89	.000000000	.000000000
90	.000000000	.000000000
91	.000000000	.003205128
92	.000000000	.000000000
93	.000000000	.003205128
94	.000000000	.000000000
95	.000000000	.006410256
96	.000000000	.000000000
97	.000000000	.006410256
98	.000000000	.000000000
99	.000000000	.003205128
>=100	.000000000	.429487179

mean queue-size (simulation) is : 4.856969696969697
 standart deviation (queue-size, simu.) is: 4.608791216124809

mean delay (simulation) is: 153.708333333333300
 standard deviation (delay, simulation) is: 247.684721276557900

P(X> 10)= .11585858586
 P(D> 10)= .57692307692

mean queue-size (analysis, lower bound) is : 15.228000000000010
 mean queue-size (analysis, upper bound) is : 15.228000000000010

mean delay (analysis, lower bound) is: 846.000000000000200
 mean delay (analysis, upper bound) is: 846.000000000000200

Delay-Throughput Evaluator, IRI Corp.

Appendix D. The CS-TDMA Program: Source Code

```

C:          Circuit-Switched TDMA System
C
C          *****
C          *   Version 1   Sept. 20, 1989   *
C          *   Professor Izhak Rubin       *
C          *   Department of Electrical Engineering *
C          *   Univeristy of California    *
C          *   Los Angels, CA 90024       *
C          *****
C: Descriptions of the variables:
C:   itbs   : total arrivals during a frame
C:   pl     : mean session arrival rate
C:   mu     : mean session service rate
C:   types  : the type of batch size distribution
C:   u      : deterministic batch size
C:   u1     : lower limit of the uniformly dist. batch size
C:   u2     : upper limit of the uniformly dist. batch size
C:   seedt  : the seed to generate the batch arrival
C:   seedt1 : the seed to generate the number of session served
C:           per slot
C:   ic(j)  : total number of sessions in that system at the start of
C:           jth slot (j=0,1,2,...,100)
C:   p(j)   : the prob. that herer are j sessions in the system at the start of
C:           a slot (j=0,1,2,...,100)
C:   qmean  : mean queue-size calculated from simulation
C: External function: rand(seed)
C:           return a value that is uniformly distributed over (0, 1)

```

```

implicit real*8(a-h,o-z)
integer xn(500),seedt,seeds,types,batch,bs
integer icd(0:100,1:500),id(0:100),dn(500)
integer ict(0:100,1:500),ic(0:100),types1,seedt1,index(500)
real*8 mu
common mu,u,u1,u2,seeds
character*12 filen
print *, 'Enter the file_name for data: '
print *, '(no more than 12 characters)'
print *, '(in PC, QUOTE the ''file_name'')'
read *, filen
open(10,file=filen)
print *, 'What would you like to do ? '
print *, ' 1: Simulation only '
print *, ' 2: Analysis only '
print *, ' 3: Simulation and analysis '
print *, 'Please enter a number:'
read *, types1
if(types1.eq.1) then
  write(10,*) ' '
  write(10,*) '** Simulation **'
  print *, 'Enter the start time (when statistic starts)'
  read *, istart
  print *, 'Enter the stop time (total simulation time)'
  read *, istop
elseif(types1.eq.3) then
  write(10,*) ' '
  write(10,*) '** Simulation and Analysis **'
  print *, 'Enter the start time (when statistic starts)'
  read *, istart
  print *, 'Enter the stop time (total simulation time)'
  read *, istop
elseif (types1.eq.2) then
  write(10,*) ' '
  write(10,*) '**** Analysis ****'
endif

```

```

C
88 print *, ' '

```

```

print *, 'Enter the frame duration (m) (<=500) and number'
print *, ' of slots allocated to a station (n) '
print *, ' n<=m, m and n should be integers'
read *, m,n
print *, 'Enter the batch arrival rate p1 '
read *, p1
print *, 'Enter the transmission rate q1 '
read *, q1
1 print *, 'Enter the batch size distribution index'
print *, ' 1: deterministic'
print *, ' 2: geometric'
print *, ' 3: uniform'
print *, ' '
read *, types
if(types.eq.1) then
    print *, 'Enter the fixed batch size u (u*p1<n*q1/m):'
    read *, u
    rho=u*p1*m/(n*q1)
    if (rho.ge.1.d0) then
        print *, 'system not stable, please reenter the parameters '
        go to 88
    endif
    b=u*p1
    a2=p1*u*(u-1)
elseif(types.eq.2) then
    print *, 'Enter the mean batch size, b (b*p1<n*q1/m):'
    read *, b
    rho=b*p1*m/(n*q1)
    if (rho.ge.1.d0) then
        print *, 'system not stable , please reenter the parameters '
        go to 88
    endif
    mu=1.0d0/b
    b=b*p1
    a2=p1*2.d0*(1-mu)/mu**2
elseif(types.eq.3) then
    print *, 'Enter the range of the batch size:'
    print *, ' (e.g., "1.0, 3.0") (u1+u2)*p1/2<n*q1/m '
    read *, u1,u2
    rho=(u1+u2)*p1*m/(2*n*q1)
    if (rho.ge.1.d0) then
        print *, 'system not stable, please reenter the parameters '
        go to 88
    endif
    b=p1*(u1+u2)/2.d0
    a2=p1*(u2*(u2+1)*(2*u2+1)/6-u2*(u2+1)/2-(u1-1)*u1*(2*u1-1)
    * /6+u1*(u1-1)/2)/(u2-u1+1)
else
    go to 1
endif

c
write(10,*) ' CS-TDMA System Parameters: '
write(10,*) 'm, n= ',m,n
write(10,*) 'the mean batch arrival rate p1 ',p1
write(10,*) 'the mean service rate q1 ',q1
write(10,*) 'system throughput rho = ',rho
if(types.eq.1) then
    write(10,*) 'the batch size is : deterministic '
    write(10,*) 'the fixed batch size is : ',u
elseif(types.eq.2) then
    write(10,*) 'the batch size is : geometric '
    write(10,*) 'the mean batch size is : ',1/mu
elseif(types.eq.3) then
    write(10,*) 'the batch size is : uniform '
    write(10,*) 'the range of the batch size is: [' ,u1,',',u2,'] '
endif

```

```

c      if(types1.eq.2) goto 125
      print *, 'To find the probabilities P(X>x) and P(D>d), please'
      print *, 'enter x and d ( x, d<=100, both should be integers)'
      read(5,*) igx,igd
      print *, 'please wait'
      seedt=1099
      seedt1=1099
      seeds=99999
      mm=100
      do 14 i=0,mm
      do 15 j=1,m
        ict(i,j)=0
15      icd(i,j)=0
      ic(i)=0
      id(i)=0
14      continue
c
      totalx=0.0d0
      tolx2=0.0d0
      ktd=0
      tdd=0
      tdd2=0
      px=0
      pd=0
      do 16 i=1,m
        index(i)=0
16      xn(i)=0
      do 117 i=1,500
117     dn(i)=0
c
c  **  simulation starts from here
c
      npl=n+1
      do 20 i=1,istop
        itbs=0
        mn=nb(seedt,p1)
        if (index(1).eq.0) then
          if(mn.eq.1) then
            bs=batch(types)
            xn(1)=xn(m)+bs
            itbs=itbs+bs
          else
            xn(1)=xn(m)
          endif
        else
          ndn=nb(seedt1,q1)
          if(ndn.eq.1) index(1)=0
          if(mn.eq.0) then
            xn(1)=xn(m)-ndn
          else
            bs=batch(types)
            xn(1)=xn(m)-ndn+bs*mn
            itbs=itbs+bs
          endif
        endif
        ll=xn(1)+ndn
        do 149 kd=1,ll
149      dn(kd)=dn(kd)+1
c
      if(ndn.eq.1) then
        kdd=dn(1)
        do 119 kd=1,xn(1)
119      dn(kd)=dn(kd+1)
        dn(xn(1)+1)=0
      endif

```

```

c
if(i.gt.istart) then
  if(xn(1).gt.mm) then
    ict(mm,1)=ict(mm,1)+1
  else
    ict(xn(1),1)=ict(xn(1),1)+1
  endif
  totalx=totalx+xn(1)
  tol2=tol2+xn(1)**2
  if(ndn.eq.1) then
    ktd=ktd+1
    tdd=tdd+kdd
    tdd2=tdd2+kdd**2
    if(kdd.gt.mm) then
      icd(mm,1)=icd(mm,1)+1
    else
      icd(kdd,1)=icd(kdd,1)+1
    endif
  endif
endif
endif

```

```

c
do 17 j=2,n
  mn=nb(seedt,p1)
  if (index(j).eq.0) then
    if(mn.eq.1) then
      bs=batch(types)
      xn(j)=xn(j-1)+bs
      itbs=itbs+bs
    else
      xn(j)=xn(j-1)
    endif
  else
    ndn=nb(seedt1,q1)
    if(ndn.eq.1) index(j)=0
    if(mn.eq.0) then
      xn(j)=xn(j-1)-ndn
    else
      bs=batch(types)
      xn(j)=xn(j-1)-ndn+bs*mn
      itbs=itbs+bs
    endif
  endif
  ll=xn(j)+ndn
  do 139 kd=1,ll
139    dn(kd)=dn(kd)+1

```

```

c
if(ndn.eq.1) then
  kdd=dn(1)
  do 129 kd=1,xn(j)
129    dn(kd)=dn(kd+1)
  dn(xn(j)+1)=0
endif

```

```

c
if(i.gt.istart) then
  if(xn(j).gt.mm) then
    ict(mm,j)=ict(mm,j)+1
  else
    ict(xn(j),j)=ict(xn(j),j)+1
  endif
  totalx=totalx+xn(j)
  tol2=tol2+xn(j)**2
  if(ndn.eq.1) then
    ktd=ktd+1
    tdd=tdd+kdd
    tdd2=tdd2+kdd**2
    if(kdd.gt.mm) then

```

```

        icd(mm,1)=icd(mm,1)+1
    else
        icd(kdd,1)=icd(kdd,1)+1
    endif
endif
endif
17 continue
c
do 18 j=np1,m
    mn=nb(seedt,p1)
    if(mn.eq.1) then
        bs=batch(types)
        xn(j)=xn(j-1)+bs
        itbs=itbs+bs
    else
        xn(j)=xn(j-1)
    endif
    do 211 kd=1,xn(j)
211    dn(kd)=dn(kd)+1
    if(i.gt.istart) then
        if(xn(j).gt.mm) then
            ict(mm,j)=ict(mm,j)+1
        else
            ict(xn(j),j)=ict(xn(j),j)+1
        endif
        totalx=totalx+xn(j)
        tolx2=tolx2+xn(j)**2
    endif
18 continue

    if(xn(m).ge.n) then
        do 78 k=1,n
78    index(k)=1
        else
            do 23 k=1,n
            if(itbs.gt.0) then
                if(index(k).eq.0) then
                    index(k)=1
                    itbs=itbs-1
                endif
            else
                goto 20
            endif
23    continue
        endif
20 continue
        do 22 j=0,mm
        do 21 k=1,m
            ic(j)=ic(j)+ict(j,k)
            id(j)=id(j)+icd(j,k)
21 continue
22 continue
c
        last=istop-istart
        print *, 'The queue-size probabilities are as follows:'
        write(6,*) 'j          P(j)          D(j)'
        print *, '-----'
        write(10,*) 'The queue-size probabilities are:'
        write(10,*) 'j          P(j)          D(j)'
        write(10,*) '-----'
        qmean=totalx/(last*m)
        var=dsqrt( tolx2/(last*m) -qmean**2 )
        wmean=tdd/(ktd*1.0d0)
        vard=dsqrt( tdd2/(ktd*1.0d0)-dmean**2 )
        do 110 i=0,mm
        if(i.lt.mm) then

```

```

        write(6,200) i,ic(i)*1.d0/(m*last),id(i)*1.d0/ktd
        write(10,200) i,ic(i)*1.d0/(m*last),id(i)*1.d0/ktd
        else
        write(6,210) i,ic(i)*1.d0/(m*last),id(i)*1.d0/ktd
        write(10,210) i,ic(i)*1.d0/(last*m),id(i)*1.d0/ktd
    endif
    if(i.gt.igx) px=px+ic(i)*1.0/(m*last)
    if(i.gt.igd) pd=pd+id(i)*1.d0/ktd
110 continue
200 format(1x,i3,2x,2f14.9)
210 format(1x,'>=',i3,2x,2f14.9)
    write(6,*) 'mean queue-size (simulation) is : ',qmean
    write(6,*) 'standart deviation (queue-size, simu.) is : ',var
    write(6,*) 'mean delay (simulation) is: ',wmean
    write(6,*) 'standart deviation (delay, simulation) is: ',vard
    write(10,*) 'mean queue-size (simulation) is : ',qmean
    write(10,*) 'standart deviation (queue-size, simu.) is: ',var
    write(10,*) 'mean delay (simulation) is: ',wmean
    write(10,*) 'standard deviation (delay, simulation) is: ',vard
    write(6,300) igx,px
    write(6,301) igd,pd
    write(10,300) igx,px
    write(10,301) igd,pd
300 format(1x,'P(X>',i3,')= ',f14.11)
301 format(1x,'P(D>',i3,')= ',f14.11)
    if(types1.eq.1) goto 126
c ***** analysis
125    amean= m*( m*b+(m*(m-1)*b**2+m*a2-(n-1)*m*b*q1
        +2*m*b*(1-q1))/(2*(n*q1-m*b)) )
        write(6,*) 'mean queue-size (analysis, lower bound) is : ',
        * amean/m-(m-1)*b/2
        write(6,*) 'mean queue-size (analysis, upper bound) is : ',
        * amean/m-(m-1)*b/2 +(n-1)*(2-q1)/2+ (n-1)*b
        write(6,*) 'mean delay (analysis, lower bound) is: ',
        * amean/(m*b)-(m-1)/2.d0
        write(6,*) 'mean delay (analysis, upper bound) is: ',
        * (amean+m*(n-1)*(2-q1)/2)/(m*b)- (m-1)/2.d0 + n-1
        write(10,*) 'mean queue-size (analysis, lower bound) is : ',
        * amean/m-(m-1)*b/2
        write(10,*) 'mean queue-size (analysis, upper bound) is : ',
        * amean/m -(m-1)*b/2 +(n-1)*(2-q1)/2+ (n-1)*b
        write(10,*) 'mean delay (analysis, lower bound) is: ',
        * amean/(m*b)-(m-1)/2.d0
        write(10,*) 'mean delay (analysis, upper bound) is: ',
        * (amean+m*(n-1)*(2-q1)/2)/(m*b)- (m-1)/2.d0 + n-1
126 close(10)
    stop
    end

c
integer function batch(types)
integer seeds,types
real*8 mu,u,u1,u2,rand
common mu,u,u1,u2,seeds
if(types.eq.1) then
    batch=int(u*1.001)
elseif(types.eq.2) then
    batch=igeom(1.0d0/mu,seeds)
else
    batch=u1+int( (u2-u1+1)*rand(seeds))
endif
return
end

```

C -----Random Generator Function-----

```

C This function will take the argument "seed" as an "input" seed,
C   and return the value which is uniformly distributed over [0, 1],
C   and the output seed to be used as the next input seed.
  function rand(seed)
  integer seed
  double precision pp,i7,i2,rand
  i7=7**5
  i2=2**31-1
  pp=i7*seed
  seed=dmod(pp,i2)
  rand=seed/(i2+1)
  return
  end

C
C ***** FUNCTION IGEOM *****
C
C FUNCTION IGEOM PRODUCES A GEOMETRICALLY DISTRIBUTED PSEUDO-RANDOM
C OBSERVATION WITH AVERAGE AVG , FROM R.N. STREAM ISTRM
  FUNCTION IGEOM(AVG,ISTRM)
  REAL*8 THETA,GI,FI,avg,uu,rand
  Uu= RAND(istrm)
  THETA = 1.0d0/AVG
  NN=INT(10.0d0*AVG)
  GI=THETA
  FI=THETA
  I=1
  IF(Uu.LE.FI) GOTO 20
  DO 10 I=2,NN
  GI = (1.-THETA)*GI
  FI = FI + GI
  IF(Uu.LE.FI) GOTO 20
  10 CONTINUE
  C WRITE(*,6000) TNOW,AVG,Uu
  c6000 FORMAT(1X,'!!!! ERROR IN % IGEOM %, TIME = ',F12.4,
  c      */2X,' AVG = ',F12.4,' Uu = ',F8.5)
  20 IGEOM = I
  RETURN
  END

C

c This function will take the value 1 with proba. p and 0 with prob. 1-p
function nb(seed,p)
real*8 t,p,rand
integer seed
t=rand(seed)
if(t.le.p) then
  nb=1
else
  nb=0
endif
return
end

```

Delay-Throughput Evaluator, IRI Corp.

Appendix E. The ITDMA Program: Input and Output Example

ITDMA EXAMPLE 1

** Simulation and Analysis **

Integrated CS/PS System Parameters:

m, n = 10 5
 the mean batch arrival rate p1 1.0000000000000000E-001
 the mean service rate q1 1.0000000000000000
 system throughput rho = 6.522882542734397E-001
 the batch size is ; geometric
 the mean batch size is : 5.0000000000000000

The queue-size probabilities are:

j	P(j)	D(j)
0	.300333333	.000000000
1	.041666667	.000000000
2	.040666667	.049743475
3	.037444444	.046174437
4	.034888889	.041043944
5	.032555556	.040374749
6	.032555556	.040374749
7	.027555556	.040374749
8	.028777778	.036805710
9	.030555556	.038367165
10	.028111111	.033236672
11	.026222222	.030336828
12	.024777778	.030113763
13	.019555556	.028106179
14	.024000000	.027660049
15	.017333333	.027213919
16	.019555556	.024760205
17	.014333333	.023644881
18	.016333333	.024314075
19	.013222222	.022306491
>= 20	.189555556	.395047959

mean system-size (simulation) is : 10.965111111111110
 standard deviation (system-size, simu.) is: 14.562551077216630

mean delay (simulation) is: 22.974124470220830
 standard deviation (delay, simulation) is: 23.174030509119400

P(X> 10)= .36488888889
 P(D> 10)= .63350434977

Blocking probability of a CS session: 7.293294738847554E-002

mean queue-size (up and lower bounds)
 72.280568408166620 4.500000000000000
 mean delay (up and lower bounds)
 144.561136816333200 9.000000000000000

```

C>itdma2
Enter the file_name for data
(no more than 12 characters)
(in PC, QUOTE the 'file_name')
'iout1'
What would you like to do ?
  1: Simulation only
  2: Analysis only
  3: Simulation and analysis
Please enter a number:
3
Enter the start time (when statistic starts)
100
Enter the stop time (total simulation time)
1000

Enter the frame duration (m) ( $m \leq 50$ ) and the maximum
number of slots for CS support (n)
( $n \leq m$ ) m and n should be integers
10 5
Enter the session (circuit) arrival rate
0.01
Enter the session transmission rate
0.04

Enter the packet batch arrival rate [mess./slot] p1
0.1
please wait
average number of CS slots used per frame (V): 2.334677242395972
Enter the batch size distribution index
  1: deterministic
  2: geometric
  3: uniform

2
Enter the mean batch size, b ( $m \cdot b \cdot p1 < m - V$ ):
5
To find the probabilities  $P(X > x)$  and  $P(D > d)$ , please
enter x and d ( $x, d \leq 20$ , both should be integers)
10 10
please wait

```

Delay-Throughput Evaluator, IRI Corp.

Appendix F. The ITDMA Program: Source Code

```

C:          Integrated CS/PS TDMA System
C
C          *****
C          *   Version 1   Sept. 20, 1989   *
C          *   Professor Izhak Rubin       *
C          *   Department of Electrical Engineering   *
C          *   Univeristy of California       *
C          *   Los Angeles, CA 90024       *
C          *****
C
C  This program finds the mean system size and delay for
C  a movable boundary integrated data/vioce system. For
C  the CS subsystem, an M/Geom/N/N queueing model is assumed.
C  While for the PS part, the data transmission rate
C  is assumed one.

C: Descriptions of the variables:
C:   m      : frame duration
C:   n      : maximum number of slot for CS support
C:           n<=m
C:   q(i,j) : transition matrix of the CS connection system
C:   alm    : mean session arrival rate
C:   uc     : mean session transimission rate
C:   pl     : mean packet arrival rate
C:   ql     : mean packet transimssion rate, here we set ql=1
C:   types  : the type of service time distribution
C:   u      : deterministic batch size
C:   u1     : lower limit of the uniformly dist. batch size
C:   u2     : upper limit of the uniformly dist. batch size
C:   seedt  : the seed to generate the number of batch arrivals per slot
C:   seeds  : the seed ... service ...
C:   ic(j)  : total number of packets in the system at the start of jth slot
C:           (j=0,1,2,...,20)
C:   p(j)   : the prob. that there are j packets in the system
C:           at the start of a slot (j=0,1,2,...,20)
C:   qmean  : mean queue-size obtained from simulation
C: External function: rand(seed)
C:           return a value that is uniformly distributed over (0, 1)
C
      implicit real*8(a-h,o-z)
      integer xn(50),types,ict(0:20,1:50),ic(0:20),types1
      integer seed,seeds,icd(0:20,1:50),id(0:20),dn(500)
      dimension a(0:50,0:50),q(0:50,0:50),g(0:50)
      dimension x(0:50),v(0:50),p(0:20),dw(0:20)
      common amu,u,u1,u2,seeds,n,m,nl,ml,seed,istart,istop
      character*12 filen
      nl=50
      ml=50
      mm=20
      print *, 'Enter the file name for data '
      print *, ' (no more than 12 characters) '
      print *, ' (in PC, QUOTE the ''file_name'') '
      read *, filen
      open(10,file=filen)
      ql=1
      print *, 'What would you like to do ? '
      print *, ' 1: Simulation only '
      print *, ' 2: Analysis only '
      print *, ' 3: Simulation and analysis '
      print *, 'Please enter a number:'
      read *, types1
      if(types1.eq.1) then
        write(10,*) '
        write(10,*) '** Simulation **'
        print *, 'Enter the start time (when statistic starts)'
        read *, istart

```

```

        print *, 'Enter the stop time (total simulation time)'
        read *, istop
    elseif(types1.eq.3) then
        write(10,*) ' '
        write(10,*) '** Simulation and Analysis **'
        print *, 'Enter the start time (when statistic starts)'
        read *, istart
        print *, 'Enter the stop time (total simulation time)'
        read *, istop
    elseif (types1.eq.2) then
        write(10,*) ' '
        write(10,*) '**** Analysis ****'
    endif
c
88  print *, ' '
    print *, 'Enter the frame duration (m) (m<=50) and the maximum'
    print *, ' number of slots for CS support (n) '
    print *, ' (n<=m) m and n should be integers'
    read *, m,n
    print *, 'Enter the session (circuit) arrival rate '
    read *, alm
    print *, 'Enter the session transmission rate '
    read *, uc
    print *, 'Enter the packet batch arrival rate [mess./slot] p1 '
    read *, p1
c
    print *, 'please wait '
c  **begin of computing v(i)
    call qmat(q,v,m,n,alm,uc,n1,m1)
    delt = 1.0d-14
    call vi(a,q,n,n1,m1,delt,v)
    sum=0.d0
    do 215 i=1,n
215  sum=sum+i*v(i)
    vbar=sum
    write(6,*) 'average number of CS slots used per frame (V):',vbar
c  ** end of computing sum of v(i)
c
1  print *, 'Enter the batch size distribution index'
    print *, ' 1: deterministic'
    print *, ' 2: geometric'
    print *, ' 3: uniform'
    print *, ' '
    read *, types
    if(types.eq.1) then
        print *, 'Enter the fixed batch size u (m*u*p1<m-V):'
        read *, u
        rho=u*p1*m/(m-sum)
        if (rho.ge.1.d0) then
            print *, ' '
            print *, 'system not stable, please reenter the parameters'
            go to 88
        endif
        b=u*p1
        a2=p1*u*(u-1)
    elseif(types.eq.2) then
        print *, 'Enter the mean batch size, b (m*b*p1<m-V):'
        read *, b
        rho=b*p1*m/(m-sum)
        if (rho.ge.1.d0) then
            print *, ' '
            print *, 'system not stable, please reenter the parameters'
            go to 88
        endif
        amu=1.0d0/b
        b=b*p1

```

```

        a2=p1*2.d0*(1-amu)/amu**2
elseif(types.eq.3) then
    print *, 'Enter the range of the batch size:'
    print *, ' (e.g., "1.0, 3.0") m*(u1+u2)*p1/2(m-V) '
    read *, u1,u2
    rho=(u1+u2)*p1*m/(2*(m-sum))
    if (rho.ge.1.d0) then
        print *, ' '
        print *, 'system not stable, please reenter the parameters'
        go to 88
    endif
    b=p1*(u1+u2)/2.d0
    a2=p1*(u2*(u2+1)*(2*u2+1)/6-u2*(u2+1)/2-(u1-1)*u1*(2*u1-1)
*      /6+u1*(u1-1)/2)/(u2-u1+1)
    else
        go to 1
    endif
c
c ** write the parameters to the output file
write(10,*) ' Integrated CS/PS System Parameters: '
write(10,*) ' m, n = ',m,n
write(10,*) 'the mean batch arrival rate p1 ',p1
write(10,*) 'the mean service rate q1 ',q1
write(10,*) 'system throughput rho = ',rho
if(types.eq.1) then
    write(10,*) 'the batch size is : deterministic '
    write(10,*) 'the fixed batch size is : ',u
elseif(types.eq.2) then
    write(10,*) 'the batch size is ; geometric '
    write(10,*) 'the mean batch size is : ',1/amu
elseif(types.eq.3) then
    write(10,*) 'the batch size is : uniform '
    write(10,*) 'the range of the batch size is: [',u1,',',u2,']'
endif
c
if(types1.eq.2) then
    write(6,*) 'please wait'
    goto 127
endif
print *, 'To find the probabilities P(X>x) and P(D>d), please'
print *, 'enter x and d (x, d<= 20, both should be integers)'
read(5,*) igx,igd
print *, 'please wait'
c ** start simulation
call simu(p1,p,qmean,q,xn,types,ic,ict,var,
*      dn,wmean,dw,vard,icd,id,mm)
c
pd=0
px=0
print *, 'The queue-size probabilities are as follows:'
write(6,*) 'j          P(j)          D(j)          '
print *, '-----'
write(10,*) 'The queue-size probabilities are:'
write(10,*) 'j          P(j)          D(j)          '
write(10,*) '-----'
do 110 i=0,mm
if(i.lt.mm) then
    write(6,200) i,p(i),dw(i)
    write(10,200) i,p(i),dw(i)
else
    write(6,210) i,p(i),dw(i)
    write(10,210) i,p(i),dw(i)
endif
if(i.gt.igx) px=px+p(i)
if(i.gt.igd) pd=pd+dw(i)
110 continue

```

```

200 format(1x,i3,2x,f14.9,2x,f14.9)
210 format(1x,'>=',i3,2x,f14.9,2x,f14.9)
    write(6,*) 'mean system-size (simulation) is : ',qmean
    write(6,*) 'standard deviation (system-size, simu.) is : ',var
    write(6,*) 'mean delay (simulation) is: ',wmean
    write(6,*) 'standard deviation (delay, simulation) is: ',vard
    write(10,*) 'mean system-size (simulation) is : ',qmean
    write(10,*) 'standard deviation (system-size, simu.) is: ',var
    write(10,*) 'mean delay (simulation) is: ',wmean
    write(10,*) 'standard deviation (delay, simulation) is: ',vard
    write(6,50) igx,px
    write(10,50) igx,px
    write(6,301) igd,pd
    write(10,301) igd,pd
50    format(1x,'P(X> ',i3,')= ',f14.11)
301    format(1x,'P(D> ',i3,')= ',f14.11)
    if(types1.eq.1) goto 126
c
c    ** analysis here
c
127    p2=a2
    pd=b
    rho=1-(m-m*pd-vbar)
    npl=n+1
    do 20 i=0,n
        do 21 j=0,n
21            a(i,j)=-q(i,j)+v(j)
20        continue
c
        do 22 i=0,n
            a(i,i)=1+a(i,i)
            a(i,npl)=m*pd-(m-1)+i
22        continue

    call gauss(a,x,n,npl,nl,ml,delt)
c
    do 25 i=0,n
25        g(i)=0
c
        do 26 i=0,n
            do 27 j=0,n
27                g(i)=g(i)+q(i,j)*x(j)
26            continue
c
c        write(6,*) 'g(i) 1 ',(g(i),i=0,n)
        vh=0.0
        do 30 i=0,n
30            vh=vh+v(i)*(m*(1-pd)-i)*g(i)
c
        call smax(g,n,sm,smx)
c
        v2=0.0
        do 29 i=2,n
29            v2=v2+i*(i-1)*v(i)
c
        vh2=m*(m-1)*(pd-1)**2 +m*p2 +2*(m*pd-m+1) *vbar +v2
        sum=0
        do 31 i=0,n
31            sum=sum+(m-i-1)*(m-i)*v(i)
c
        up=m*pd-(m-1) +(m-1)*pd/2 -sum/(2*m) + vh2/(2*(1-rho))
        bl1=up-vh/(1-rho) + sm
        up1=up-vh/(1-rho)+(1-pd)*(m-1)+(m-1)*(1-rho)/m + smx
c
c    ** to find the maximum of two lower bounds
        bl2=pd+p2/(2*(1-pd))

```

```

      if(b12.gt.b11) b11=b12
c  ** to check if it is locally stable
      if(m*pd.lt.m-n) then
        up2=(2**(1-pd)+p2)/2/(1-pd)
        *      +n*(p2-pd**2+pd)/2/(1-pd)/(m-n-m*pd)
        up2=up2+(m-n-1)*n/2/m
        if(up2.lt.up1) up1=up2
      endif
      if(m*pd.gt.m-n) write(6,*) 'not locally stable'
      write(6,*) 'Blocking probability of a CS session: ',v(n)
      write(10,*) 'Blocking probability of a CS session: ',v(n)
      write(6,*) ' mean queue-size (up and lower bounds)'
      write(6,*) up1,' ',b11
      write(6,*) ' mean delay (up and lower bounds)'
      write(6,*) up1/b,' ',b11/b
      write(10,*) ' mean queue-size (up and lower bounds) '
      write(10,*) up1,' ',b11
      write(10,*) ' mean delay (up and lower bounds)'
      write(10,*) up1/b,' ',b11/b
c
126  close(10)
      stop
      end
c
c
      SUBROUTINE smax(x,n,sm,smx)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION x(0:*)
      sm=x(0)
      smx=x(0)
      DO 10 I=1,n
        if (x(i).gt.smx) then
          smx=x(i)
        else
          if (x(i).lt.sm) sm=x(i)
        endif
10    CONTINUE
      RETURN
      END
c
c
      subroutine simu(pl,p,qmean,q,xn,types,ic,ict,var,
*      dn,wmean,dw,vard,icd,id,mm)
      implicit real*8(a-h,o-z)
      integer seed,seeds,seedt,seedt1,seeds,xn(1),ic(0:mm)
      integer ict(0:mm,1:m1),types,batch,bs,dn(1),icd(0:mm,1:m1)
      dimension q(0:n1,0:m1),p(0:mm),dw(0:mm),id(0:mm)
      common amu,u,u1,u2,seeds,n,m,n1,m1,seed,istart,istop
      seed=1099
      seedt=1099
      seedt1=1099
      seeds=99999
      do 14 i=0,mm
      do 15 j=1,m
        ict(i,j)=0
15    icd(i,j)=0
      ic(i)=0
      id(i)=0
14    continue
c
      totalx=0.0d0
      tolx2=0.0d0
      ktd=0
      tdd=0
      tdd2=0
      do 16 i=1,m

```

```

16      xn(i)=0
      do 117 i=1,500
117     dn(i)=0
        ni=0
c
c *  siumlation starts from here
c
      do 20 i=1,istop
c
      nj=nij(seed,n,ni,q,n1,m1)
      ni=nj
      mmni=m-ni
      mmni1=mmni+1
c
      mn=nb(seedt,p1)
      if(xn(m).eq.0) then
        ndn=0
        if(mn.eq.0) then
          xn(1)=0
        else
          bs=batch(types)
          xn(1)=mn*bs
        endif
      else
        ndn=1
        if(mn.eq.0) then
          xn(1)=xn(m)-1
        else
          bs=batch(types)
          xn(1)=xn(m)-1+bs*mn
        endif
      endif
      ll=xn(1)+ndn
      do 149 kd=1,ll
149     dn(kd)=dn(kd)+1
c
      if(ndn.eq.1) then
        kdd=dn(1)
        do 119 kd=1,xn(1)
119     dn(kd)=dn(kd+1)
        dn(xn(1)+1)=0
      endif
c
      if(i.gt.istart) then
        if(xn(1).gt.mm) then
          ict(mm,1)=ict(mm,1)+1
        else
          ict(xn(1),1)=ict(xn(1),1)+1
        endif
        totalx=totalx+xn(1)
        tol2=tol2+xn(1)**2
        if(ndn.eq.1) then
          ktd=ktd+1
          tdd=tdd+kdd
          tdd2=tdd2+kdd**2
          if(kdd.gt.mm) then
            icd(mm,1)=icd(mm,1)+1
          else
            icd(kdd,1)=icd(kdd,1)+1
          endif
        endif
      endif
c
      do 17 j=2,mmni
        mn=nb(seedt,p1)
        if(xn(j-1).eq.0) then

```

```

        ndn=0
        if(mn.eq.0) then
            xn(j)=0
        else
            bs=batch(types)
            xn(j)=mn*bs
        endif
    else
        ndn=1
        if(mn.eq.0) then
            xn(j)=xn(j-1)-1
        else
            bs=batch(types)
            xn(j)= xn(j-1)-1+bs*mn
        endif
    endif
c
    ll=xn(j)+ndn
    do 139 kd=1,ll
139     dn(kd)=dn(kd)+1
c
        if(ndn.eq.1) then
            kdd=dn(1)
            do 129 kd=1,xn(j)
129         dn(kd)=dn(kd+1)
            dn(xn(j)+1)=0
        endif
c
        if(i.gt.istart) then
            if(xn(j).gt.mm) then
                ict(mm,j)=ict(mm,j)+1
            else
                ict(xn(j),j)=ict(xn(j),j)+1
            endif
            totalx=totalx+xn(j)
            tol2=tol2+xn(j)**2
            if(ndn.eq.1) then
                ktd=ktd+1
                tdd=tdd+kdd
                tdd2=tdd2+kdd**2
                if(kdd.gt.mm) then
                    icd(mm,1)=icd(mm,1)+1
                else
                    icd(kdd,1)=icd(kdd,1)+1
                endif
            endif
        endif
c
17     continue
c
        if(mn1.gt.m) goto 20
        do 18 j=mn1,m
            mn=nb(seedt,p1)
            bs=batch(types)
            xn(j)=xn(j-1)+mn*bs
            do 211 kd=1,xn(j)
211         dn(kd)=dn(kd)+1
c
            if(i.gt.istart) then
                if(xn(j).gt.mm) then
                    ict(mm,j)=ict(mm,j)+1
                else
                    ict(xn(j),j)=ict(xn(j),j)+1
                endif
                totalx=totalx+xn(j)
                tol2=tol2+xn(j)**2

```

```

endif
C
18 continue
20 continue
C
last=istop-istart
qmean=totalx/(last*m)
var=dsqrt ( tolx2/(last*m) -qmean**2 )
wmean=tdd/(ktd*1.0d0)
vard=dsqrt( tdd2/(ktd*1.0d0)-wmean**2 )
C
do 22 j=0,mm
do 21 k=1,m
ic(j)=ic(j)+ict(j,k)
id(j)=id(j)+icd(j,k)
21 continue
p(j)=ic(j)*1.d0/(last*m)
dw(j)=id(j)*1.d0/ktd
22 continue
C
return
end
C
C
function batch(types)
integer seeds,types,batch,seed
real*8 amu,u,u1,u2,rand
common amu,u,u1,u2,seeds,n,m,n1,m1,seed,istart,istop
if(types.eq.1) then
batch=int(u*1.001)
elseif(types.eq.2) then
batch=igeom(1.0d0/amu,seeds)
else
batch=int(u1)+int( (u2-u1+1)*rand(seeds))
endif
return
end

C -----Random Generator Function-----
C This function will take the argument "seed" as an "input" seed,
C and return the value which is uniformly distributed over [0, 1],
C and the output seed to be used as the next input seed.
function rand(seed)
integer seed
double precision pp,i7,i2,rand
i7=7**5
i2=2**31-1
pp=i7*seed
seed=dmod(pp,i2)
rand=seed/(i2+1)
return
end

C
C ***** FUNCTION IGEOM *****
C
C FUNCTION IGEOM PRODUCES A GEOMETRICALLY DISTRIBUTED PSEUDO-RANDOM
C OBSERVATION WITH AVERAGE AVG , FROM R.N. STREAM ISTRM
FUNCTION IGEOM(AVG,ISTRM)
REAL*8 THETA,GI,FI,avg,uu,rand
Uu= RAND(istrm)
THETA = 1.0d0/AVG
NN=INT(10.0d0*AVG)
GI=THETA
FI=THETA
I=1

```

```

        IF(Uu.LE.FI) GOTO 20
        DO 10 I=2,NN
        GI = (1.-THETA)*GI
        FI = FI + GI
        IF(Uu.LE.FI) GOTO 20
10    CONTINUE
c    WRITE(*,21) TNOW,AVG,Uu
c21   FORMAT(1X,'!!!! ERROR IN % IGEOM %, TIME = ',F12.4,
c    */2X,' AVG =',F12.4,' Uu = ',F8.5)
20    IGEOM = I
        RETURN
        END

C

c    This function will take the value 1 with proba. p and 0 with prob. 1-p
function nb(seed,pl)
real*8 t,pl,rand
integer seed
t=rand(seed)
if(t.le.pl) then
    nb=1
else
    nb=0
endif
return
end

c
function nij(seed,nc,i,q,nl,ml)
implicit real*8(a-h,o-z)
dimension q(0:nl,0:ml)
integer seed
t=rand(seed)
sum=0.0
do 20 j=0,nc
    sum=sum+q(i,j)
    if( t.le.sum ) then
        nij=j
        goto 15
    endif
20    continue
15    return
end

c
c
c    subroutine vi(a,q,n,nl,ml,delt,v)
c    this subroutine finds the steady probability v(i)
implicit real*8(a-h,o-z)
dimension a(0:nl,0:ml),q(0:nl,0:ml),v(0:*)
c
    npl=n+1
    Do 10 i=0,npl
        a(0,i)=1.d0
    do 15 i=1,n
        do 15 j=0,n
            a(i,j)--q(j,i)
        do 16 i=1,n
            a(i,i)=1.d0+a(i,i)
16        a(i,npl)=0.d0
c
        call gauss(a,v,n,npl,nl,ml,delt)
        return
        end

        subroutine qmat(q,x,m,n,alm,uc,nl,ml)

```

```

        implicit real*8(a-h,o-z)
        dimension q(0:n1,0:m1),x(0:*)
c
        do 10 k=0,n
10      x(k)=(m*alm)**k*dexp(-m*alm) /ifac(k)
c
        nml=n-1
        do 12 i=0,n
          do 11 j=0,nml
11      q(i,j)=0.d0
          q(i,n)=1.d0
12      continue
c
        do 15 i=0,n
          do 16 j=0,nml
            km=max(0,i-j)
            do 17 k=km,i
17      q(i,j)=q(i,j)+cnk(i,k)*uc**k*(1-uc)**(i-k)*x(j-(i-k))
16      continue
15      continue
c
        do 20 i=0,n
          do 25 k=0,nml
25      q(i,n)=q(i,n)-q(i,k)
20      continue
        return
        end
c
        function ifac(k)
        if(k.eq.0) then
          ifac=1
        else
          i=1
          do 10 j=1,k
10      i=i*j
          ifac=i
        endif
        return
        end
c
        function cnk(n,k)
        implicit real*8(a-h,o-z)
        cnk=(1.d0*ifac(n))/(ifac(k)*ifac(n-k))
        return
        end
C
SUBROUTINE GAUSS(A,X,N,NP1,N1,M1,DELT)
C  THIS ALGORITHM SOLVES THE LINEAR SYSTEM EQUATION
C  USING THE GAUSS-ELIMINATION METHOD WITH PARTIAL PIVOTING
C  IT ALSO FINDS THE DETERMINENT OF A
C  DOUBLE PRECISION IS USED IN HTIS SUBROUTINE
C  SUBROUTINE PVT IS USED TO INTERCHANGE ROWS IF NECESSARY
C  SUBROUTINE UX IS CALLED TO FIND THE SOLUTION
C  IF THE ERROR IS NEEDED, THEN CALL SUBROUTINE ERROR
C
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION A(0:n1,0:m1),X(0:*)
C
NM1=N-1
DO 15 I=0,NM1
  IP1=I+1
  CALL PVTGUS(A,N,NP1,IP,I,N1,M1)
C
  IF ( DABS(A(I,I)) .LT. DELT ) GOTO 20
C
DO 61 K=IP1,N

```

```

        R1=A(K,I)/A(I,I)
        DO 62 J=IP1,NP1
62      A(K,J)=A(K,J)-R1*A(I,J)
61      CONTINUE
C
15      CONTINUE
C
      IF ( DABS(A(N,N)) .LT. DELT ) GOTO 20
      CALL UX(A,N,NP1,X,N1,M1)
C
      GO TO 103
C
20      WRITE (6, *) ' THE MATRIX IS SINGULAR '
103     RETURN
      END
C
C
      SUBROUTINE PVTGUS(A,N,NP1,IP,J,N1,M1)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(0:n1,0:m1)
      IP=J
      BIG=DABS(A(J,J))
      DO 52 I=J,N
        IF ( BIG .LT. DABS(A(I,J)) ) THEN
          BIG=DABS(A(I,J))
          IP=I
        ENDIF
52      CONTINUE
      IF (IP.EQ.J) RETURN
      DO 61 K=0,NP1
        SAV=A(J,K)
        A(J,K)=A(IP,K)
        A(IP,K)=SAV
61      CONTINUE
      RETURN
      END
C
C
C
C
      SUBROUTINE UX(A,N,NP1,X,N1,M1)
      IMPLICIT REAL*8 (A-H,O-Z)
      THIS SUBROUTINE USES BACKWARD SUBSTITUTION TO FIND
      THE SOLUTION OF LINEAR SYSTEM EQUATION
      A*X=B, USING THE MATRIX COMPUTED BY GUASS OR GAUSSP
      OR GAUSSM
      DIMENSION A(0:n1,0:m1),X(0:*)
C
      X(N)=A(N,NP1)/A(N,N)
      DO 70 I=1,N
        IN=N-I
        SUM=0.0
        IN1=IN+1
        DO 75 J=IN1,N
75          SUM=SUM+A(IN,J)*X(J)
          X(IN)=(A(IN,NP1)-SUM)/A(IN,IN)
70      CONTINUE
C
      RETURN
      END

```

Appendix G.

The FDDI Program: Symmetric Case Example

89/10/02
17:41:36

fddi.1.inp

1

** Simulation of a Timed Token Rotation Protocol (FDDI-I Type)
program: FDDI
Professor Izhak Rubin, UCLA

Enter Select Input Mode

1. from KEYBOARD
2. from DATA FILE

1

Enter the output data file name

ff7.2

Enter Feature Selection

1. Symmetric System
2. 2 Classes of Stations
3. Different Loading from Station to Station

1

Enter the statistics collection start time (msec)

4000

Enter the stop time (msec)

10000

Enter w,d,x, for computing $P(W>w)$, $P(D>d)$, $P(X>x)$

1 1 1

Enter the number of stations (N)

6

Enter the walk time from a station to its neighboring station (in msec; r)

0.1

Enter Target Token Rotation Time (msec; TTRT)

200

Synchronous traffic arrival rate (packets/msec/station; as)

0.001

Mean synchronous packet transmission time (msec; plens)

0.05

Enter the bandwidth time (msec; $< (TTRT - N \cdot r) / N$)

(the max synchronous message transmission per visit)

0.

Enter number of message priority classes per station (Np)

3

Enter priority-1 threshold (msec; $T_{pri}(1)$)

100

Enter priority-1 arrival rate (aa; packets/msec/station)

0.1

Enter priority-1 mean transmission time (msec; plena)

0.3

Enter priority-2 threshold (msec; $T_{pri}(2)$)

76.5

Enter priority-2 arrival rate (aa; packets/msec/station)

0.1

Enter priority-2 mean transmission time (msec; plena)

0.3

Enter priority-3 threshold (msec; $T_{pri}(3)$)

56.2

Enter priority-3 arrival rate (aa; packets/msec/station)

0.1

Enter priority-3 mean transmission time (msec; plena)

0.3

89/10/04
11:04:47

fddi.inpl

1

This is an example of the input file for the symmetric system.

ff7.2	output file
1	ifeat
4000	c1
10000	c2
1 1 1	w d x
6	N
0.1	r
200	TTRT
0.001	as
0.05	plens
0.	BWT
3	Np
100.	T_pri(1)
0.1	aa(1)
0.3	plena(1)
76.5	T_pri(2)
0.1	aa(2)
0.3	plena(2)
56.2	T_pri(3)
0.1	aa(3)
0.3	plena(3)

89/10/04
10:24:37

fddi.1.out

.. Performance of a Timed Token Rotation Protocol (FDDI-type) Ring Networks ..

Feature Selected: Symmetric Systems

Statistics Start (msec): 2000.0

Statistics Stop (msec): 20000.0

TTRT (msec): 200.000

Number of Stations (N): 6

Number of Priorities (Np): 3

Max Throughput (1-walktime/2TTRT): 0.9985

Normalized Throughput (specified): 0.5700

Normalized Throughput (realized): 0.5565

Realized Mean Cycle Time (msec): 1.3530

Realized Mean Dwell Time (msec): 0.7530

Max Cycle Time (msec): 11.8001 at 6518th cycle, t= 8747.4

Walk Time (msec): 0.1000

Bandwidth Time (BWT:msec): 30.0000

Arrival Rate for Synchronous Traffic (packets/msec/station): 0.1000

Mean Packet Length for Synchronous Traffic (msec): 0.0500

Asynchronous Traffic: Priority-j Tpri aa(l,j) plena(l,j)

1 100.0000 0.1000 0.3000

2 76.5000 0.1000 0.3000

3 56.2000 0.1000 0.3000

Token goes 14801 cycles in simulation

Sync Traffic:	E(X)	sigma(X)	E(W)	sigma(W)	E(D)	sigma(D)
Async: pri-1	0.1187	0.3559	0.7789	0.7288	0.8285	0.7288
Async: pri-2	0.1185	0.3517	0.8094	0.7391	1.1077	0.7604
Async: pri-3	0.1215	0.3568	0.8537	0.7685	1.1530	0.7867
Station stat	0.1203	0.3543	0.9152	0.8217	1.2174	0.8389

Station	stat	E(X)	sigma(X)	E(W)	sig (W)	E(D)	sigma(D)
1	0	1704	0.1176	0.3488	0.7861	0.7220	0.8355
1	1	1632	0.1118	0.3397	0.7799	0.7174	0.7395
2	1798	0.1211	0.3528	0.8330	0.7491	1.1293	0.7493
3	1773	0.1200	0.3489	0.8794	0.7686	1.1802	0.7851
2	0	1706	0.1157	0.3491	0.7673	0.7332	0.7333
1	1765	0.1211	0.3575	0.8387	0.7691	1.1308	0.7941
2	1799	0.1232	0.3607	0.8854	0.8171	1.1829	0.8338
3	1747	0.1178	0.3524	0.8871	0.8094	1.1860	0.8282
3	0	1749	0.1191	0.3508	0.7519	0.7176	0.7182
1	1745	0.1195	0.3515	0.8029	0.7177	1.0979	0.7314
2	1798	0.1211	0.3528	0.8330	0.7491	1.1293	0.7618
3	1773	0.1200	0.3489	0.8794	0.7686	1.1802	0.7851
4	0	1730	0.1194	0.3519	0.8064	0.7397	0.7396
1	1741	0.1197	0.3509	0.8157	0.7273	1.1156	0.7499
2	1802	0.1231	0.3602	0.8566	0.7815	1.1506	0.8053
3	1795	0.1227	0.3619	0.9482	0.8421	1.2515	0.8421
5	0	1784	0.1213	0.3541	0.7861	0.7443	0.7442
1	1750	0.1197	0.3528	0.7984	0.7378	1.1026	0.7634
2	1736	0.1192	0.3514	0.8527	0.7544	1.1503	0.7677
3	1761	0.1191	0.3505	0.9224	0.8443	1.2247	0.8601
6	0	1762	0.1190	0.3505	0.7755	0.7143	0.7148
1	1743	0.1191	0.3547	0.8210	0.7623	1.1240	0.7807
2	1795	0.1228	0.3626	0.8675	0.7763	1.1759	0.7977
3	1816	0.1231	0.3552	0.9066	0.8460	1.2080	0.8653

59/10/04
10:24:37

fdi.1.out

2

16 0. 0. 1.3085 0.82394 1.3085 0.80810
17 0. 0. 1.3854 0.84038 1.3854 0.82981
18 0. 0. 1.4624 0.85329 1.4624 0.84331
19 0. 0. 1.5394 0.86854 1.5394 0.86268
20 0. 0. 1.6164 0.87852 1.6164 0.87265
21 0. 0. 1.6933 0.89026 1.6933 0.88263
22 0. 0. 1.7703 0.90258 1.7703 0.89554
23 0. 0. 1.8473 0.91256 1.8473 0.90493
24 0. 0. 1.9242 0.91901 1.9242 0.91549
25 0. 0. 2.0012 0.92606 2.0012 0.92136
26 0. 0. 2.0782 0.93251 2.0782 0.93075
27 0. 0. 2.1551 0.93897 2.1551 0.93603
28 0. 0. 2.2321 0.94425 2.2321 0.93955
29 0. 0. 2.3091 0.95246 2.3091 0.94836
30 0. 0. 2.3860 1.00000 2.3860 1.00000

Async: priority-1 Normalized throughput (Rho_a) = 0.030000

J P(X=j) P(Xa=j) P(Xd=j) P(Xt=j) W P(W<=w) d P(D<=d)
0 0.89567 0.91115 0.91115 0.90844 0.0770 0.07782 0.0770 0.00490
1 0.09734 0.08272 0.08272 0.08599 0.1539 0.15564 0.1539 0.01838
2 0.00646 0.00613 0.00613 0.00519 0.2309 0.21998 0.2309 0.04841
3 0.00053 0. 0.00039 0. 0.3079 0.28860 0.3079 0.07904
4 0. 0. 0. 0. 0.3848 0.34926 0.3848 0.11949
5 0. 0. 0. 0. 0.4618 0.41299 0.4618 0.16728
6 0. 0. 0. 0. 0.5388 0.46752 0.5388 0.22181
7 0. 0. 0. 0. 0.6158 0.53002 0.6158 0.28738
8 0. 0. 0. 0. 0.6927 0.57537 0.6927 0.35294
9 0. 0. 0. 0. 0.7697 0.61458 0.7697 0.41728
10 0. 0. 0. 0. 0.8467 0.64767 0.8467 0.48039
11 0. 0. 0. 0. 0.9236 0.68627 0.9236 0.52451
12 0. 0. 0. 0. 1.0006 0.71875 1.0006 0.56863
13 0. 0. 0. 0. 1.0776 0.74939 1.0776 0.61275
14 0. 0. 0. 0. 1.1545 0.77267 1.1545 0.65502
15 0. 0. 0. 0. 1.2315 0.79473 1.2315 0.68137
16 0. 0. 0. 0. 1.3085 0.81679 1.3085 0.71140
17 0. 0. 0. 0. 1.3854 0.83578 1.3854 0.74326
18 0. 0. 0. 0. 1.4624 0.85294 1.4624 0.76654
19 0. 0. 0. 0. 1.5394 0.87132 1.5394 0.78799
20 0. 0. 0. 0. 1.6164 0.88787 1.6164 0.80882
21 0. 0. 0. 0. 1.6933 0.89706 1.6933 0.82537
22 0. 0. 0. 0. 1.7703 0.91115 1.7703 0.84117
23 0. 0. 0. 0. 1.8473 0.92463 1.8473 0.87010
24 0. 0. 0. 0. 1.9242 0.93137 1.9242 0.88419
25 0. 0. 0. 0. 2.0012 0.93750 2.0012 0.89951
26 0. 0. 0. 0. 2.0782 0.94424 2.0782 0.91115
27 0. 0. 0. 0. 2.1551 0.94975 2.1551 0.91850
28 0. 0. 0. 0. 2.2321 0.95466 2.2321 0.92831
29 0. 0. 0. 0. 2.3091 0.95895 2.3091 0.93321
30 0. 0. 0. 0. 2.3860 1.00000 2.3860 1.00000

Async: priority-2 Normalized throughput (Rho_a) = 0.030000

J P(X=j) P(Xa=j) P(Xd=j) P(Xt=j) W P(W<=w) d P(D<=d)
1 0.88928 0.89594 0.89594 0.89839 0.0770 0.06804 0.0770 0.00515
0 0.10215 0.09891 0.09891 0.09385 0.1539 0.14122 0.1539 0.01601
2 0.00812 0.00515 0.00515 0.00741 0.2309 0.20354 0.2309 0.03945
3 0.00045 0. 0.00034 0. 0.3079 0.25672 0.3079 0.06575
4 0. 0. 0. 0. 0.3848 0.31447 0.3848 0.10921
5 0. 0. 0. 0. 0.4618 0.37107 0.4618 0.15323
6 0. 0. 0. 0. 0.5388 0.43625 0.5388 0.20469
7 0. 0. 0. 0. 0.6158 0.49971 0.6158 0.26129
8 0. 0. 0. 0. 0.6927 0.54031 0.6927 0.31332
9 0. 0. 0. 0. 0.7697 0.58491 0.7697 0.37278

Async: priority-3 Normalized throughput (Rho_a) = 0.030000

J P(X=j) P(Xa=j) P(Xd=j) P(Xt=j) W P(W<=w) d P(D<=d)
0 0.89191 0.87798 0.87798 0.89122 0.0770 0.06217 0.0770 0.00465
1 0.09832 0.11098 0.11098 0.09856 0.1539 0.11040 0.1539 0.01627
2 0.00894 0.01104 0.01104 0.00935 0.2309 0.16734 0.2309 0.03370
3 0.00083 0. 0. 0.00086 0.3079 0.21964 0.3079 0.05520
4 0. 0. 0. 0. 0.3848 0.26961 0.3848 0.09006
5 0. 0. 0. 0. 0.4618 0.31784 0.4618 0.12377
6 0. 0. 0. 0. 0.5388 0.39338 0.5388 0.16851
7 0. 0. 0. 0. 0.6158 0.44741 0.6158 0.21964
8 0. 0. 0. 0. 0.6927 0.49506 0.6927 0.26845
9 0. 0. 0. 0. 0.7697 0.53632 0.7697 0.32016
10 0. 0. 0. 0. 0.8467 0.57292 0.8467 0.37478
11 0. 0. 0. 0. 0.9236 0.60895 0.9236 0.43289
12 0. 0. 0. 0. 1.0006 0.63800 1.0006 0.47356
13 0. 0. 0. 0. 1.0776 0.66938 1.0776 0.52337
14 0. 0. 0. 0. 1.1545 0.69611 1.1545 0.56363
15 0. 0. 0. 0. 1.2315 0.71761 1.2315 0.59965
16 0. 0. 0. 0. 1.3085 0.74782 1.3085 0.62464
17 0. 0. 0. 0. 1.3854 0.76583 1.3854 0.65950
18 0. 0. 0. 0. 1.4624 0.78850 1.4624 0.68390
19 0. 0. 0. 0. 1.5394 0.80535 1.5394 0.71877
20 0. 0. 0. 0. 1.6164 0.81697 1.6164 0.74317
21 0. 0. 0. 0. 1.6933 0.83091 1.6933 0.76293
22 0. 0. 0. 0. 1.7703 0.84660 1.7703 0.78559
23 0. 0. 0. 0. 1.8473 0.86229 1.8473 0.80186
24 0. 0. 0. 0. 1.9242 0.87682 1.9242 0.82510
25 0. 0. 0. 0. 2.0012 0.88611 2.0012 0.83498
26 0. 0. 0. 0. 2.0782 0.90122 2.0782 0.84951
27 0. 0. 0. 0. 2.1551 0.90819 2.1551 0.86345
28 0. 0. 0. 0. 2.2321 0.91865 2.2321 0.87449
29 0. 0. 0. 0. 2.3091 0.92330 2.3091 0.88960
30 0. 0. 0. 0. 2.3860 1.00000 2.3860 1.00000

** Station 2

Synchronous Traffic: Normalized throughput (Rho_s) = 0.005000

J P(X=j) P(Xa=j) P(Xd=j) P(Xt=j) W P(W<=w) d P(D<=d)
0 0.89364 0.92263 0.92263 0.92833 0.0770 0.07503 0.0770 0.01814
1 0.09741 0.07327 0.07327 0.06614 0.1539 0.15944 0.1539 0.10317

fddi.1.out[illegible]

Async: priority-1 Normalized throughput (Rho a) = 0.030000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xc=j)	w	P(W<w)	d	P(D<d)
0	0.00876	0.90085	0.90085	0.00829	0.0770	0.06459	0.0770	0.00453
1	0.10222	0.09008	0.09008	0.09338	0.1539	0.13088	0.1539	0.02266
2	0.00819	0.00907	0.00907	0.00747	0.2309	0.19830	0.2309	0.04419
3	0.00083	0.	0.	0.00086	0.3079	0.27479	0.3079	0.07365
4	0.	0.	0.	0.	0.3848	0.34108	0.3848	0.11501
5	0.	0.	0.	0.	0.4618	0.40037	0.4618	0.16487
6	0.	0.	0.	0.	0.5388	0.45949	0.5388	0.21983
7	0.	0.	0.	0.	0.6158	0.51728	0.6158	0.27819
8	0.	0.	0.	0.	0.6927	0.55241	0.6927	0.34788
9	0.	0.	0.	0.	0.7697	0.58754	0.7697	0.40793
10	0.	0.	0.	0.	0.8467	0.62210	0.8467	0.46119
11	0.	0.	0.	0.	0.9236	0.65156	0.9236	0.50198
12	0.	0.	0.	0.	1.0006	0.68725	1.0006	0.55354
13	0.	0.	0.	0.	1.0776	0.71615	1.0776	0.58640
14	0.	0.	0.	0.	1.1545	0.74618	1.1545	0.62266
15	0.	0.	0.	0.	1.2315	0.77054	1.2315	0.65552
16	0.	0.	0.	0.	1.3085	0.78697	1.3085	0.68725
17	0.	0.	0.	0.	1.3854	0.80397	1.3854	0.71615
18	0.	0.	0.	0.	1.4624	0.82436	1.4624	0.73711
19	0.	0.	0.	0.	1.5394	0.84306	1.5394	0.75751
20	0.	0.	0.	0.	1.6164	0.85552	1.6164	0.78300
21	0.	0.	0.	0.	1.6933	0.87365	1.6933	0.80283
22	0.	0.	0.	0.	1.7703	0.89122	1.7703	0.81870
23	0.	0.	0.	0.	1.8473	0.90368	1.8473	0.84079
24	0.	0.	0.	0.	1.9242	0.91388	1.9242	0.85779
25	0.	0.	0.	0.	2.0012	0.92068	2.0012	0.87139
26	0.	0.	0.	0.	2.0782	0.92861	2.0782	0.88839
27	0.	0.	0.	0.	2.1551	0.93541	2.1551	0.89972
28	0.	0.	0.	0.	2.2321	0.94164	2.2321	0.91105
29	0.	0.	0.	0.	2.3091	0.94674	2.3091	0.92238
30	0.	0.	0.	0.	2.3860	1.00000	2.3860	1.00000

114

89/10/04
10:24:57

4

fdi.1.out

** Station 3

Synchronous Traffic: Normalized throughput (Rho_a) = 0.005000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
0	0.88928	0.92910	0.92910	0.92711	0.0770	0.08348	0.0770	0.03945
1	0.10305	0.06461	0.06461	0.06827	0.1539	0.16066	0.1539	0.11264
2	0.00707	0.00572	0.00572	0.00425	0.2309	0.23842	0.2309	0.18239
3	0.00053	0.00057	0.00057	0.00036	0.3079	0.32247	0.3079	0.26358
4	0.00008	0.	0.	0.00001	0.3848	0.38536	0.3848	0.35163
5	0.	0.	0.	0.	0.4618	0.44368	0.4618	0.41052
6	0.	0.	0.	0.	0.5388	0.50257	0.5388	0.46598
7	0.	0.	0.	0.	0.6158	0.56146	0.6158	0.52087
8	0.	0.	0.	0.	0.6927	0.59863	0.6927	0.57233
9	0.	0.	0.	0.	0.7697	0.63636	0.7697	0.61063
10	0.	0.	0.	0.	0.8467	0.67410	0.8467	0.64951
11	0.	0.	0.	0.	0.9236	0.70383	0.9236	0.68268
12	0.	0.	0.	0.	1.0006	0.73528	1.0006	0.71927
13	0.	0.	0.	0.	1.0776	0.76101	1.0776	0.74328
14	0.	0.	0.	0.	1.1545	0.78959	1.1545	0.76958
15	0.	0.	0.	0.	1.2315	0.81018	1.2315	0.79817
16	0.	0.	0.	0.	1.3085	0.83248	1.3085	0.82161
17	0.	0.	0.	0.	1.3854	0.84963	1.3854	0.84162
18	0.	0.	0.	0.	1.4624	0.86449	1.4624	0.85706
19	0.	0.	0.	0.	1.5394	0.87593	1.5394	0.86907
20	0.	0.	0.	0.	1.6164	0.88736	1.6164	0.87879
21	0.	0.	0.	0.	1.6933	0.89537	1.6933	0.89251
22	0.	0.	0.	0.	1.7703	0.90795	1.7703	0.89994
23	0.	0.	0.	0.	1.8473	0.91309	1.8473	0.90795
24	0.	0.	0.	0.	1.9242	0.92053	1.9242	0.91710
25	0.	0.	0.	0.	2.0012	0.92739	2.0012	0.92453
26	0.	0.	0.	0.	2.0782	0.93368	2.0782	0.93025
27	0.	0.	0.	0.	2.1551	0.94626	2.1551	0.93939
28	0.	0.	0.	0.	2.2321	0.95197	2.2321	0.94797
29	0.	0.	0.	0.	2.3091	0.95826	2.3091	0.95540
30	0.	0.	0.	0.	2.3860	1.00000	2.3860	1.00000

Async: priority-1 Normalized throughput (Rho_a) = 0.030000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
0	0.88928	0.89685	0.89685	0.90145	0.0770	0.06991	0.0770	0.00630
1	0.10230	0.09685	0.09685	0.09098	0.1539	0.13926	0.1539	0.01891
2	0.00804	0.00630	0.00630	0.00719	0.2309	0.20688	0.2309	0.03725
3	0.00038	0.	0.	0.00037	0.3079	0.27622	0.3079	0.06533
4	0.	0.	0.	0.	0.3848	0.34040	0.3848	0.10888
5	0.	0.	0.	0.	0.4618	0.40172	0.4618	0.14900
6	0.	0.	0.	0.	0.5388	0.46246	0.5388	0.20974
7	0.	0.	0.	0.	0.6158	0.51920	0.6158	0.27908
8	0.	0.	0.	0.	0.6927	0.56390	0.6927	0.33295
9	0.	0.	0.	0.	0.7697	0.60172	0.7697	0.39713
10	0.	0.	0.	0.	0.8467	0.63324	0.8467	0.45387
11	0.	0.	0.	0.	0.9236	0.67106	0.9236	0.51117
12	0.	0.	0.	0.	1.0006	0.70774	1.0006	0.56160
13	0.	0.	0.	0.	1.0776	0.73983	1.0776	0.60688
14	0.	0.	0.	0.	1.1545	0.77020	1.1545	0.64527
15	0.	0.	0.	0.	1.2315	0.79255	1.2315	0.67622
16	0.	0.	0.	0.	1.3085	0.80745	1.3085	0.70716

Async: priority-2 Normalized throughput (Rho_a) = 0.030000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
0	0.88740	0.89433	0.89433	0.89481	0.0770	0.07786	0.0770	0.00501
1	0.10463	0.09844	0.09844	0.09806	0.1539	0.14016	0.1539	0.01780
2	0.00744	0.00723	0.00723	0.00678	0.2309	0.20912	0.2309	0.03838
3	0.00053	0.	0.	0.00035	0.3079	0.26696	0.3079	0.06174
4	0.	0.	0.	0.	0.3848	0.33092	0.3848	0.10067
5	0.	0.	0.	0.	0.4618	0.39210	0.4618	0.14405
6	0.	0.	0.	0.	0.5388	0.44828	0.5388	0.20356
7	0.	0.	0.	0.	0.6158	0.49944	0.6158	0.26863
8	0.	0.	0.	0.	0.6927	0.54338	0.6927	0.33148
9	0.	0.	0.	0.	0.7697	0.58788	0.7697	0.39321
10	0.	0.	0.	0.	0.8467	0.62013	0.8467	0.44716
11	0.	0.	0.	0.	0.9236	0.65684	0.9236	0.49944
12	0.	0.	0.	0.	1.0006	0.69410	1.0006	0.54060
13	0.	0.	0.	0.	1.0776	0.72191	1.0776	0.59121
14	0.	0.	0.	0.	1.1545	0.74638	1.1545	0.62681
15	0.	0.	0.	0.	1.2315	0.77030	1.2315	0.66296
16	0.	0.	0.	0.	1.3085	0.79199	1.3085	0.69522
17	0.	0.	0.	0.	1.3854	0.80979	1.3854	0.72080
18	0.	0.	0.	0.	1.4624	0.82536	1.4624	0.74972
19	0.	0.	0.	0.	1.5394	0.84093	1.5394	0.77197
20	0.	0.	0.	0.	1.6164	0.85428	1.6164	0.79366
21	0.	0.	0.	0.	1.6933	0.86707	1.6933	0.81034
22	0.	0.	0.	0.	1.7703	0.88098	1.7703	0.82369
23	0.	0.	0.	0.	1.8473	0.89600	1.8473	0.83648
24	0.	0.	0.	0.	1.9242	0.90489	1.9242	0.84761
25	0.	0.	0.	0.	2.0012	0.91935	2.0012	0.86429
26	0.	0.	0.	0.	2.0782	0.92547	2.0782	0.88209
27	0.	0.	0.	0.	2.1551	0.93215	2.1551	0.89544
28	0.	0.	0.	0.	2.2321	0.94049	2.2321	0.90656
29	0.	0.	0.	0.	2.3091	0.94828	2.3091	0.91546
30	0.	0.	0.	0.	2.3860	1.00000	2.3860	1.00000

Async: priority-3 Normalized throughput (Rho_a) = 0.030000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
0	0.88770	0.90412	0.90412	0.89119	0.0770	0.06261	0.0770	0.00338
1	0.10486	0.08855	0.08855	0.10156	0.1539	0.11506	0.1539	0.01918
2	0.00714	0.00677	0.00677	0.00690	0.2309	0.16582	0.2309	0.01384
3	0.00030	0.00036	0.00036	0.00031	0.3079	0.22730	0.3079	0.01979
4	0.	0.	0.	0.00004	0.3848	0.28314	0.3848	0.01855
5	0.	0.	0.	0.	0.4618	0.33503	0.4618	0.11465
6	0.	0.	0.	0.	0.5388	0.39312	0.5388	0.11582
7	0.	0.	0.	0.	0.6158	0.45198	0.6158	0.21125
8	0.	0.	0.	0.	0.6927	0.51100	0.6927	0.21103
9	0.	0.	0.	0.	0.7697	0.55556	0.7697	0.31025
10	0.	0.	0.	0.	0.8467	0.59447	0.8467	0.41158

fddi.1.out

11	0.	0.	0.	0.9236	0.63508	0.9236	0.44670
12	0.	0.	0.	1.0006	0.66949	1.0006	0.50423
13	0.	0.	0.	1.0776	0.70276	1.0776	0.55386
14	0.	0.	0.	1.1545	0.73660	1.1545	0.59616
15	0.	0.	0.	1.2315	0.76593	1.2315	0.63508
16	0.	0.	0.	1.3085	0.78793	1.3085	0.66159
17	0.	0.	0.	1.3854	0.80880	1.3854	0.69374
18	0.	0.	0.	1.4624	0.82459	1.4624	0.72984
19	0.	0.	0.	1.5394	0.84320	1.5394	0.75014
20	0.	0.	0.	1.6164	0.86069	1.6164	0.77778
21	0.	0.	0.	1.6933	0.87310	1.6933	0.80259
22	0.	0.	0.	1.7703	0.88832	1.7703	0.82290
23	0.	0.	0.	1.8473	0.89961	1.8473	0.84320
24	0.	0.	0.	1.9242	0.91314	1.9242	0.85505
25	0.	0.	0.	2.0012	0.92442	2.0012	0.87253
26	0.	0.	0.	2.0782	0.93006	2.0782	0.88607
27	0.	0.	0.	2.1551	0.93739	2.1551	0.89622
28	0.	0.	0.	2.2321	0.94360	2.2321	0.90694
29	0.	0.	0.	2.3091	0.95262	2.3091	0.91935
30	0.	0.	0.	2.3860	1.00000	2.3860	1.00000

** Station 4

Synchronous Traffic: Normalized throughput (Rho_s)= 0.005000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	W	P(W<=w)	d	P(D<=d)
0	0.08943	0.92486	0.92486	0.92292	0.0770	0.07514	0.0770	0.02832
1	0.10215	0.07168	0.07168	0.07204	0.1539	0.13642	0.1539	0.10173
2	0.00797	0.00347	0.00347	0.00481	0.2309	0.20405	0.2309	0.15780
3	0.00045	0.	0.	0.00023	0.3079	0.27399	0.3079	0.22543
4	0.	0.	0.	0.	0.3848	0.33584	0.3848	0.29653
5	0.	0.	0.	0.	0.4618	0.40520	0.4618	0.36358
6	0.	0.	0.	0.	0.5388	0.46243	0.5388	0.42832
7	0.	0.	0.	0.	0.6158	0.52139	0.6158	0.48786
8	0.	0.	0.	0.	0.6927	0.56705	0.6927	0.53815
9	0.	0.	0.	0.	0.7697	0.60925	0.7697	0.58092
10	0.	0.	0.	0.	0.8467	0.64393	0.8467	0.62081
11	0.	0.	0.	0.	0.9236	0.68497	0.9236	0.65838
12	0.	0.	0.	0.	1.0006	0.71618	1.0006	0.69653
13	0.	0.	0.	0.	1.0776	0.74277	1.0776	0.72543
14	0.	0.	0.	0.	1.1545	0.76243	1.1545	0.74971
15	0.	0.	0.	0.	1.2315	0.78671	1.2315	0.76936
16	0.	0.	0.	0.	1.3085	0.80867	1.3085	0.79249
17	0.	0.	0.	0.	1.3854	0.82543	1.3854	0.81445
18	0.	0.	0.	0.	1.4624	0.84277	1.4624	0.83064
19	0.	0.	0.	0.	1.5394	0.85723	1.5394	0.84682
20	0.	0.	0.	0.	1.6164	0.86821	1.6164	0.86243
21	0.	0.	0.	0.	1.6933	0.87919	1.6933	0.87225
22	0.	0.	0.	0.	1.7703	0.89017	1.7703	0.88324
23	0.	0.	0.	0.	1.8473	0.90405	1.8473	0.89364
24	0.	0.	0.	0.	1.9242	0.91040	1.9242	0.90694
25	0.	0.	0.	0.	2.0012	0.91908	2.0012	0.91387
26	0.	0.	0.	0.	2.0782	0.93006	2.0782	0.92139
27	0.	0.	0.	0.	2.1551	0.93815	2.1551	0.93353
28	0.	0.	0.	0.	2.2321	0.94393	2.2321	0.94104
29	0.	0.	0.	0.	2.3091	0.94798	2.3091	0.94682
30	0.	0.	0.	0.	2.3860	1.00000	2.3860	1.00000

Async: priority-1 Normalized throughput (Rho_a)= 0.030000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	W	P(W<=w)	d	P(D<=d)
0	0.08868	0.90580	0.90580	0.89899	0.0770	0.06376	0.0770	0.00460
1	0.10335	0.08845	0.08845	0.09420	0.1539	0.13498	0.1539	0.01493
2	0.00752	0.00517	0.00517	0.00654	0.2309	0.20161	0.2309	0.03963

3	0.00045	0.00057	0.00057	0.00027	0.3079	0.27168	0.3079	0.07065
4	0.	0.	0.	0.00000	0.3848	0.33372	0.3848	0.10052
5	0.	0.	0.	0.	0.4618	0.39403	0.4618	0.15106
6	0.	0.	0.	0.	0.5388	0.46065	0.5388	0.21252
7	0.	0.	0.	0.	0.6158	0.50661	0.6158	0.27398
8	0.	0.	0.	0.	0.6927	0.54222	0.6927	0.33544
9	0.	0.	0.	0.	0.7697	0.57783	0.7697	0.40149
10	0.	0.	0.	0.	0.8467	0.62091	0.8467	0.45261
11	0.	0.	0.	0.	0.9236	0.66054	0.9236	0.49799
12	0.	0.	0.	0.	1.0006	0.69213	1.0006	0.53647
13	0.	0.	0.	0.	1.0776	0.72142	1.0776	0.57840
14	0.	0.	0.	0.	1.1545	0.75531	1.1545	0.61459
15	0.	0.	0.	0.	1.2315	0.77886	1.2315	0.64963
16	0.	0.	0.	0.	1.3085	0.80126	1.3085	0.68409
17	0.	0.	0.	0.	1.3854	0.82252	1.3854	0.72430
18	0.	0.	0.	0.	1.4624	0.83860	1.4624	0.75244
19	0.	0.	0.	0.	1.5394	0.85870	1.5394	0.77829
20	0.	0.	0.	0.	1.6164	0.87191	1.6164	0.79667
21	0.	0.	0.	0.	1.6933	0.88800	1.6933	0.81850
22	0.	0.	0.	0.	1.7703	0.89948	1.7703	0.83400
23	0.	0.	0.	0.	1.8473	0.91442	1.8473	0.85296
24	0.	0.	0.	0.	1.9242	0.92476	1.9242	0.87249
25	0.	0.	0.	0.	2.0012	0.93452	2.0012	0.88800
26	0.	0.	0.	0.	2.0782	0.94141	2.0782	0.90121
27	0.	0.	0.	0.	2.1551	0.94543	2.1551	0.91384
28	0.	0.	0.	0.	2.2321	0.95233	2.2321	0.92131
29	0.	0.	0.	0.	2.3091	0.95864	2.3091	0.92935
30	0.	0.	0.	0.	2.3860	1.00000	2.3860	1.00000

Async: priority-2 Normalized throughput (Rho_a)= 0.030000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	W	P(W<=w)	d	P(D<=d)
0	0.088695	0.88790	0.88790	0.89437	0.0770	0.05993	0.0770	0.00333
1	0.103373	0.10100	0.10100	0.09714	0.1539	0.12486	0.1539	0.01776
2	0.00864	0.01054	0.01054	0.00780	0.2309	0.18757	0.2309	0.03774
3	0.00060	0.00055	0.00055	0.00054	0.3079	0.24917	0.3079	0.06437
4	0.00008	0.	0.	0.00015	0.3848	0.31909	0.3848	0.10100
5	0.	0.	0.	0.	0.4618	0.38346	0.4618	0.15261
6	0.	0.	0.	0.	0.5388	0.43618	0.5388	0.20755
7	0.	0.	0.	0.	0.6158	0.49223	0.6158	0.25971
8	0.	0.	0.	0.	0.6927	0.54051	0.6927	0.33407
9	0.	0.	0.	0.	0.7697	0.58380	0.7697	0.38735
10	0.	0.	0.	0.	0.8467	0.61654	0.8467	0.43361
11	0.	0.	0.	0.	0.9236	0.65261	0.9236	0.48446
12	0.	0.	0.	0.	1.0006	0.69034	1.0006	0.52886
13	0.	0.	0.	0.	1.0776	0.71476	1.0776	0.57603
14	0.	0.	0.	0.	1.1545	0.73918	1.1545	0.61432
15	0.	0.	0.	0.	1.2315	0.76415	1.2315	0.65483
16	0.	0.	0.	0.	1.3085	0.78912	1.3085	0.68257
17	0.	0.	0.	0.	1.3854	0.80744	1.3854	0.71032
18	0.	0.	0.	0.	1.4624	0.82242	1.4624	0.73807
19	0.	0.	0.	0.	1.5394	0.84295	1.5394	0.76859
20	0.	0.	0.	0.	1.6164	0.86016	1.6164	0.78857
21	0.	0.	0.	0.	1.6933	0.87292	1.6933	0.80744
22	0.	0.	0.	0.	1.7703	0.88346	1.7703	0.82963
23	0.	0.	0.	0.	1.8473	0.90122	1.8473	0.84961
24	0.	0.	0.	0.	1.9242	0.90899	1.9242	0.86515
25	0.	0.	0.	0.	2.0012	0.91565	2.0012	0.87847
26	0.	0.	0.	0.	2.0782	0.92286	2.0782	0.88735
27	0.	0.	0.	0.	2.1551	0.92230	2.1551	0.89345
28	0.	0.	0.	0.	2.2321	0.93896	2.2321	0.90455
29	0.	0.	0.	0.	2.3091	0.94340	2.3091	0.91065
30	0.	0.	0.	0.	2.3860	1.00000	2.3860	1.00000

89/10/64
10:24:37

fddi.1.out

6

Async: priority-3 Normalized throughput (Rho_a) = 0.030000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
0	0.88748	0.88747	0.88747	0.88747	0.0770	0.05070	0.0770	0.00279
1	0.10343	0.09805	0.09805	0.10302	0.1539	0.10418	0.1539	0.01337
2	0.00819	0.01226	0.01226	0.00931	0.2309	0.15822	0.2309	0.03064
3	0.00075	0.00167	0.00167	0.00066	0.3079	0.21727	0.3079	0.05515
4	0.00008	0.00056	0.00056	0.00020	0.3848	0.27242	0.3848	0.08691
5	0.00000	0.	0.	0.00006	0.4618	0.32702	0.4618	0.11755
6	0.	0.	0.	0.	0.5388	0.39443	0.5388	0.17047
7	0.	0.	0.	0.	0.6158	0.44067	0.6158	0.22618
8	0.	0.	0.	0.	0.6927	0.48524	0.6927	0.28245
9	0.	0.	0.	0.	0.7697	0.53259	0.7697	0.32813
10	0.	0.	0.	0.	0.8467	0.56657	0.8467	0.38162
11	0.	0.	0.	0.	0.9236	0.60279	0.9236	0.43343
12	0.	0.	0.	0.	1.0006	0.63565	1.0006	0.48022
13	0.	0.	0.	0.	1.0776	0.66518	1.0776	0.51198
14	0.	0.	0.	0.	1.1545	0.69304	1.1545	0.55097
15	0.	0.	0.	0.	1.2315	0.71755	1.2315	0.58607
16	0.	0.	0.	0.	1.3085	0.74206	1.3085	0.63343
17	0.	0.	0.	0.	1.3854	0.76602	1.3854	0.66630
18	0.	0.	0.	0.	1.4624	0.78607	1.4624	0.69025
19	0.	0.	0.	0.	1.5394	0.80279	1.5394	0.72201
20	0.	0.	0.	0.	1.6164	0.81894	1.6164	0.74429
21	0.	0.	0.	0.	1.6933	0.83398	1.6933	0.76435
22	0.	0.	0.	0.	1.7703	0.85070	1.7703	0.78496
23	0.	0.	0.	0.	1.8473	0.86685	1.8473	0.80334
24	0.	0.	0.	0.	1.9242	0.88245	1.9242	0.82173
25	0.	0.	0.	0.	2.0012	0.89415	2.0012	0.83788
26	0.	0.	0.	0.	2.0782	0.90474	2.0782	0.85181
27	0.	0.	0.	0.	2.1551	0.91532	2.1551	0.86407
28	0.	0.	0.	0.	2.2321	0.92201	2.2321	0.88078
29	0.	0.	0.	0.	2.3091	0.92981	2.3091	0.89304
30	0.	0.	0.	0.	2.3860	1.00000	2.3860	1.00000

.. Station 5

Synchronous Traffic: Normalized throughput (Rho_a) = 0.005000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
0	0.88710	0.93105	0.93105	0.92214	0.0770	0.07791	0.0770	0.02635
1	0.10523	0.06334	0.06334	0.07333	0.1539	0.15247	0.1539	0.10202
2	0.00707	0.00448	0.00448	0.00414	0.2309	0.21749	0.2309	0.17321
3	0.00053	0.00056	0.00056	0.00029	0.3079	0.28027	0.3079	0.23879
4	0.	0.00056	0.00056	0.	0.3848	0.33913	0.3848	0.30213
5	0.00008	0.	0.	0.00001	0.4618	0.41368	0.4618	0.36211
6	0.	0.	0.	0.	0.5388	0.47141	0.5388	0.43274
7	0.	0.	0.	0.	0.6158	0.53419	0.6158	0.49327
8	0.	0.	0.	0.	0.6927	0.58352	0.6927	0.55045
9	0.	0.	0.	0.	0.7697	0.62668	0.7697	0.60202
10	0.	0.	0.	0.	0.8467	0.65695	0.8467	0.63733
11	0.	0.	0.	0.	0.9236	0.69787	0.9236	0.67040
12	0.	0.	0.	0.	1.0006	0.72758	1.0006	0.71132
13	0.	0.	0.	0.	1.0776	0.75953	1.0776	0.73879
14	0.	0.	0.	0.	1.1545	0.78195	1.1545	0.76738
15	0.	0.	0.	0.	1.2315	0.80381	1.2315	0.79092
16	0.	0.	0.	0.	1.3085	0.81951	1.3085	0.80774
17	0.	0.	0.	0.	1.3854	0.83072	1.3854	0.82231
18	0.	0.	0.	0.	1.4624	0.84697	1.4624	0.83688
19	0.	0.	0.	0.	1.5394	0.86603	1.5394	0.85370
20	0.	0.	0.	0.	1.6164	0.88173	1.6164	0.87052
21	0.	0.	0.	0.	1.6933	0.89070	1.6933	0.88845
22	0.	0.	0.	0.	1.7703	0.89798	1.7703	0.89350
23	0.	0.	0.	0.	1.8473	0.90695	1.8473	0.90191

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
0	0.89003	0.89543	0.89543	0.90179	0.0770	0.07657	0.0770	0.00286
1	0.10102	0.09600	0.09600	0.08991	0.1539	0.15257	0.1539	0.01657
2	0.00819	0.00857	0.00857	0.00756	0.2309	0.21200	0.2309	0.04000
3	0.00075	0.	0.	0.00074	0.3079	0.27257	0.3079	0.07771
4	0.	0.	0.	0.	0.3848	0.34686	0.3848	0.12171
5	0.	0.	0.	0.	0.4618	0.40857	0.4618	0.16743
6	0.	0.	0.	0.	0.5388	0.46743	0.5388	0.22000
7	0.	0.	0.	0.	0.6158	0.53029	0.6158	0.27943
8	0.	0.	0.	0.	0.6927	0.57200	0.6927	0.34971
9	0.	0.	0.	0.	0.7697	0.61943	0.7697	0.40629
10	0.	0.	0.	0.	0.8467	0.64971	0.8467	0.46114
11	0.	0.	0.	0.	0.9236	0.68629	0.9236	0.50971
12	0.	0.	0.	0.	1.0006	0.71371	1.0006	0.56571
13	0.	0.	0.	0.	1.0776	0.74114	1.0776	0.59886
14	0.	0.	0.	0.	1.1545	0.76629	1.1545	0.64000
15	0.	0.	0.	0.	1.2315	0.78571	1.2315	0.67486
16	0.	0.	0.	0.	1.3085	0.80857	1.3085	0.70114
17	0.	0.	0.	0.	1.3854	0.82400	1.3854	0.72857
18	0.	0.	0.	0.	1.4624	0.83829	1.4624	0.76114
19	0.	0.	0.	0.	1.5394	0.85314	1.5394	0.78114
20	0.	0.	0.	0.	1.6164	0.86971	1.6164	0.80171
21	0.	0.	0.	0.	1.6933	0.88457	1.6933	0.81829
22	0.	0.	0.	0.	1.7703	0.89714	1.7703	0.83371
23	0.	0.	0.	0.	1.8473	0.90914	1.8473	0.85257
24	0.	0.	0.	0.	1.9242	0.91829	1.9242	0.86457
25	0.	0.	0.	0.	2.0012	0.92629	2.0012	0.87943
26	0.	0.	0.	0.	2.0782	0.93314	2.0782	0.89143
27	0.	0.	0.	0.	2.1551	0.94171	2.1551	0.90343
28	0.	0.	0.	0.	2.2321	0.94914	2.2321	0.91200
29	0.	0.	0.	0.	2.3091	0.95600	2.3091	0.92400
30	0.	0.	0.	0.	2.3860	1.00000	2.3860	1.00000

Async: priority-1 Normalized throughput (Rho_a) = 0.030000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
0	0.89003	0.89543	0.89543	0.90179	0.0770	0.07657	0.0770	0.00286
1	0.10102	0.09600	0.09600	0.08991	0.1539	0.15257	0.1539	0.01657
2	0.00819	0.00857	0.00857	0.00756	0.2309	0.21200	0.2309	0.04000
3	0.00075	0.	0.	0.00074	0.3079	0.27257	0.3079	0.07771
4	0.	0.	0.	0.	0.3848	0.34686	0.3848	0.12171
5	0.	0.	0.	0.	0.4618	0.40857	0.4618	0.16743
6	0.	0.	0.	0.	0.5388	0.46743	0.5388	0.22000
7	0.	0.	0.	0.	0.6158	0.53029	0.6158	0.27943
8	0.	0.	0.	0.	0.6927	0.57200	0.6927	0.34971
9	0.	0.	0.	0.	0.7697	0.61943	0.7697	0.40629
10	0.	0.	0.	0.	0.8467	0.64971	0.8467	0.46114
11	0.	0.	0.	0.	0.9236	0.68629	0.9236	0.50971
12	0.	0.	0.	0.	1.0006	0.71371	1.0006	0.56571
13	0.	0.	0.	0.	1.0776	0.74114	1.0776	0.59886
14	0.	0.	0.	0.	1.1545	0.76629	1.1545	0.64000
15	0.	0.	0.	0.	1.2315	0.78571	1.2315	0.67486
16	0.	0.	0.	0.	1.3085	0.80857	1.3085	0.70114
17	0.	0.	0.	0.	1.3854	0.82400	1.3854	0.72857
18	0.	0.	0.	0.	1.4624	0.83829	1.4624	0.76114
19	0.	0.	0.	0.	1.5394	0.85314	1.5394	0.78114
20	0.	0.	0.	0.	1.6164	0.86971	1.6164	0.80171
21	0.	0.	0.	0.	1.6933	0.88457	1.6933	0.81829
22	0.	0.	0.	0.	1.7703	0.89714	1.7703	0.83371
23	0.	0.	0.	0.	1.8473	0.90914	1.8473	0.85257
24	0.	0.	0.	0.	1.9242	0.91829	1.9242	0.86457
25	0.	0.	0.	0.	2.0012	0.92629	2.0012	0.87943
26	0.	0.	0.	0.	2.0782	0.93314	2.0782	0.89143
27	0.	0.	0.	0.	2.1551	0.94171	2.1551	0.90343
28	0.	0.	0.	0.	2.2321	0.94914	2.2321	0.91200
29	0.	0.	0.	0.	2.3091	0.95600	2.3091	0.92400
30	0.	0.	0.	0.	2.3860	1.00000	2.3860	1.00000

Async: priority-2 Normalized throughput (Rho_a) = 0.030000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
0	0.88966	0.89228	0.89228	0.89763	0.0770	0.06336	0.0770	0.00403
1	0.10185	0.10311	0.10311	0.09441	0.1539	0.13191	0.1539	0.01728
2	0.00812	0.00461	0.00461	0.00754	0.2309	0.19816	0.2309	0.03687
3	0.00038	0.	0.	0.00042	0.3079	0.25230	0.3079	0.06567
4	0.	0.	0.	0.	0.3848	0.32143	0.3848	0.10599
5	0.	0.	0.	0.	0.4618	0.36694	0.4618	0.14977
6	0.	0.	0.	0.	0.5388	0.42281	0.5388	0.19124
7	0.	0.	0.	0.	0.6158	0.48214	0.6158	0.24597
8	0.	0.	0.	0.	0.6927	0.53456	0.6927	0.30760
9	0.	0.	0.	0.	0.7697	0.57776	0.7697	0.36809
10	0.	0.	0.	0.	0.8467	0.61636	0.8467	0.41763
11	0.	0.	0.	0.	0.9236	0.65668	0.9236	0.47293
12	0.	0.	0.	0.	1.0006	0.69412	1.0006	0.52880
13	0.	0.	0.	0.	1.0776	0.71032	1.0776	0.57776
14	0.	0.	0.	0.	1.1545	0.73963	1.1545	0.61406
15	0.	0.	0.	0.	1.2315	0.75979	1.2315	0.65783
16	0.	0.	0.	0.	1.3085	0.77592	1.3085	0.68594
17	0.	0.	0.	0.	1.3854	0.79551	1.3854	0.71889

89/10/04
10:24:37

fddi.1.out

18	0.	0.	1.4624	0.82200	1.4624	0.73963
19	0.	0.	1.5394	0.83641	1.5394	0.75922
20	0.	0.	1.6164	0.85174	1.6164	0.77880
21	0.	0.	1.6933	0.86733	1.6933	0.79896
22	0.	0.	1.7703	0.88479	1.7703	0.81797
23	0.	0.	1.8473	0.89689	1.8473	0.83756
24	0.	0.	1.9242	0.90726	1.9242	0.85599
25	0.	0.	2.0012	0.91417	2.0012	0.86578
26	0.	0.	2.0782	0.92339	2.0782	0.88594
27	0.	0.	2.1551	0.92857	2.1551	0.89631
28	0.	0.	2.2321	0.93664	2.2321	0.90438
29	0.	0.	2.3091	0.94470	2.3091	0.91417
30	0.	0.	2.3860	1.00000	2.3860	1.00000

Async: priority-3 Normalized throughput (Rho_a)= 0.030000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<=w)	d	P(D<=d)
0	0.88921	0.89381	0.89381	0.88990	0.0770	0.06019	0.0770	0.00398
1	0.10305	0.09767	0.09767	0.10133	0.1539	0.11925	0.1539	0.01760
2	0.00714	0.00852	0.00852	0.00790	0.2309	0.18115	0.2309	0.03407
3	0.00060	0.	0.	0.00087	0.3079	0.24191	0.3079	0.05508
4	0.	0.	0.	0.	0.3848	0.29018	0.3848	0.08461
5	0.	0.	0.	0.	0.4618	0.35548	0.4618	0.12493
6	0.	0.	0.	0.	0.5388	0.40204	0.5388	0.17547
7	0.	0.	0.	0.	0.6158	0.45372	0.6158	0.22998
8	0.	0.	0.	0.	0.6927	0.50199	0.6927	0.27939
9	0.	0.	0.	0.	0.7697	0.54231	0.7697	0.33469
10	0.	0.	0.	0.	0.8467	0.58149	0.8467	0.40829
11	0.	0.	0.	0.	0.9236	0.62351	0.9236	0.48661
12	0.	0.	0.	0.	1.0006	0.66815	1.0006	0.56028
13	0.	0.	0.	0.	1.0776	0.68086	1.0776	0.54628
14	0.	0.	0.	0.	1.1545	0.70926	1.1545	0.58433
15	0.	0.	0.	0.	1.2315	0.73538	1.2315	0.61953
16	0.	0.	0.	0.	1.3085	0.76661	1.3085	0.65361
17	0.	0.	0.	0.	1.3854	0.79046	1.3854	0.68313
18	0.	0.	0.	0.	1.4624	0.80636	1.4624	0.71550
19	0.	0.	0.	0.	1.5394	0.82453	1.5394	0.73878
20	0.	0.	0.	0.	1.6164	0.83759	1.6164	0.76320
21	0.	0.	0.	0.	1.6933	0.85633	1.6933	0.78421
22	0.	0.	0.	0.	1.7703	0.86655	1.7703	0.80352
23	0.	0.	0.	0.	1.8473	0.87848	1.8473	0.82396
24	0.	0.	0.	0.	1.9242	0.88529	1.9242	0.83589
25	0.	0.	0.	0.	2.0012	0.89495	2.0012	0.85179
26	0.	0.	0.	0.	2.0782	0.90857	2.0782	0.86712
27	0.	0.	0.	0.	2.1551	0.91709	2.1551	0.87961
28	0.	0.	0.	0.	2.2321	0.92277	2.2321	0.88927
29	0.	0.	0.	0.	2.3091	0.92845	2.3091	0.89551
30	0.	0.	0.	0.	2.3860	1.00000	2.3860	1.00000

** Station 6

Synchronous Traffic: Normalized throughput (Rho_a)= 0.005000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<=w)	d	P(D<=d)
0	0.88988	0.92452	0.92452	0.92467	0.0770	0.08740	0.0770	0.03519
1	0.10140	0.07321	0.07321	0.07008	0.1539	0.17196	0.1539	0.01521
2	0.00857	0.00227	0.00227	0.00509	0.2309	0.23780	0.2309	0.19864
3	0.00015	0.	0.	0.00016	0.3079	0.30250	0.3079	0.25823
4	0.	0.	0.	0.	0.3848	0.36322	0.3848	0.32690
5	0.	0.	0.	0.	0.4618	0.42565	0.4618	0.38649
6	0.	0.	0.	0.	0.5388	0.47957	0.5388	0.44098
7	0.	0.	0.	0.	0.6158	0.53575	0.6158	0.50965
8	0.	0.	0.	0.	0.6927	0.58173	0.6927	0.55278
9	0.	0.	0.	0.	0.7697	0.62145	0.7697	0.59762

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<=w)	d	P(D<=d)
0	0.89026	0.89960	0.89960	0.89914	0.0770	0.06770	0.0770	0.00287
1	0.10155	0.08950	0.08950	0.09363	0.1539	0.13884	0.1539	0.01721
2	0.00707	0.01090	0.01090	0.00650	0.2309	0.20539	0.2309	0.04016
3	0.00113	0.	0.	0.00073	0.3079	0.27424	0.3079	0.06598
4	0.	0.	0.	0.	0.3848	0.34251	0.3848	0.10212
5	0.	0.	0.	0.	0.4618	0.40103	0.4618	0.15372
6	0.	0.	0.	0.	0.5388	0.46758	0.5388	0.21515
7	0.	0.	0.	0.	0.6158	0.51692	0.6158	0.28112
8	0.	0.	0.	0.	0.6927	0.55766	0.6927	0.34366
9	0.	0.	0.	0.	0.7697	0.59552	0.7697	0.39759
10	0.	0.	0.	0.	0.8467	0.62937	0.8467	0.44865
11	0.	0.	0.	0.	0.9236	0.66667	0.9236	0.50316
12	0.	0.	0.	0.	1.0006	0.69822	1.0006	0.53987
13	0.	0.	0.	0.	1.0776	0.72978	1.0776	0.58003
14	0.	0.	0.	0.	1.1545	0.76076	1.1545	0.62536
15	0.	0.	0.	0.	1.2315	0.78657	1.2315	0.65691
16	0.	0.	0.	0.	1.3085	0.80379	1.3085	0.69593
17	0.	0.	0.	0.	1.3854	0.82387	1.3854	0.72576
18	0.	0.	0.	0.	1.4624	0.84050	1.4624	0.75330
19	0.	0.	0.	0.	1.5394	0.85600	1.5394	0.77625
20	0.	0.	0.	0.	1.6164	0.86976	1.6164	0.79461
21	0.	0.	0.	0.	1.6933	0.88124	1.6933	0.81526
22	0.	0.	0.	0.	1.7703	0.88812	1.7703	0.83534
23	0.	0.	0.	0.	1.8473	0.89788	1.8473	0.85313
24	0.	0.	0.	0.	1.9242	0.90878	1.9242	0.87034
25	0.	0.	0.	0.	2.0012	0.91853	2.0012	0.88583
26	0.	0.	0.	0.	2.0782	0.92484	2.0782	0.89845
27	0.	0.	0.	0.	2.1551	0.93345	2.1551	0.90247
28	0.	0.	0.	0.	2.2321	0.94148	2.2321	0.91452
29	0.	0.	0.	0.	2.3091	0.95123	2.3091	0.91968
30	0.	0.	0.	0.	2.3860	1.00000	2.3860	1.00000

Async: priority-1 Normalized throughput (Rho_a)= 0.030000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<=w)	d	P(D<=d)
0	0.89026	0.89960	0.89960	0.89914	0.0770	0.06770	0.0770	0.00287
1	0.10155	0.08950	0.08950	0.09363	0.1539	0.13884	0.1539	0.01721
2	0.00707	0.01090	0.01090	0.00650	0.2309	0.20539	0.2309	0.04016
3	0.00113	0.	0.	0.00073	0.3079	0.27424	0.3079	0.06598
4	0.	0.	0.	0.	0.3848	0.34251	0.3848	0.10212
5	0.	0.	0.	0.	0.4618	0.40103	0.4618	0.15372
6	0.	0.	0.	0.	0.5388	0.46758	0.5388	0.21515
7	0.	0.	0.	0.	0.6158	0.51692	0.6158	0.28112
8	0.	0.	0.	0.	0.6927	0.55766	0.6927	0.34366
9	0.	0.	0.	0.	0.7697	0.59552	0.7697	0.39759
10	0.	0.	0.	0.	0.8467	0.62937	0.8467	0.44865
11	0.	0.	0.	0.	0.9236	0.66667	0.9236	0.50316
12	0.	0.	0.	0.	1.0006	0.69822	1.0006	0.53987
13	0.	0.	0.	0.	1.0776	0.72978	1.0776	0.58003
14	0.	0.	0.	0.	1.1545	0.76076	1.1545	0.62536
15	0.	0.	0.	0.	1.2315	0.78657	1.2315	0.65691
16	0.	0.	0.	0.	1.3085	0.80379	1.3085	0.69593
17	0.	0.	0.	0.	1.3854	0.82387	1.3854	0.72576
18	0.	0.	0.	0.	1.4624	0.84050	1.4624	0.75330
19	0.	0.	0.	0.	1.5394	0.85600	1.5394	0.77625
20	0.	0.	0.	0.	1.6164	0.86976	1.6164	0.79461
21	0.	0.	0.	0.	1.6933	0.88124	1.6933	0.81526
22	0.	0.	0.	0.	1.7703	0.88812	1.7703	0.83534
23	0.	0.	0.	0.	1.8473	0.89788	1.8473	0.85313
24	0.	0.	0.	0.	1.9242	0.90878	1.9242	0.87034
25	0.	0.	0.	0.	2.0012	0.91853	2.0012	0.88583
26	0.	0.	0.	0.	2.0782	0.92484	2.0782	0.89845
27	0.	0.	0.	0.	2.1551	0.93345	2.1551	0.90247
28	0.	0.	0.	0.	2.2321	0.94148	2.2321	0.91452
29	0.	0.	0.	0.	2.3091	0.95123	2.3091	0.91968
30	0.	0.	0.	0.	2.3860	1.00000	2.3860	1.00000

Async: priority-2 Normalized throughput (Rho_a)= 0.030000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<=w)	d	P(D<=d)
0	0.88763	0.88524	0.88524	0.89374	0.0770	0.05125	0.0770	0.00223
1	0.10305	0.10139	0.10139	0.09656	0.1539	0.10696	0.1539	0.01938
2	0.00842	0.01114	0.01114	0.00863	0.2309	0.17326	0.2309	0.03454
3	0.00075	0.00167	0.00167	0.00090	0.3079	0.24011	0.3079	0.05292

89/10/04
10:24:37

8

fddi.1.out

4	0.00008	0.00056	0.00056	0.00015	0.3848	0.30418	0.3848	0.08691
5	0.00008	0.	0.	0.00002	0.4618	0.36880	0.4618	0.13148
6	0.	0.	0.	0.	0.5388	0.43844	0.5388	0.18217
7	0.	0.	0.	0.	0.6158	0.48691	0.6158	0.24401
8	0.	0.	0.	0.	0.6927	0.53760	0.6927	0.30418
9	0.	0.	0.	0.	0.7697	0.57883	0.7697	0.36323
10	0.	0.	0.	0.	0.8467	0.61783	0.8467	0.42396
11	0.	0.	0.	0.	0.9236	0.65404	0.9236	0.47688
12	0.	0.	0.	0.	1.0006	0.67855	1.0006	0.51755
13	0.	0.	0.	0.	1.0776	0.71811	1.0776	0.56267
14	0.	0.	0.	0.	1.1545	0.74540	1.1545	0.60446
15	0.	0.	0.	0.	1.2315	0.77047	1.2315	0.63788
16	0.	0.	0.	0.	1.3085	0.79053	1.3085	0.67354
17	0.	0.	0.	0.	1.3854	0.80613	1.3854	0.70752
18	0.	0.	0.	0.	1.4624	0.82284	1.4624	0.74039
19	0.	0.	0.	0.	1.5394	0.84345	1.5394	0.75877
20	0.	0.	0.	0.	1.6164	0.86017	1.6164	0.78329
21	0.	0.	0.	0.	1.6933	0.87019	1.6933	0.80390
22	0.	0.	0.	0.	1.7703	0.88022	1.7703	0.82396
23	0.	0.	0.	0.	1.8473	0.89192	1.8473	0.83844
24	0.	0.	0.	0.	1.9242	0.89972	1.9242	0.85237
25	0.	0.	0.	0.	2.0012	0.90696	2.0012	0.86630
26	0.	0.	0.	0.	2.0782	0.91643	2.0782	0.87688
27	0.	0.	0.	0.	2.1551	0.92145	2.1551	0.88579
28	0.	0.	0.	0.	2.2321	0.93148	2.2321	0.89359
29	0.	0.	0.	0.	2.3091	0.93705	2.3091	0.90362
30	0.	0.	0.	0.	2.3860	1.00000	2.3860	1.00000

Async: priority-3 Normalized throughput (Rho_a) = 0.030000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(N<w)	d	P(D<=d)
0	0.88537	0.89372	0.89372	0.88752	0.0770	0.06112	0.0770	0.00275
1	0.10666	0.09857	0.09857	0.10375	0.1539	0.12059	0.1539	0.01156
2	0.00752	0.00661	0.00661	0.00813	0.2309	0.18007	0.2309	0.03249
3	0.00038	0.00110	0.00110	0.00054	0.3079	0.23623	0.3079	0.06057
4	0.00008	0.	0.	0.00005	0.3848	0.30341	0.3848	0.09526
5	0.	0.	0.	0.	0.4618	0.36344	0.4618	0.13932
6	0.	0.	0.	0.	0.5388	0.41685	0.5388	0.19273
7	0.	0.	0.	0.	0.6158	0.48183	0.6158	0.24835
8	0.	0.	0.	0.	0.6927	0.52203	0.6927	0.29736
9	0.	0.	0.	0.	0.7697	0.56498	0.7697	0.35958
10	0.	0.	0.	0.	0.8467	0.60573	0.8467	0.41300
11	0.	0.	0.	0.	0.9236	0.64152	0.9236	0.46421
12	0.	0.	0.	0.	1.0006	0.67731	1.0006	0.52037
13	0.	0.	0.	0.	1.0776	0.70595	1.0776	0.56600
14	0.	0.	0.	0.	1.1545	0.73128	1.1545	0.61068
15	0.	0.	0.	0.	1.2315	0.75716	1.2315	0.63932
16	0.	0.	0.	0.	1.3085	0.77533	1.3085	0.67236
17	0.	0.	0.	0.	1.3854	0.79460	1.3854	0.69273
18	0.	0.	0.	0.	1.4624	0.80947	1.4624	0.72907
19	0.	0.	0.	0.	1.5394	0.82269	1.5394	0.75220
20	0.	0.	0.	0.	1.6164	0.83590	1.6164	0.77258
21	0.	0.	0.	0.	1.6933	0.85518	1.6933	0.79185
22	0.	0.	0.	0.	1.7703	0.86509	1.7703	0.81167
23	0.	0.	0.	0.	1.8473	0.87225	1.8473	0.82874
24	0.	0.	0.	0.	1.9242	0.88436	1.9242	0.83535
25	0.	0.	0.	0.	2.0012	0.89317	2.0012	0.84637
26	0.	0.	0.	0.	2.0782	0.90363	2.0782	0.85793
27	0.	0.	0.	0.	2.1551	0.91520	2.1551	0.86784
28	0.	0.	0.	0.	2.2321	0.92236	2.2321	0.88161
29	0.	0.	0.	0.	2.3091	0.93117	2.3091	0.89593
30	0.	0.	0.	0.	2.3860	1.00000	2.3860	1.00000

Delay-Throughput Evaluator, IRI Corp.

Appendix H.

The FDDI Program: An Example for the 2 Classes of Stations Case

89/10/02
17:47:39

fddi.2.inp

1

** Simulation of a Timed Token Rotation Protocol (FDDI-I Type)
program: FDDI
Professor Izhak Rubin, UCLA

Enter Select Input Mode

1. from KEYBOARD
2. from DATA FILE

1

Enter the output data file name

ff8.1

Enter Feature Selection

1. Symmetric System
2. 2 Classes of Stations
3. Different Loading from Station to Station

2

Enter the statistics collection start time (msec)

400

Enter the stop time (msec)

2000

Enter w,d,x, for computing $P(W>w)$, $P(D>d)$, $P(X>x)$

10 10 2

Enter the number of class-1 stations (N1)

3

Enter the number of class-1 stations (N2)

3

Enter the walk time from a station to its neighboring station (in msec; r)

1

Enter number of message priority classes per station (Np)

3

Enter Target Token Rotation Time (msec; TTRT)

200

Synchronous traffic arrival rate for class-1 and class-2 stations
(packets/msec/station) (as1, as2)

0.1 1

Mean synchronous packet transmission time for class-1 and class-2 stations
(msec) (plens1, plens2)

0.1 0.1

Enter the BandWidth Time for class-1 and class-2 stations (msec)
 $(BWT1 \cdot N1 + BWT2 \cdot N2 < (TTRT - \text{walk time}))$

(the max synchronous message transmission per visit)

20 40

Asynchronous Traffic

Enter priority-1 threshold for both classes (msec; $T_{pri}(\text{class1},1)$ $T_{pri}(\text{class2},1)$)

40 75

Enter priority-1 arrival rate for both classes (packets/msec/station) (aa1 aa2)

0.01 0.1

Enter priority-1 mean transmission time (msec) (plena1 plena2)

0.001 0.001

Enter priority-2 threshold for both classes (msec; $T_{pri}(\text{class1},2)$ $T_{pri}(\text{class2},2)$)

20 30

Enter priority-2 arrival rate for both classes (packets/msec/station) (aa1 aa2)

0.01 0.1

Enter priority-2 mean transmission time (msec) (plena1 plena2)

0.001 0.001

Enter priority-3 threshold for both classes (msec; $T_{pri}(\text{class1},3)$ $T_{pri}(\text{class2},3)$)

10 16

Enter priority-3 arrival rate for both classes (packets/msec/station) (aa1 aa2)

0.01 0.1

Enter priority-3 mean transmission time (msec) (plena1 plena2)

0.001 0.001

This is an example of the input file for 2 classes of stations.

ff8.1	output file
2	ifeat
400	cstart
2000	cend
1 1 1	w d x
3	N1
3	N2
1	r
3	Np
200	TTRT
0.1 1	as1, as2
0.1 0.1	plens1, plens2
20 40	BWT1, BWT2
40 75	T_pri(1,1) T_pri(2,1)
0.01 0.1	aa(1,1) aa(2,1)
0.001 0.001	plena(1,1) plena(2,1)
20 30	T_pri(1,2) T_pri(2,2)
0.01 0.1	aa(1,2) aa(2,2)
0.001 0.001	plena(1,2) plena(2,2)
10 16	T_pri(1,3) T_pri(2,3)
0.01 0.1	aa(1,3) aa(2,3)
0.001 0.001	plena(1,3) plena(2,3)

89/10/03
23:27:04

fddi.2.out

.. Performance of a Timed Token Rotation Protocol (FDDI-type) Ring Network ..

Feature Selected: 2 Classes of Stations

Statistics Start (msec): 2000.0

Statistics Stop (msec): 50000.0

TTRT (msec): 200.000

Number of Stations (N): 6

Number of Priorities (Np): 3

Max Throughput (l-walktime/2TTRT): 0.9850

Normalized Throughput (specified): 0.3310

Normalized Throughput (realized): 0.3295

Realized Mean Cycle Time (msec): 8.9487

Realized Mean Dwell Time (msec): 2.9487

Max Cycle Time (msec): 13.2007 at 4539th cycle, t= 40522.7

Station 1

Walk Time (t:msec): 1.0000

Bandwidth Time (BWt:msec): 20.0000

Arrival Rate for Synchronous Traffic (packets/msec/station): 0.1000

Mean Packet Length for Synchronous Traffic (msec): 0.1000

Asynchronous Traffic: Priority-j T_pri aa(l,j) plena(l,j)

1 40.0000 0.0100 0.0010

2 20.0000 0.0100 0.0010

3 10.0000 0.0100 0.0010

Station 2

Walk Time (t:msec): 1.0000

Bandwidth Time (BWt:msec): 40.0000

Arrival Rate for Synchronous Traffic (packets/msec/station): 1.0000

Mean Packet Length for Synchronous Traffic (msec): 0.1000

Asynchronous Traffic: Priority-j T_pri aa(2,j) plena(2,j)

1 75.0000 0.1000 0.0010

2 30.0000 0.1000 0.0010

3 16.0000 0.1000 0.0010

System Configuration

station 1 2 3 4 5 6

class: 1 2 1 2 1 2

Token goes 5592 cycles in simulation

.. Class-1 E(X) sigma(X) E(N) sigma(N) E(D) sigma(D)

Sync Traffic: 0.8762 0.9350 4.4611 2.6229 4.5611 2.6240

Async: pri-1 0.0856 0.2930 4.4284 2.6095 4.4294 2.6095

Async: pri-2 0.0884 0.2999 4.5088 2.6295 4.5098 2.6295

Async: pri-3 0.1017 0.3232 6.2256 5.4173 6.2266 5.4173

.. Class-2 E(X) sigma(X) E(N) sigma(N) E(D) sigma(D)

Sync Traffic: 8.0314 2.9137 4.0648 2.3676 4.1647 2.3683

Async: pri-1 0.7926 0.8902 4.5159 2.6376 4.5169 2.6376

Async: pri-2 0.8068 0.9000 4.5204 2.6442 4.5214 2.6442

Async: pri-3 0.8038 0.8912 4.4816 2.6481 4.4826 2.6481

Station stat E(X) sigma(X) E(N) sigma(N) E(D) sigma(D)

1 0 4718 0.8716 0.9446 4.4061 2.6365 4.5065 2.6374

1 443 0.0817 0.2871 4.3798 2.5843 4.3807 2.5843

2 482 0.0884 0.3004 4.3937 2.6134 4.3947 2.6134

3 461 0.1001 0.3292 6.2350 5.6339 6.2360 5.6339

2 0 4794 8.0455 2.9493 4.0535 2.3599 4.1536 2.3606

89/10/03
23:27:04

fddi.2.out

6 0.8926467 0.3861633 8.9487285 0.8115571 8.0560818 0.6370821

** Station 1

Synchronous Traffic: Normalized throughput (Rho_a)= 0.010000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<=w)	d	P(D<=d)
0	0.42189	0.65727	0.66470	0.2873	0.03476	0.2873	0.2873	0.02310
1	0.35887	0.25689	0.25689	0.25024	0.5745	0.07016	0.5745	0.05490
2	0.16257	0.06528	0.06528	0.06740	0.8618	0.10577	0.8618	0.09284
3	0.04251	0.01674	0.01674	0.01368	1.1491	0.13841	1.1491	0.12717
4	0.01174	0.00276	0.00276	0.00336	1.4364	0.17295	1.4364	0.16024
5	0.00205	0.00042	0.00042	0.00048	1.7236	0.20454	1.7236	0.19224
6	0.00042	0.00008	0.00008	0.00008	2.0109	0.23506	2.0109	0.22510
7	0.00019	0.00021	0.00021	0.00006	2.2982	0.27257	2.2982	0.26028
8	0.00019	0.00000	0.00000	0.00000	2.5854	0.30097	2.5854	0.29123
9	0.00019	0.00000	0.00000	0.00000	2.8727	0.33362	2.8727	0.32217
10	0.00000	0.00000	0.00000	0.00000	3.1600	0.36668	3.1600	0.35333
11	0.00000	0.00000	0.00000	0.00000	3.4473	0.39932	3.4473	0.38872
12	0.00000	0.00000	0.00000	0.00000	3.7345	0.43069	3.7345	0.41649
13	0.00000	0.00000	0.00000	0.00000	4.0218	0.46439	4.0218	0.45379
14	0.00000	0.00000	0.00000	0.00000	4.3091	0.49343	4.3091	0.48368
15	0.00000	0.00000	0.00000	0.00000	4.5963	0.52649	4.5963	0.51229
16	0.00000	0.00000	0.00000	0.00000	4.8836	0.56231	4.8836	0.54939
17	0.00000	0.00000	0.00000	0.00000	5.1709	0.59474	5.1709	0.58499
18	0.00000	0.00000	0.00000	0.00000	5.4582	0.62675	5.4582	0.61573
19	0.00000	0.00000	0.00000	0.00000	5.7454	0.65324	5.7454	0.64307
20	0.00000	0.00000	0.00000	0.00000	6.0327	0.68143	6.0327	0.67189
21	0.00000	0.00000	0.00000	0.00000	6.3200	0.71725	6.3200	0.70369
22	0.00000	0.00000	0.00000	0.00000	6.6073	0.75223	6.6073	0.74142
23	0.00000	0.00000	0.00000	0.00000	6.8945	0.78635	6.8945	0.77342
24	0.00000	0.00000	0.00000	0.00000	7.1818	0.81475	7.1818	0.80373
25	0.00000	0.00000	0.00000	0.00000	7.4691	0.84676	7.4691	0.83658
26	0.00000	0.00000	0.00000	0.00000	7.7563	0.87897	7.7563	0.86816
27	0.00000	0.00000	0.00000	0.00000	8.0436	0.90801	8.0436	0.89805
28	0.00000	0.00000	0.00000	0.00000	8.3309	0.93196	8.3309	0.92497
29	0.00000	0.00000	0.00000	0.00000	8.6182	0.94913	8.6182	0.94235
30	0.00000	0.00000	0.00000	0.00000	8.9054	1.00000	8.9054	1.00000

Async: priority-1 Normalized throughput (Rho_a)= 0.000010

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<=w)	d	P(D<=d)
0	0.92207	0.95485	0.95485	0.96068	0.2873	0.04063	0.2873	0.04063
1	0.07420	0.04515	0.04515	0.03808	0.5745	0.06546	0.5745	0.06546
2	0.00373	0.00124	0.00124	0.00124	0.8618	0.09707	0.8618	0.09707
3	0.00000	0.00000	0.00000	0.00000	1.1491	0.13350	1.1491	0.13350
4	0.00000	0.00000	0.00000	0.00000	1.4364	0.18510	1.4364	0.18510
5	0.00000	0.00000	0.00000	0.00000	1.7236	0.20316	1.7236	0.20316
6	0.00000	0.00000	0.00000	0.00000	2.0109	0.22799	2.0109	0.22799
7	0.00000	0.00000	0.00000	0.00000	2.2982	0.25508	2.2982	0.25508
8	0.00000	0.00000	0.00000	0.00000	2.5854	0.28442	2.5854	0.28442
9	0.00000	0.00000	0.00000	0.00000	2.8727	0.31377	2.8727	0.31377
10	0.00000	0.00000	0.00000	0.00000	3.1600	0.35214	3.1600	0.35214
11	0.00000	0.00000	0.00000	0.00000	3.4473	0.38149	3.4473	0.37923
12	0.00000	0.00000	0.00000	0.00000	3.7345	0.41535	3.7345	0.41535
13	0.00000	0.00000	0.00000	0.00000	4.0218	0.46501	4.0218	0.46501
14	0.00000	0.00000	0.00000	0.00000	4.3091	0.50790	4.3091	0.50790
15	0.00000	0.00000	0.00000	0.00000	4.5963	0.53499	4.5963	0.53499
16	0.00000	0.00000	0.00000	0.00000	4.8836	0.56208	4.8836	0.56208
17	0.00000	0.00000	0.00000	0.00000	5.1709	0.59594	5.1709	0.59594
18	0.00000	0.00000	0.00000	0.00000	5.4582	0.63657	5.4582	0.63657
19	0.00000	0.00000	0.00000	0.00000	5.7454	0.65688	5.7454	0.65688
20	0.00000	0.00000	0.00000	0.00000	6.0327	0.69300	6.0327	0.69300
21	0.00000	0.00000	0.00000	0.00000	6.3200	0.73138	6.3200	0.73138

22	0.00000	0.00000	0.00000	0.00000	6.6073	0.77201	6.6073	0.77201
23	0.00000	0.00000	0.00000	0.00000	6.8945	0.79684	6.8945	0.79684
24	0.00000	0.00000	0.00000	0.00000	7.1818	0.83296	7.1818	0.83296
25	0.00000	0.00000	0.00000	0.00000	7.4691	0.85553	7.4691	0.85553
26	0.00000	0.00000	0.00000	0.00000	7.7563	0.88939	7.7563	0.88939
27	0.00000	0.00000	0.00000	0.00000	8.0436	0.91422	8.0436	0.91422
28	0.00000	0.00000	0.00000	0.00000	8.3309	0.93228	8.3309	0.93228
29	0.00000	0.00000	0.00000	0.00000	8.6182	0.96163	8.6182	0.96163
30	0.00000	0.00000	0.00000	0.00000	8.9054	1.00000	8.9054	1.00000

Async: priority-2 Normalized throughput (Rho_a)= 0.000010

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<=w)	d	P(D<=d)
0	0.91573	0.95228	0.95228	0.95751	0.2873	0.04357	0.2873	0.04149
1	0.08091	0.03942	0.03942	0.04102	0.5745	0.06846	0.5745	0.06846
2	0.00261	0.00830	0.00830	0.00118	0.8618	0.09129	0.8618	0.09129
3	0.00075	0.00028	0.00028	0.00028	1.1491	0.13485	1.1491	0.13485
4	0.00000	0.00000	0.00000	0.00000	1.4364	0.16805	1.4364	0.16805
5	0.00000	0.00000	0.00000	0.00000	1.7236	0.19502	1.7236	0.19502
6	0.00000	0.00000	0.00000	0.00000	2.0109	0.22407	2.0109	0.22407
7	0.00000	0.00000	0.00000	0.00000	2.2982	0.26763	2.2982	0.26763
8	0.00000	0.00000	0.00000	0.00000	2.5854	0.30705	2.5854	0.30705
9	0.00000	0.00000	0.00000	0.00000	2.8727	0.33195	2.8727	0.33195
10	0.00000	0.00000	0.00000	0.00000	3.1600	0.36515	3.1600	0.36515
11	0.00000	0.00000	0.00000	0.00000	3.4473	0.39834	3.4473	0.39834
12	0.00000	0.00000	0.00000	0.00000	3.7345	0.43361	3.7345	0.43361
13	0.00000	0.00000	0.00000	0.00000	4.0218	0.46266	4.0218	0.46266
14	0.00000	0.00000	0.00000	0.00000	4.3091	0.49170	4.3091	0.49170
15	0.00000	0.00000	0.00000	0.00000	4.5963	0.52905	4.5963	0.52905
16	0.00000	0.00000	0.00000	0.00000	4.8836	0.57054	4.8836	0.57054
17	0.00000	0.00000	0.00000	0.00000	5.1709	0.60996	5.1709	0.60996
18	0.00000	0.00000	0.00000	0.00000	5.4582	0.63693	5.4582	0.63693
19	0.00000	0.00000	0.00000	0.00000	5.7454	0.66805	5.7454	0.66805
20	0.00000	0.00000	0.00000	0.00000	6.0327	0.70747	6.0327	0.70747
21	0.00000	0.00000	0.00000	0.00000	6.3200	0.74274	6.3200	0.74274
22	0.00000	0.00000	0.00000	0.00000	6.6073	0.75726	6.6073	0.75726
23	0.00000	0.00000	0.00000	0.00000	6.8945	0.77801	6.8945	0.77801
24	0.00000	0.00000	0.00000	0.00000	7.1818	0.81328	7.1818	0.81328
25	0.00000	0.00000	0.00000	0.00000	7.4691	0.83817	7.4691	0.83817
26	0.00000	0.00000	0.00000	0.00000	7.7563	0.86929	7.7563	0.86929
27	0.00000	0.00000	0.00000	0.00000	8.0436	0.89627	8.0436	0.89627
28	0.00000	0.00000	0.00000	0.00000	8.3309	0.92116	8.3309	0.92116
29	0.00000	0.00000	0.00000	0.00000	8.6182	0.95436	8.6182	0.95436
30	0.00000	0.00000	0.00000	0.00000	8.9054	1.00000	8.9054	1.00000

Async: priority-3 Normalized throughput (Rho_a)= 0.000010

j	P (X=j)	P (Xa=j)	P (Xd=j)	P (Xt=j)	w	P (W<=w)	d	P (D<=d)
0	0.90660	0.93926	0.93926	0.94411	0.2873	0.03037	0.2873	0.03037
1	0.08837	0.04989	0.04989	0.05324	0.5745	0.05640	0.5745	0.05640
2	0.00391	0.00651	0.00651	0.00181	0.8618	0.08026	0.8618	0.08026
3	0.00075	0.00217	0.00217	0.00053	1.1491	0.10412	1.1491	0.10412
4	0.00019	0.00217	0.00217	0.00015	1.4364	0.14100	1.4364	0.14100
5	0.00019	0.00016	0.00016	0.00016	1.7236	0.16920	1.7236	0.16920
6	0.0	0.0	0.0	0.0	2.0109	0.18004	2.0109	0.18004
7	0.0	0.0	0.0	0.0	2.2982	0.20824	2.2982	0.20824
8	0.0	0.0	0.0	0.0	2.5854	0.24946	2.5854	0.24946
9	0.0	0.0	0.0	0.0	2.8727	0.27115	2.8727	0.27115
10	0.0	0.0	0.0	0.0	3.1600	0.28416	3.1600	0.28416
11	0.0	0.0	0.0	0.0	3.4473	0.32104	3.4473	0.32104
12	0.0	0.0	0.0	0.0	3.7345	0.35575	3.7345	0.35575
13	0.0	0.0	0.0	0.0	4.0218	0.39479	4.0218	0.39479
14	0.0	0.0	0.0	0.0	4.3091	0.42516	4.3091	0.42516
15	0.0	0.0	0.0	0.0	4.5963	0.45770	4.5963	0.45770

89/10/03
23:27:04

fddi.2.out

3

16	0.	0.	4.8836	0.49024	4.8836	0.49024
17	0.	0.	5.1709	0.50759	5.1709	0.50759
18	0.	0.	5.4582	0.54447	5.4582	0.54447
19	0.	0.	5.7454	0.58333	5.7454	0.58333
20	0.	0.	6.0327	0.59436	6.0327	0.59436
21	0.	0.	6.3200	0.62256	6.3200	0.62256
22	0.	0.	6.6073	0.64642	6.6073	0.64642
23	0.	0.	6.8945	0.67245	6.8945	0.67245
24	0.	0.	7.1818	0.69197	7.1818	0.69197
25	0.	0.	7.4691	0.70716	7.4691	0.70716
26	0.	0.	7.7563	0.73102	7.7563	0.73102
27	0.	0.	8.0436	0.76356	8.0436	0.76356
28	0.	0.	8.3309	0.79610	8.3309	0.79610
29	0.	0.	8.6182	0.80911	8.6182	0.80911
30	0.	0.	8.9054	1.00000	8.9054	1.00000

** Station 2

Synchronous Traffic: Normalized throughput (Rho_s) = 0.100000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	W	P(W<w)	d	P(D<d)
0	0.00019	0.11194	0.11194	0.11395	0.2873	0.03439	0.2873	0.02255
1	0.00205	0.12338	0.12338	0.12283	0.5745	0.06920	0.5745	0.05724
2	0.01212	0.12246	0.12246	0.12212	0.8618	0.10516	0.8618	0.09278
3	0.03300	0.11788	0.11788	0.11937	1.1491	0.14208	1.1491	0.12834
4	0.05537	0.11269	0.11269	0.11142	1.4364	0.17676	1.4364	0.16440
5	0.09732	0.10060	0.10060	0.10276	1.7236	0.21274	1.7236	0.19959
6	0.12547	0.08509	0.08509	0.08689	2.0109	0.24986	2.0109	0.23711
7	0.13143	0.06879	0.06879	0.06678	2.2982	0.28428	2.2982	0.27248
8	0.12752	0.05357	0.05357	0.05227	2.5854	0.31953	2.5854	0.30781
9	0.11670	0.03877	0.03877	0.03769	2.8727	0.35405	2.8727	0.34239
10	0.10123	0.02585	0.02585	0.02651	3.1600	0.39025	3.1600	0.37739
11	0.07457	0.01680	0.01680	0.01587	3.4473	0.42608	3.4473	0.41362
12	0.04884	0.01017	0.01017	0.01000	3.7345	0.46352	3.7345	0.45026
13	0.03113	0.00613	0.00613	0.00568	4.0218	0.50013	4.0218	0.48689
14	0.02144	0.00298	0.00298	0.00303	4.3091	0.53446	4.3091	0.52291
15	0.01119	0.00156	0.00156	0.00159	4.5963	0.56984	4.5963	0.55772
16	0.00466	0.00081	0.00081	0.00066	4.8836	0.60660	4.8836	0.59384
17	0.00261	0.00038	0.00038	0.00034	5.1709	0.64172	5.1709	0.62938
18	0.00224	0.00010	0.00010	0.00014	5.4582	0.67856	5.4582	0.66611
19	0.00056	0.00004	0.00004	0.00007	5.7454	0.71495	5.7454	0.71912
20	0.00019	0.00002	0.00002	0.00000	6.0327	0.75139	6.0327	0.73892
21	0.00019	0.	0.	0.	6.3200	0.78613	6.3200	0.77367
22	0.	0.	0.	0.	6.6073	0.82109	6.6073	0.80877
23	0.	0.	0.	0.	6.8945	0.85774	6.8945	0.84494
24	0.	0.	0.	0.	7.1818	0.89171	7.1818	0.88031
25	0.	0.	0.	0.	7.4691	0.92321	7.4691	0.91245
26	0.	0.	0.	0.	7.7563	0.94858	7.7563	0.94051
27	0.	0.	0.	0.	8.0436	0.96734	8.0436	0.96171
28	0.	0.	0.	0.	8.3309	0.98193	8.3309	0.97720
29	0.	0.	0.	0.	8.6182	0.99045	8.6182	0.98799
30	0.	0.	0.	0.	8.9054	1.00000	8.9054	1.00000

Async: priority-1 Normalized throughput (Rho_a) = 0.000100

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	W	P(W<w)	d	P(D<d)
0	0.44705	0.66464	0.66464	0.65913	0.2873	0.03001	0.2873	0.03001
1	0.36503	0.25184	0.25184	0.25540	0.5745	0.06191	0.5745	0.06191
2	0.14224	0.06590	0.06590	0.06845	0.8618	0.09570	0.8618	0.09570
3	0.03747	0.01406	0.01406	0.01411	1.1491	0.12844	1.1491	0.12802
4	0.00652	0.00273	0.00273	0.00245	1.4364	0.16180	1.4364	0.16180
5	0.00149	0.00084	0.00084	0.00036	1.7236	0.18909	1.7236	0.18888
6	0.00019	0.	0.	0.	2.0109	0.22204	2.0109	0.22204
7	0.	0.	0.	0.	2.2982	0.25268	2.2982	0.25268

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	W	P(W<w)	d	P(D<d)
0	0.44780	0.65353	0.65353	0.66001	0.2873	0.03343	0.2873	0.03343
1	0.35757	0.25526	0.25526	0.25194	0.5745	0.06500	0.5745	0.06500
2	0.14504	0.07119	0.07119	0.06979	0.8618	0.10111	0.8618	0.10091
3	0.03784	0.01651	0.01651	0.01446	1.1491	0.13702	1.1491	0.13702
4	0.01025	0.00310	0.00310	0.00342	1.4364	0.16962	1.4364	0.16962
5	0.00149	0.00041	0.00041	0.00037	1.7236	0.19934	1.7236	0.19934
6	0.	0.	0.	0.00000	2.0109	0.23380	2.0109	0.23380
7	0.	0.	0.	0.	2.2982	0.26125	2.2982	0.26083
8	0.	0.	0.	0.	2.5854	0.28725	2.5854	0.28704
9	0.	0.	0.	0.	2.8727	0.32130	2.8727	0.32130
10	0.	0.	0.	0.	3.1600	0.35493	3.1600	0.35473
11	0.	0.	0.	0.	3.4473	0.38816	3.4473	0.38816
12	0.	0.	0.	0.	3.7345	0.42303	3.7345	0.42303
13	0.	0.	0.	0.	4.0218	0.45976	4.0218	0.45976
14	0.	0.	0.	0.	4.3091	0.49051	4.3091	0.49051
15	0.	0.	0.	0.	4.5963	0.52394	4.5963	0.52373
16	0.	0.	0.	0.	4.8836	0.55365	4.8836	0.55365
17	0.	0.	0.	0.	5.1709	0.58502	5.1709	0.58481
18	0.	0.	0.	0.	5.4582	0.61886	5.4582	0.61886
19	0.	0.	0.	0.	5.7454	0.65064	5.7454	0.65064
20	0.	0.	0.	0.	6.0327	0.68366	6.0327	0.68366
21	0.	0.	0.	0.	6.3200	0.71255	6.3200	0.71234
22	0.	0.	0.	0.	6.6073	0.74412	6.6073	0.74412
23	0.	0.	0.	0.	6.8945	0.77548	6.8945	0.77548
24	0.	0.	0.	0.	7.1818	0.80726	7.1818	0.80726
25	0.	0.	0.	0.	7.4691	0.83698	7.4691	0.83698
26	0.	0.	0.	0.	7.7563	0.86669	7.7563	0.86669
27	0.	0.	0.	0.	8.0436	0.89744	8.0436	0.89744
28	0.	0.	0.	0.	8.3309	0.92365	8.3309	0.92365
29	0.	0.	0.	0.	8.6182	0.94573	8.6182	0.94573
30	0.	0.	0.	0.	8.9054	1.00000	8.9054	1.00000

Async: priority-2 Normalized throughput (Rho_a) = 0.000100

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	W	P(W<w)	d	P(D<d)
0	0.44780	0.65353	0.65353	0.66001	0.2873	0.03343	0.2873	0.03343
1	0.35757	0.25526	0.25526	0.25194	0.5745	0.06500	0.5745	0.06500
2	0.14504	0.07119	0.07119	0.06979	0.8618	0.10111	0.8618	0.10091
3	0.03784	0.01651	0.01651	0.01446	1.1491	0.13702	1.1491	0.13702
4	0.01025	0.00310	0.00310	0.00342	1.4364	0.16962	1.4364	0.16962
5	0.00149	0.00041	0.00041	0.00037	1.7236	0.19934	1.7236	0.19934
6	0.	0.	0.	0.00000	2.0109	0.23380	2.0109	0.23380
7	0.	0.	0.	0.	2.2982	0.26125	2.2982	0.26083
8	0.	0.	0.	0.	2.5854	0.28725	2.5854	0.28704
9	0.	0.	0.	0.	2.8727	0.32130	2.8727	0.32130
10	0.	0.	0.	0.	3.1600	0.35493	3.1600	0.35473
11	0.	0.	0.	0.	3.4473	0.38816	3.4473	0.38816
12	0.	0.	0.	0.	3.7345	0.42303	3.7345	0.42303
13	0.	0.	0.	0.	4.0218	0.45976	4.0218	0.45976
14	0.	0.	0.	0.	4.3091	0.49051	4.3091	0.49051
15	0.	0.	0.	0.	4.5963	0.52394	4.5963	0.52373
16	0.	0.	0.	0.	4.8836	0.55365	4.8836	0.55365
17	0.	0.	0.	0.	5.1709	0.58502	5.1709	0.58481
18	0.	0.	0.	0.	5.4582	0.61886	5.4582	0.61886
19	0.	0.	0.	0.	5.7454	0.65064	5.7454	0.65064
20	0.	0.	0.	0.	6.0327	0.68366	6.0327	0.68366
21	0.	0.	0.	0.	6.3200	0.71255	6.3200	0.71234
22	0.	0.	0.	0.	6.6073	0.74412	6.6073	0.74412
23	0.	0.	0.	0.	6.8945	0.77548	6.8945	0.77548
24	0.	0.	0.	0.	7.1818	0.80726	7.1818	0.80726
25	0.	0.	0.	0.	7.4691	0.83698	7.4691	0.83698
26	0.	0.	0.	0.	7.7563	0.86669	7.7563	0.86669
27	0.	0.	0.	0.	8.0436	0.89744	8.0436	0.89744
28	0.	0.	0.	0.	8.3309	0.92365	8.3309	0.92365
29	0.	0.	0.	0.	8.6182	0.94573	8.6182	0.94573
30	0.	0.	0.	0.	8.9054	1.00000	8.9054	1.00000

Async: priority-3 Normalized throughput (Rho_a) = 0.000100

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	W	P(W<w)	d	P(D<d)
0	0.45283	0.66688	0.66688	0.66181	0.2873	0.03142	0.2873	0.03142
1	0.35962	0.24841	0.24841	0.25468	0.5745	0.06200	0.5745	0.06118

89/10/03
23:27:04

fddi.2.out

4

2	0.14169	0.06837	0.06837	0.06704	0.8618	0.09724	0.8618	0.09703
3	0.03729	0.01444	0.01444	0.01379	1.1491	0.13142	1.1491	0.13142
4	0.00802	0.00191	0.00191	0.00247	1.4364	0.16773	1.4364	0.16773
5	0.00056	0.	0.	0.00021	1.7236	0.19703	1.7236	0.19639
6	0.	0.	0.	0.	2.0109	0.23355	2.0109	0.23355
7	0.	0.	0.	0.	2.2982	0.26157	2.2982	0.26136
8	0.	0.	0.	0.	2.5854	0.29639	2.5854	0.29639
9	0.	0.	0.	0.	2.8727	0.32590	2.8727	0.32590
10	0.	0.	0.	0.	3.1600	0.35329	3.1600	0.35329
11	0.	0.	0.	0.	3.4473	0.38832	3.4473	0.38832
12	0.	0.	0.	0.	3.7345	0.42102	3.7345	0.42059
13	0.	0.	0.	0.	4.0218	0.45138	4.0218	0.45138
14	0.	0.	0.	0.	4.3091	0.48747	4.3091	0.48747
15	0.	0.	0.	0.	4.5963	0.52251	4.5963	0.52251
16	0.	0.	0.	0.	4.8836	0.55117	4.8836	0.55117
17	0.	0.	0.	0.	5.1709	0.58429	5.1709	0.58386
18	0.	0.	0.	0.	5.4582	0.61444	5.4582	0.61444
19	0.	0.	0.	0.	5.7454	0.64713	5.7454	0.64713
20	0.	0.	0.	0.	6.0327	0.67749	6.0327	0.67728
21	0.	0.	0.	0.	6.3200	0.70531	6.3200	0.70531
22	0.	0.	0.	0.	6.6073	0.73503	6.6073	0.73503
23	0.	0.	0.	0.	6.8945	0.76858	6.8945	0.76837
24	0.	0.	0.	0.	7.1818	0.80042	7.1818	0.80042
25	0.	0.	0.	0.	7.4691	0.83355	7.4691	0.83355
26	0.	0.	0.	0.	7.7563	0.86136	7.7563	0.86115
27	0.	0.	0.	0.	8.0436	0.89448	8.0436	0.89448
28	0.	0.	0.	0.	8.3309	0.92081	8.3309	0.92081
29	0.	0.	0.	0.	8.6182	0.94013	8.6182	0.94013
30	0.	0.	0.	0.	8.9054	1.00000	8.9054	1.00000

Station 3

Synchronous Traffic: Normalized throughput (Rho_s) = 0.010000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
1	0.41424	0.65829	0.65829	0.65283	0.2873	0.03268	0.2873	0.02200
2	0.36484	0.25519	0.25519	0.25965	0.5745	0.06746	0.5745	0.05405
3	0.16219	0.06893	0.06893	0.07004	0.8618	0.09449	0.8618	0.08339
4	0.04661	0.01446	0.01446	0.01444	1.1491	0.12613	1.1491	0.11628
5	0.01025	0.00230	0.00230	0.00272	1.4364	0.16132	1.4364	0.15085
6	0.00130	0.00063	0.00063	0.00077	1.7236	0.19338	1.7236	0.18248
7	0.00037	0.00021	0.00021	0.00006	2.0109	0.22648	2.0109	0.21454
8	0.00019	0.	0.	0.00000	2.2982	0.26000	2.2982	0.24680
9	0.	0.	0.	0.	2.5854	0.29038	2.5854	0.27928
10	0.	0.	0.	0.	2.8727	0.32391	2.8727	0.31322
11	0.	0.	0.	0.	3.1600	0.35345	3.1600	0.34423
12	0.	0.	0.	0.	3.4473	0.38257	3.4473	0.37251
13	0.	0.	0.	0.	3.7345	0.41420	3.7345	0.40561
14	0.	0.	0.	0.	4.0218	0.44458	4.0218	0.43453
15	0.	0.	0.	0.	4.3091	0.48104	4.3091	0.46742
16	0.	0.	0.	0.	4.5963	0.51351	4.5963	0.50241
17	0.	0.	0.	0.	4.8836	0.54389	4.8836	0.53258
18	0.	0.	0.	0.	5.1709	0.57679	5.1709	0.56673
19	0.	0.	0.	0.	5.4582	0.60591	5.4582	0.59480
20	0.	0.	0.	0.	5.7454	0.64215	5.7454	0.62854
21	0.	0.	0.	0.	6.0327	0.67484	6.0327	0.66164
22	0.	0.	0.	0.	6.3200	0.70438	6.3200	0.69202
23	0.	0.	0.	0.	6.6073	0.74209	6.6073	0.73141
24	0.	0.	0.	0.	6.8945	0.77897	6.8945	0.76409
25	0.	0.	0.	0.	7.1818	0.81291	7.1818	0.80327
26	0.	0.	0.	0.	7.4691	0.84538	7.4691	0.83260
27	0.	0.	0.	0.	7.7563	0.87723	7.7563	0.86801
28	0.	0.	0.	0.	8.0436	0.90928	8.0436	0.89776
29	0.	0.	0.	0.	8.3309	0.93484	8.3309	0.92730
30	0.	0.	0.	0.	8.6182	1.00000	8.6182	1.00000

29	0.	0.	0.	0.	8.6182	0.95391	8.6182	0.94699
30	0.	0.	0.	0.	8.9054	1.00000	8.9054	1.00000

Async: priority-1 Normalized throughput (Rho_a) = 0.000010

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
1	0.91536	0.94444	0.94444	0.95603	0.2873	0.02675	0.2873	0.02675
2	0.07979	0.05556	0.05556	0.04266	0.5745	0.04733	0.5745	0.04733
3	0.00485	0.	0.	0.00131	0.8618	0.09877	0.8618	0.09877
4	0.	0.	0.	0.	1.1491	0.13374	1.1491	0.13374
5	0.	0.	0.	0.	1.4364	0.16461	1.4364	0.16461
6	0.	0.	0.	0.	1.7236	0.19959	1.7236	0.19959
7	0.	0.	0.	0.	2.0109	0.23868	2.0109	0.23868
8	0.	0.	0.	0.	2.2982	0.27366	2.2982	0.27366
9	0.	0.	0.	0.	2.5854	0.31070	2.5854	0.31070
10	0.	0.	0.	0.	2.8727	0.34156	2.8727	0.34156
11	0.	0.	0.	0.	3.1600	0.36214	3.1600	0.36214
12	0.	0.	0.	0.	3.4473	0.38683	3.4473	0.38683
13	0.	0.	0.	0.	3.7345	0.41770	3.7345	0.41770
14	0.	0.	0.	0.	4.0218	0.45062	4.0218	0.45062
15	0.	0.	0.	0.	4.3091	0.47942	4.3091	0.47942
16	0.	0.	0.	0.	4.5963	0.51440	4.5963	0.51440
17	0.	0.	0.	0.	4.8836	0.54321	4.8836	0.54321
18	0.	0.	0.	0.	5.1709	0.57819	5.1709	0.57819
19	0.	0.	0.	0.	5.4582	0.61111	5.4582	0.61111
20	0.	0.	0.	0.	5.7454	0.64403	5.7454	0.64403
21	0.	0.	0.	0.	6.0327	0.68107	6.0327	0.68107
22	0.	0.	0.	0.	6.3200	0.71399	6.3200	0.71399
23	0.	0.	0.	0.	6.6073	0.74486	6.6073	0.74486
24	0.	0.	0.	0.	6.8945	0.77984	6.8945	0.77984
25	0.	0.	0.	0.	7.1818	0.80247	7.1818	0.80247
26	0.	0.	0.	0.	7.4691	0.83745	7.4691	0.83745
27	0.	0.	0.	0.	7.7563	0.87449	7.7563	0.87449
28	0.	0.	0.	0.	8.0436	0.89918	8.0436	0.89918
29	0.	0.	0.	0.	8.3309	0.92387	8.3309	0.92387
30	0.	0.	0.	0.	8.6182	0.95267	8.6182	0.95267

Async: priority-2 Normalized throughput (Rho_a) = 0.000010

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
1	0.91555	0.95000	0.95000	0.95523	0.2873	0.02708	0.2873	0.02708
2	0.07998	0.05000	0.05000	0.04351	0.5745	0.06667	0.5745	0.06667
3	0.00447	0.	0.	0.00126	0.8618	0.08542	0.8618	0.08542
4	0.	0.	0.	0.	1.1491	0.11458	1.1491	0.11458
5	0.	0.	0.	0.	1.4364	0.15000	1.4364	0.15000
6	0.	0.	0.	0.	1.7236	0.18750	1.7236	0.18750
7	0.	0.	0.	0.	2.0109	0.20625	2.0109	0.20625
8	0.	0.	0.	0.	2.2982	0.23750	2.2982	0.23750
9	0.	0.	0.	0.	2.5854	0.27500	2.5854	0.27500
10	0.	0.	0.	0.	2.8727	0.31042	2.8727	0.31042
11	0.	0.	0.	0.	3.1600	0.33342	3.1600	0.33342
12	0.	0.	0.	0.	3.4473	0.37708	3.4473	0.37708
13	0.	0.	0.	0.	3.7345	0.42083	3.7345	0.42083
14	0.	0.	0.	0.	4.0218	0.43958	4.0218	0.43958
15	0.	0.	0.	0.	4.3091	0.47917	4.3091	0.47917
16	0.	0.	0.	0.	4.5963	0.52083	4.5963	0.52083
17	0.	0.	0.	0.	4.8836	0.56042	4.8836	0.56042
18	0.	0.	0.	0.	5.1709	0.58750	5.1709	0.58750
19	0.	0.	0.	0.	5.4582	0.61458	5.4582	0.61458
20	0.	0.	0.	0.	5.7454	0.62917	5.7454	0.62917
21	0.	0.	0.	0.	6.0327	0.66250	6.0327	0.66250
22	0.	0.	0.	0.	6.3200	0.70833	6.3200	0.70833
23	0.	0.	0.	0.	6.6073	0.74167	6.6073	0.74167

89/10/03
23:27:04

fdi.2.out

5

```
23 0. 0. 0. 0. 6.8945 0.77292 6.8945 0.77292 0.00000 0.00000
24 0. 0. 0. 0. 7.1818 0.80000 7.1818 0.80000 0.00000 0.00000
25 0. 0. 0. 0. 7.4691 0.82500 7.4691 0.82500 0.00000 0.00000
26 0. 0. 0. 0. 7.7563 0.84375 7.7563 0.84375 0.00000 0.00000
27 0. 0. 0. 0. 8.0436 0.87500 8.0436 0.87500 0.00000 0.00000
28 0. 0. 0. 0. 8.3309 0.89792 8.3309 0.89792 0.00000 0.00000
29 0. 0. 0. 0. 8.6182 0.92292 8.6182 0.92292 0.00000 0.00000
30 0. 0. 0. 0. 8.9054 1.00000 8.9054 1.00000 0.00000 0.00000
```

Async: priority-3 Normalized throughput (Rho_a)= 0.000010

```
J P(X=j) P(Xa=j) P(Xd=j) P(Xt=j) W P(W<=w) d P(D<=d)
0 0.90268 0.94118 0.94130 0.94217 0.2873 0.02516 0.2873 0.02516
1 0.09191 0.05672 0.05660 0.05579 0.5745 0.04822 0.5745 0.04822
2 0.00522 0.00210 0.00210 0.00201 0.8618 0.09015 0.8618 0.09015
3 0.00019 0. 0.00003 0. 1.1491 0.10901 1.1491 0.10901
4 0. 0. 0. 0. 1.4364 0.12579 1.4364 0.12579
5 0. 0. 0. 0. 1.7236 0.15723 1.7236 0.15723
6 0. 0. 0. 0. 2.0109 0.17610 2.0109 0.17610
7 0. 0. 0. 0. 2.2982 0.20126 2.2982 0.20126
8 0. 0. 0. 0. 2.5854 0.22409 2.5854 0.22409
9 0. 0. 0. 0. 2.8727 0.27044 2.8727 0.27044
10 0. 0. 0. 0. 3.1600 0.30189 3.1600 0.29979
11 0. 0. 0. 0. 3.4473 0.34172 3.4473 0.34172
12 0. 0. 0. 0. 3.7345 0.35639 3.7345 0.35639
13 0. 0. 0. 0. 4.0218 0.38365 4.0218 0.38365
14 0. 0. 0. 0. 4.3091 0.42977 4.3091 0.42767
15 0. 0. 0. 0. 4.5963 0.46541 4.5963 0.46541
16 0. 0. 0. 0. 4.8836 0.48008 4.8836 0.48008
17 0. 0. 0. 0. 5.1709 0.50734 5.1709 0.50734
18 0. 0. 0. 0. 5.4582 0.53459 5.4582 0.53459
19 0. 0. 0. 0. 5.7454 0.56604 5.7454 0.56604
20 0. 0. 0. 0. 6.0327 0.60168 6.0327 0.60168
21 0. 0. 0. 0. 6.3200 0.63103 6.3200 0.63103
22 0. 0. 0. 0. 6.6073 0.66667 6.6073 0.66457
23 0. 0. 0. 0. 6.8945 0.70231 6.8945 0.70231
24 0. 0. 0. 0. 7.1818 0.72537 7.1818 0.72537
25 0. 0. 0. 0. 7.4691 0.74633 7.4691 0.74633
26 0. 0. 0. 0. 7.7563 0.76520 7.7563 0.76520
27 0. 0. 0. 0. 8.0436 0.78407 8.0436 0.78407
28 0. 0. 0. 0. 8.3309 0.81342 8.3309 0.81342
29 0. 0. 0. 0. 8.6182 0.83019 8.6182 0.83019
30 0. 0. 0. 0. 8.9054 1.00000 8.9054 1.00000
```

** Station 4

Synchronous Traffic: Normalized throughput (Rho_s)= 0.100000

```
J P(X=j) P(Xa=j) P(Xd=j) P(Xt=j) W P(W<=w) d P(D<=d)
0 0. 0. 0. 0. 0.2873 0.03457 0.2873 0.02230
1 0.00261 0.12319 0.12319 0.12373 0.5745 0.07064 0.5745 0.05820
2 0.01100 0.12177 0.12177 0.12235 0.8618 0.10682 0.8618 0.09457
3 0.03262 0.11940 0.11940 0.11858 1.1491 0.14074 1.1491 0.12845
4 0.05798 0.11316 0.11316 0.11268 1.4364 0.17775 1.4364 0.16425
5 0.08576 0.10232 0.10232 0.10111 1.7236 0.21311 1.7236 0.20097
6 0.11294 0.08695 0.08695 0.08714 2.0109 0.25024 2.0109 0.23744
7 0.14355 0.06869 0.06869 0.07012 2.2982 0.28574 2.2982 0.27297
8 0.12975 0.05270 0.05270 0.05203 2.5854 0.32188 2.5854 0.30903
9 0.11223 0.03804 0.03804 0.03668 2.8727 0.35747 2.8727 0.34529
10 0.10664 0.02468 0.02468 0.02375 3.1600 0.39307 3.1600 0.38069
11 0.07122 0.01566 0.01566 0.01547 3.4473 0.42776 3.4473 0.41541
12 0.04717 0.00940 0.00940 0.00902 3.7345 0.46314 3.7345 0.45073
13 0.03039 0.00557 0.00557 0.00505 4.0218 0.49901 4.0218 0.48684
14 0.01920 0.00281 0.00281 0.00322 4.3091 0.53451 4.3091 0.52181
```

```
15 0.00876 0.00161 0.00161 0.00138 4.5963 0.57149 4.5963 0.55851
16 0.00615 0.00075 0.00075 0.00068 4.8836 0.60635 4.8836 0.59397
17 0.00242 0.00033 0.00033 0.00030 5.1709 0.64181 5.1709 0.62990
18 0.00168 0.00023 0.00023 0.00019 5.4582 0.67774 5.4582 0.66566
19 0.00037 0.00006 0.00006 0.00004 5.7454 0.71398 5.7454 0.70144
20 0.00056 0. 0. 0.00002 6.0327 0.74943 6.0327 0.73668
21 0. 0. 0. 0. 6.3200 0.78506 6.3200 0.77303
22 0. 0. 0. 0. 6.6073 0.82198 6.6073 0.80912
23 0. 0. 0. 0. 6.8945 0.85650 6.8945 0.84542
24 0. 0. 0. 0. 7.1818 0.89178 7.1818 0.87934
25 0. 0. 0. 0. 7.4691 0.92203 7.4691 0.91127
26 0. 0. 0. 0. 7.7563 0.94816 7.7563 0.93983
27 0. 0. 0. 0. 8.0436 0.96749 8.0436 0.96123
28 0. 0. 0. 0. 8.3309 0.98189 8.3309 0.97751
29 0. 0. 0. 0. 8.6182 0.99066 8.6182 0.98811
30 0. 0. 0. 0. 8.9054 1.00000 8.9054 1.00000
```

Async: priority-1 Normalized throughput (Rho_a)= 0.000100

```
J P(X=j) P(Xa=j) P(Xd=j) P(Xt=j) W P(W<=w) d P(D<=d)
0 0.45675 0.66068 0.66068 0.66750 0.2873 0.02931 0.2873 0.02910
1 0.35384 0.25417 0.25417 0.24536 0.5745 0.06119 0.5745 0.06098
2 0.14355 0.06889 0.06889 0.06937 0.8618 0.09093 0.8618 0.09050
3 0.00710 0.01305 0.01305 0.01484 1.1491 0.12602 1.1491 0.12580
4 0.00746 0.00257 0.00257 0.00235 1.4364 0.15233 1.4364 0.15233
5 0.00093 0.00064 0.00064 0.00047 1.7236 0.18335 1.7236 0.18335
6 0.00037 0. 0. 0.00012 2.0109 0.21823 2.0109 0.21823
7 0. 0. 0. 0. 2.2982 0.25160 2.2982 0.25160
8 0. 0. 0. 0. 2.5854 0.28712 2.5854 0.28712
9 0. 0. 0. 0. 2.8727 0.31750 2.8727 0.31750
10 0. 0. 0. 0. 3.1600 0.35387 3.1600 0.35387
11 0. 0. 0. 0. 3.4473 0.38682 3.4473 0.38661
12 0. 0. 0. 0. 3.7345 0.42319 3.7345 0.42319
13 0. 0. 0. 0. 4.0218 0.45614 4.0218 0.45614
14 0. 0. 0. 0. 4.3091 0.48866 4.3091 0.48845
15 0. 0. 0. 0. 4.5963 0.51583 4.5963 0.51583
16 0. 0. 0. 0. 4.8836 0.54472 4.8836 0.54472
17 0. 0. 0. 0. 5.1709 0.57167 5.1709 0.57167
18 0. 0. 0. 0. 5.4582 0.60398 5.4582 0.60398
19 0. 0. 0. 0. 5.7454 0.63586 5.7454 0.63586
20 0. 0. 0. 0. 6.0327 0.66645 6.0327 0.66624
21 0. 0. 0. 0. 6.3200 0.69619 6.3200 0.69619
22 0. 0. 0. 0. 6.6073 0.72743 6.6073 0.72743
23 0. 0. 0. 0. 6.8945 0.76444 6.8945 0.76444
24 0. 0. 0. 0. 7.1818 0.80210 7.1818 0.80210
25 0. 0. 0. 0. 7.4691 0.83248 7.4691 0.83248
26 0. 0. 0. 0. 7.7563 0.86757 7.7563 0.86757
27 0. 0. 0. 0. 8.0436 0.89795 8.0436 0.89773
28 0. 0. 0. 0. 8.3309 0.92255 8.3309 0.92255
29 0. 0. 0. 0. 8.6182 0.94651 8.6182 0.94651
30 0. 0. 0. 0. 8.9054 1.00000 8.9054 1.00000
```

Async: priority-2 Normalized throughput (Rho_a)= 0.000100

```
J P(X=j) P(Xa=j) P(Xd=j) P(Xt=j) W P(W<=w) d P(D<=d)
0 0.44836 0.66226 0.66226 0.65731 0.2873 0.03124 0.2873 0.03103
1 0.35682 0.25388 0.25388 0.25529 0.5745 0.06205 0.5745 0.06205
2 0.14933 0.06876 0.06876 0.07096 0.8618 0.09266 0.8618 0.09266
3 0.03766 0.01258 0.01258 0.01384 1.1491 0.12935 1.1491 0.12935
4 0.00652 0.00210 0.00210 0.00220 1.4364 0.16017 1.4364 0.16017
5 0.00112 0.00042 0.00042 0.00037 1.7236 0.19266 1.7236 0.19266
6 0.00019 0. 0. 0.00005 2.0109 0.22621 2.0109 0.22599
7 0. 0. 0. 0. 2.2982 0.25451 2.2982 0.25451
8 0. 0. 0. 0. 2.5854 0.28407 2.5854 0.28366
```

89/10/03
23:27:04

fddi.2.out

6

Async: priority-3										Normalized throughput (Rho_a)= 0.000100									
j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)		j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)	
0	0.4985	0.66157	0.66157	0.6232	0.2873	0.03039	0.2873	0.03018		1	0.36950	0.25285	0.25285	0.26208	0.5745	0.06495	0.5745	0.05609	
1	0.35682	0.25440	0.25440	0.25089	0.5745	0.06308	0.5745	0.06266		2	0.15735	0.07001	0.07001	0.06855	0.8618	0.09405	0.8618	0.08372	
2	0.14635	0.06790	0.06790	0.06941	0.8618	0.09807	0.8618	0.09807		3	0.05183	0.01075	0.01075	0.01530	1.1491	0.13075	1.1491	0.11852	
3	0.03934	0.01257	0.01257	0.01502	1.1491	0.12699	1.1491	0.12699		4	0.00746	0.00169	0.00169	0.00200	1.4364	0.16196	1.4364	0.15015	
4	0.00615	0.00293	0.00293	0.00191	1.4364	0.16157	1.4364	0.16136		5	0.00149	0.	0.	0.00023	1.7236	0.19338	1.7236	0.18368	
5	0.00130	0.00063	0.00063	0.00039	1.7236	0.19384	1.7236	0.19384		6	0.	0.	0.	0.	2.0109	0.22227	2.0109	0.21130	
6	0.00019	0.	0.	0.00006	2.0109	0.22590	2.0109	0.22590		7	0.	0.	0.	0.	2.2982	0.25517	2.2982	0.24420	
7	0.	0.	0.	0.	2.2982	0.25985	2.2982	0.25985		8	0.	0.	0.	0.	2.5854	0.29123	2.5854	0.27836	
8	0.	0.	0.	0.	2.5854	0.28877	2.5854	0.28856		9	0.	0.	0.	0.	2.8727	0.32391	2.8727	0.31189	
9	0.	0.	0.	0.	2.8727	0.32125	2.8727	0.32125		10	0.	0.	0.	0.	3.1600	0.35365	3.1600	0.34121	
10	0.	0.	0.	0.	3.1600	0.35289	3.1600	0.35289		11	0.	0.	0.	0.	3.4473	0.38887	3.4473	0.37853	
11	0.	0.	0.	0.	3.4473	0.38579	3.4473	0.38579		12	0.	0.	0.	0.	3.7345	0.41818	3.7345	0.40932	
12	0.	0.	0.	0.	3.7345	0.41995	3.7345	0.41995		13	0.	0.	0.	0.	4.0218	0.44918	4.0218	0.43948	
13	0.	0.	0.	0.	4.0218	0.45536	4.0218	0.45536		14	0.	0.	0.	0.	4.3091	0.47976	4.3091	0.46942	
14	0.	0.	0.	0.	4.3091	0.48806	4.3091	0.48785		15	0.	0.	0.	0.	4.5963	0.50907	4.5963	0.49873	
15	0.	0.	0.	0.	4.5963	0.52200	4.5963	0.52179		16	0.	0.	0.	0.	4.8836	0.53965	4.8836	0.52973	
16	0.	0.	0.	0.	4.8836	0.55113	4.8836	0.55092		17	0.	0.	0.	0.	5.1709	0.57739	5.1709	0.56495	
17	0.	0.	0.	0.	5.1709	0.58403	5.1709	0.58403		18	0.	0.	0.	0.	5.4582	0.61071	5.4582	0.59701	
18	0.	0.	0.	0.	5.4582	0.61798	5.4582	0.61777		19	0.	0.	0.	0.	5.7454	0.64256	5.7454	0.63075	
19	0.	0.	0.	0.	5.7454	0.64837	5.7454	0.64837		20	0.	0.	0.	0.	6.0327	0.67187	6.0327	0.66090	
20	0.	0.	0.	0.	6.0327	0.68148	6.0327	0.68106		21	0.	0.	0.	0.	6.3200	0.70751	6.3200	0.69633	
21	0.	0.	0.	0.	6.3200	0.71396	6.3200	0.71396		22	0.	0.	0.	0.	6.6073	0.74167	6.6073	0.72944	
22	0.	0.	0.	0.	6.6073	0.74728	6.6073	0.74728		23	0.	0.	0.	0.	6.8945	0.77119	6.8945	0.76086	
23	0.	0.	0.	0.	6.8945	0.78206	6.8945	0.78206		24	0.	0.	0.	0.	7.1818	0.79924	7.1818	0.79038	
24	0.	0.	0.	0.	7.1818	0.81496	7.1818	0.81454		25	0.	0.	0.	0.	7.4691	0.83572	7.4691	0.82223	
25	0.	0.	0.	0.	7.4691	0.84367	7.4691	0.84325		26	0.	0.	0.	0.	7.7563	0.87136	7.7563	0.85955	
26	0.	0.	0.	0.	7.7563	0.87091	7.7563	0.87091		27	0.	0.	0.	0.	8.0436	0.90173	8.0436	0.89140	
27	0.	0.	0.	0.	8.0436	0.90088	8.0436	0.90067		28	0.	0.	0.	0.	8.3309	0.92788	8.3309	0.91923	
28	0.	0.	0.	0.	8.3309	0.92435	8.3309	0.92435		29	0.	0.	0.	0.	8.6182	0.95002	8.6182	0.94348	
29	0.	0.	0.	0.	8.6182	0.94761	8.6182	0.94761		30	0.	0.	0.	0.	8.9054	1.00000	8.9054	1.00000	
30	0.	0.	0.	0.	8.9054	1.00000	8.9054	1.00000											

Station 5

Synchronous Traffic: Normalized throughput (Rho_a)= 0.010000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
0	0.41238	0.66470	0.66470	0.65184	0.2873	0.03290	0.2873	0.02193

Async: priority-1										Normalized throughput (Rho_a)= 0.000010									
j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)		j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)	
0	0.91723	0.96767	0.96767	0.95811	0.2873	0.02586	0.2873	0.02586		1	0.36950	0.25285	0.25285	0.26208	0.5745	0.06495	0.5745	0.05609	
1	0.07998	0.03233	0.03233	0.04091	0.5745	0.06034	0.5745	0.06034		2	0.15735	0.07001	0.07001	0.06855	0.8618	0.09405	0.8618	0.08372	
2	0.00280	0.	0.	0.00099	0.8618	0.09483	0.8618	0.09483		3	0.05183	0.01075	0.01075	0.01530	1.1491	0.13075	1.1491	0.11852	
3	0.	0.	0.	0.	1.1491	0.11853	1.1491	0.11853		4	0.00746	0.00169	0.00169	0.00200	1.4364	0.16196	1.4364	0.15015	
4	0.	0.	0.	0.	1.4364	0.15948	1.4364	0.15948		5	0.00149	0.	0.	0.	1.7236	0.19338	1.7236	0.18368	
5	0.	0.	0.	0.	1.7236	0.18750	1.7236	0.18750		6	0.	0.	0.	0.	2.0109	0.22227	2.0109	0.21130	
6	0.	0.	0.	0.	2.0109	0.21983	2.0109	0.21983		7	0.	0.	0.	0.	2.2982	0.25517	2.2982	0.24420	
7	0.	0.	0.	0.	2.2982	0.24569	2.2982	0.24569		8	0.	0.	0.	0.	2.5854	0.29123	2.5854	0.27836	
8	0.	0.	0.	0.	2.5854	0.30172	2.5854	0.30172		9	0.	0.	0.	0.	2.8727	0.32391	2.8727	0.31189	
9	0.	0.	0.	0.	2.8727	0.34267	2.8727	0.34267		10	0.	0.	0.	0.	3.1600	0.35365	3.1600	0.34121	
10	0.	0.	0.	0.	3.1600	0.36638	3.1600	0.36638		11	0.	0.	0.	0.	3.4473	0.38887	3.4473	0.37853	
11	0.	0.	0.	0.	3.4473	0.38793	3.4473	0.38793		12	0.	0.	0.	0.	3.7345	0.41818	3.7345	0.40932	
12	0.	0.	0.	0.	3.7345	0.41810	3.7345	0.41810		13	0.	0.	0.	0.	4.0218	0.44918	4.0218	0.43948	
13	0.	0.	0.	0.	4.0218	0.46121	4.0218	0.46121		14	0.	0.	0.	0.	4.3091	0.47976	4.3091	0.46942	
14	0.	0.	0.	0.	4.3091	0.48922	4.3091	0.48922		15	0.	0.	0.	0.	4.5963	0.50907	4.5963	0.49873	
15	0.	0.	0.	0.	4.5963	0.53233	4.5963	0.53233		16	0.	0.	0.	0.	4.8836	0.53965	4.8836	0.52973	
16	0.	0.	0.	0.	4.8836	0.55388	4.8836	0.55388		17	0.	0.	0.	0.	5.1709	0.57739	5.1709	0.56495	
17	0.	0.	0.	0.	5.1709	0.58836	5.1709	0.58836		18	0.	0.	0.	0.	5.4582	0.61071	5.4582	0.59701	
18	0.	0.	0.	0.	5.4582	0.62931	5.4582	0.62931		19	0.	0.	0.	0.	5.7454	0.64256	5.7454	0.63075	
19	0.	0.	0.	0.	5.7454	0.67672	5.7454	0.67672		20	0.	0.	0.	0.	6.0327	0.67187	6.0327	0.66090	
20	0.	0.	0.	0.	6.0327	0.70043	6.0327	0.70043		21	0.	0.	0.	0.	6.3200	0.70751	6.3200	0.69633	
21	0.	0.	0.	0.	6.3200	0.73922	6.3200	0.73922		22	0.	0.	0.	0.	6.6073	0.74167	6.6073	0.72944	
22	0.	0.	0.	0.	6.6073	0.76078	6.6073	0.76078		23	0.	0.	0.	0.	6.8945	0.77119	6.8945	0.76086	
23	0.	0.	0.	0.	6.8945	0.78448	6.8945	0.78448		24	0.	0.	0.	0.	7.1818	0.79924	7.1818	0.79038	
24	0.	0.	0.	0.	7.1818	0.80388	7.1818	0.80388		25	0.	0.	0.	0.	7.4691	0.83572	7.4691	0.82223	
25	0.	0.	0.	0.	7.4691	0.84052	7.4691	0.84052		26	0.	0.	0.	0.	7.7563	0.87136	7.7563	0.85955	
26	0.	0.	0.	0.	7.7563	0.86853	7.7563	0.86853		27	0.	0.	0.	0.	8.0436	0.90173	8.0436	0.89140	
27	0.	0.	0.	0.	8.0436	0.90302	8.0436	0.90302		28	0.	0.	0.	0.	8.3309	0.92788	8.3309	0.91923	
28	0.	0.	0.	0.	8.3309	0.92241	8.3309	0.92241		29	0.	0.	0.	0.	8.6182	0.95002	8.6182	0.94348	
29	0.	0.	0.	0.	8.6182	0.93966	8.6182	0.93966		30	0.	0.	0.	0.	8.9054	1.00000	8.9054	1.00000	

89/10/03
23:27:04

fddi.2.out

7

Synchronous Traffic: Normalized throughput (Rho_a) = 0.100000									
Station 6									
j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)	
0	0.91667	0.94748	0.94748	0.95678	0.2873	0.03361	0.2873	0.03361	0.68026
1	0.07867	0.05252	0.05252	0.04156	0.5745	0.06513	0.5745	0.06513	0.72103
2	0.00466	0.	0.	0.00166	0.8618	0.10084	0.8618	0.10084	0.74678
3	0.	0.	0.	0.	1.1491	0.12815	1.1491	0.12815	0.77039
4	0.	0.	0.	0.	1.4364	0.15126	1.4364	0.15126	0.79399
5	0.	0.	0.	0.	1.7236	0.18697	1.7236	0.18697	0.81116
6	0.	0.	0.	0.	2.0109	0.21849	2.0109	0.21849	0.83309
7	0.	0.	0.	0.	2.2982	0.24580	2.2982	0.24580	0.84618
8	0.	0.	0.	0.	2.5854	0.27311	2.5854	0.27311	0.85418
9	0.	0.	0.	0.	2.8727	0.30882	2.8727	0.30882	0.86182
10	0.	0.	0.	0.	3.1600	0.34244	3.1600	0.34244	0.86827
11	0.	0.	0.	0.	3.4473	0.36975	3.4473	0.36975	0.87454
12	0.	0.	0.	0.	3.7345	0.40126	3.7345	0.40126	0.88081
13	0.	0.	0.	0.	4.0218	0.43067	4.0218	0.43067	0.88712
14	0.	0.	0.	0.	4.3091	0.45798	4.3091	0.45798	0.89349
15	0.	0.	0.	0.	4.5963	0.49370	4.5963	0.49370	0.90000
16	0.	0.	0.	0.	4.8836	0.53151	4.8836	0.53151	0.90626
17	0.	0.	0.	0.	5.1709	0.57563	5.1709	0.57563	0.91253
18	0.	0.	0.	0.	5.4582	0.60924	5.4582	0.60924	0.91882
19	0.	0.	0.	0.	5.7454	0.64916	5.7454	0.64916	0.92511
20	0.	0.	0.	0.	6.0327	0.67857	6.0327	0.67857	0.93140
21	0.	0.	0.	0.	6.3200	0.71639	6.3200	0.71639	0.93769
22	0.	0.	0.	0.	6.6073	0.74790	6.6073	0.74790	0.94398
23	0.	0.	0.	0.	6.8945	0.76891	6.8945	0.76891	0.95027
24	0.	0.	0.	0.	7.1818	0.78992	7.1818	0.78992	0.95656
25	0.	0.	0.	0.	7.4691	0.82353	7.4691	0.82353	0.96285
26	0.	0.	0.	0.	7.7563	0.86765	7.7563	0.86765	0.96914
27	0.	0.	0.	0.	8.0436	0.89706	8.0436	0.89706	0.97543
28	0.	0.	0.	0.	8.3309	0.92437	8.3309	0.92437	0.98172
29	0.	0.	0.	0.	8.6182	0.95168	8.6182	0.95168	0.98801
30	0.	0.	0.	0.	8.9054	1.00000	8.9054	1.00000	0.99430

Synchronous Traffic: Normalized throughput (Rho_a) = 0.100000									
Station 6									
j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)	
0	0.90250	0.95279	0.95279	0.93977	0.2873	0.03863	0.2873	0.03863	0.68026
1	0.09284	0.04721	0.04721	0.05805	0.5745	0.06009	0.5745	0.06009	0.72103
2	0.00466	0.	0.	0.00218	0.8618	0.07725	0.8618	0.07725	0.74678
3	0.	0.	0.	0.	1.1491	0.09227	1.1491	0.09227	0.77039
4	0.	0.	0.	0.	1.4364	0.12661	1.4364	0.12661	0.79399
5	0.	0.	0.	0.	1.7236	0.13734	1.7236	0.13734	0.81116
6	0.	0.	0.	0.	2.0109	0.16524	2.0109	0.16524	0.83309
7	0.	0.	0.	0.	2.2982	0.19313	2.2982	0.19313	0.84618
8	0.	0.	0.	0.	2.5854	0.23820	2.5854	0.23820	0.85418
9	0.	0.	0.	0.	2.8727	0.26824	2.8727	0.26824	0.86182
10	0.	0.	0.	0.	3.1600	0.30472	3.1600	0.30472	0.86827
11	0.	0.	0.	0.	3.4473	0.32189	3.4473	0.32189	0.87454
12	0.	0.	0.	0.	3.7345	0.34120	3.7345	0.34120	0.88081
13	0.	0.	0.	0.	4.0218	0.37983	4.0218	0.37983	0.88712
14	0.	0.	0.	0.	4.3091	0.40558	4.3091	0.40558	0.89349
15	0.	0.	0.	0.	4.5963	0.42918	4.5963	0.42918	0.90000
16	0.	0.	0.	0.	4.8836	0.46352	4.8836	0.46352	0.90626
17	0.	0.	0.	0.	5.1709	0.48927	5.1709	0.48927	0.91253
18	0.	0.	0.	0.	5.4582	0.50858	5.4582	0.50858	0.91882
19	0.	0.	0.	0.	5.7454	0.53863	5.7454	0.53863	0.92511
20	0.	0.	0.	0.	6.0327	0.56438	6.0327	0.56438	0.93140
21	0.	0.	0.	0.	6.3200	0.59657	6.3200	0.59657	0.93769
22	0.	0.	0.	0.	6.6073	0.62661	6.6073	0.62661	0.94398
23	0.	0.	0.	0.	6.8945	0.65451	6.8945	0.65451	0.95027

Async: priority-3 Normalized throughput (Rho_a) = 0.000100

Synchronous Traffic: Normalized throughput (Rho_a) = 0.100000									
Station 6									
j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)	
0	0.90250	0.95279	0.95279	0.93977	0.2873	0.03863	0.2873	0.03863	0.68026
1	0.09284	0.04721	0.04721	0.05805	0.5745	0.06009	0.5745	0.06009	0.72103
2	0.00466	0.	0.	0.00218	0.8618	0.07725	0.8618	0.07725	0.74678
3	0.	0.	0.	0.	1.1491	0.09227	1.1491	0.09227	0.77039
4	0.	0.	0.	0.	1.4364	0.12661	1.4364	0.12661	0.79399
5	0.	0.	0.	0.	1.7236	0.13734	1.7236	0.13734	0.81116
6	0.	0.	0.	0.	2.0109	0.16524	2.0109	0.16524	0.83309
7	0.	0.	0.	0.	2.2982	0.19313	2.2982	0.19313	0.84618
8	0.	0.	0.	0.	2.5854	0.23820	2.5854	0.23820	0.85418
9	0.	0.	0.	0.	2.8727	0.26824	2.8727	0.26824	0.86182
10	0.	0.	0.	0.	3.1600	0.30472	3.1600	0.30472	0.86827
11	0.	0.	0.	0.	3.4473	0.32189	3.4473	0.32189	0.87454
12	0.	0.	0.	0.	3.7345	0.34120	3.7345	0.34120	0.88081
13	0.	0.	0.	0.	4.0218	0.37983	4.0218	0.37983	0.88712
14	0.	0.	0.	0.	4.3091	0.40558	4.3091	0.40558	0.89349
15	0.	0.	0.	0.	4.5963	0.42918	4.5963	0.42918	0.90000
16	0.	0.	0.	0.	4.8836	0.46352	4.8836	0.46352	0.90626
17	0.	0.	0.	0.	5.1709	0.48927	5.1709	0.48927	0.91253
18	0.	0.	0.	0.	5.4582	0.50858	5.4582	0.50858	0.91882
19	0.	0.	0.	0.	5.7454	0.53863	5.7454	0.53863	0.92511
20	0.	0.	0.	0.	6.0327	0.56438	6.0327	0.56438	0.93140
21	0.	0.	0.	0.	6.3200	0.59657	6.3200	0.59657	0.93769
22	0.	0.	0.	0.	6.6073	0.62661	6.6073	0.62661	0.94398
23	0.	0.	0.	0.	6.8945	0.65451	6.8945	0.65451	0.95027

Async: priority-1 Normalized throughput (Rho_a) = 0.000100

Synchronous Traffic: Normalized throughput (Rho_a) = 0.100000									
Station 6									
j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)	
0	0.90250	0.95279	0.95279	0.93977	0.2873	0.03863	0.2873	0.03863	0.68026
1	0.09284	0.04721	0.04721	0.05805	0.5745	0.06009	0.5745	0.06009	0.72103
2	0.00466	0.	0.	0.00218	0.8618	0.07725	0.8618	0.07725	0.74678
3	0.	0.	0.	0.	1.1491	0.09227	1.1491	0.09227	0.77039
4	0.	0.	0.	0.	1.4364	0.12661	1.4364	0.12661	0.79399
5	0.	0.	0.	0.	1.7236	0.13734	1.7236	0.13734	0.81116
6	0.	0.	0.	0.	2.0109	0.16524	2.0109	0.16524	0.83309
7	0.	0.	0.	0.	2.2982	0.19313	2.2982	0.19313	0.84618
8	0.	0.	0.	0.	2.5854	0.23820	2.5854	0.23820	0.85418
9	0.	0.	0.	0.	2.8727	0.26824	2.8727	0.26824	0.86182
10	0.	0.	0.	0.	3.1600	0.30472	3.1600	0.30472	0.86827
11	0.	0.	0.	0.	3.4473	0.32189	3.4473	0.32189	0.87454
12	0.	0.	0.	0.	3.7345	0.34120	3.7345	0.34120	0.88081
13	0.	0.	0.	0.	4.0218	0.37983	4.0218	0.37983	0.88712
14	0.	0.	0.	0.	4.3091	0.40558	4.3091	0.40558	0.89349
15	0.	0.	0.	0.	4.5963	0.42918	4.5963	0.42918	0.90000
16	0.	0.	0.	0.	4.8836	0.46352	4.8836	0.46352	0.90626
17	0.	0.	0.	0.	5.1709	0.48927	5.1709	0.48927	0.91253
18	0.	0.	0.	0.	5.4582	0.50858	5.4582	0.50858	0.91882
19	0.	0.	0.	0.	5.7454	0.53863	5.7454	0.53863	0.92511
20	0.	0.	0.	0.	6.0327	0.56438	6.0327	0.56438	0.93140
21	0.	0.	0.	0.	6.3200	0.59657	6.3200	0.59657	0.93769
22	0.	0.	0.	0.	6.6073	0.62661	6.6073	0.62661	0.94398
23	0.	0.	0.	0.	6.8945	0.65451	6.8945	0.65451	0.95027

fddi.2.out

[illegible]

Async: priority-2 Normalized throughput (Rho a)= 0.000100

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
0	0.4444	0.65970	0.65970	0.65444	0.2873	0.02925	0.2873	0.02904
1	0.36503	0.24838	0.24838	0.25706	0.5745	0.05933	0.5745	0.05933
2	0.13889	0.07270	0.07270	0.08816	0.8618	0.08920	0.8618	0.08899
3	0.04157	0.01650	0.01650	0.01661	1.1491	0.12179	1.1491	0.12179
4	0.00839	0.00230	0.00230	0.00247	1.4364	0.15479	1.4364	0.15459
5	0.00168	0.00042	0.00042	0.00043	1.7236	0.18676	1.7236	0.18634
6	0.	0.	0.	0.00003	2.0109	0.21872	2.0109	0.21872
7	0.	0.	0.	0.	2.2982	0.24734	2.2982	0.24733
8	0.	0.	0.	0.	2.5854	0.27909	2.5854	0.27888
9	0.	0.	0.	0.	2.8727	0.31335	2.8727	0.31272
10	0.	0.	0.	0.	3.1600	0.34886	3.1600	0.34866
11	0.	0.	0.	0.	3.4473	0.38020	3.4473	0.37999
12	0.	0.	0.	0.	3.7345	0.41111	3.7345	0.41111
13	0.	0.	0.	0.	4.0218	0.44161	4.0218	0.44161
14	0.	0.	0.	0.	4.3091	0.47357	4.3091	0.47357
15	0.	0.	0.	0.	4.5963	0.50261	4.5963	0.50261
16	0.	0.	0.	0.	4.8836	0.53541	4.8836	0.53520
17	0.	0.	0.	0.	5.1709	0.56612	5.1709	0.56612
18	0.	0.	0.	0.	5.4582	0.59662	5.4582	0.59662
19	0.	0.	0.	0.	5.7454	0.62774	5.7454	0.62774
20	0.	0.	0.	0.	6.0327	0.66305	6.0327	0.66284
21	0.	0.	0.	0.	6.3200	0.69522	6.3200	0.69522
22	0.	0.	0.	0.	6.6073	0.72927	6.6073	0.72927
23	0.	0.	0.	0.	6.8945	0.76332	6.8945	0.76290
24	0.	0.	0.	0.	7.1818	0.79549	7.1818	0.79528
25	0.	0.	0.	0.	7.4691	0.83037	7.4691	0.83017
26	0.	0.	0.	0.	7.7563	0.86087	7.7563	0.86087
27	0.	0.	0.	0.	8.0436	0.89325	8.0436	0.89304
28	0.	0.	0.	0.	8.3309	0.91728	8.3309	0.91707
29	0.	0.	0.	0.	8.6182	0.93963	8.6182	0.93963
30	0.	0.	0.	0.	8.9054	1.00000	8.9054	1.00000

Async: priority-3 Normalized throughput (Rho a) = 0.000100

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(DC<d)
0	0.43312	0.65717	0.65717	0.65300	0.2873	0.03237	0.2873	0.03237
1	0.36745	0.25326	0.25326	0.25926	0.5745	0.07105	0.5745	0.07105
2	0.14896	0.07064	0.07064	0.07005	0.8618	0.10313	0.8618	0.10342
3	0.03934	0.01588	0.01588	0.01497	1.1491	0.13864	1.1491	0.13844
4	0.00746	0.00265	0.00265	0.00320	1.4364	0.16796	1.4364	0.16796
5	0.00149	0.00020	0.00020	0.00046	1.7236	0.19707	1.7236	0.19707
6	0.	0.00020	0.00020	0.00004	2.0109	0.22842	2.0109	0.22842
7	0.00019	0.	0.	0.00002	2.2982	0.26344	2.2982	0.26323
8	0.	0.	0.	0.	2.5854	0.29397	2.5854	0.29397
9	0.	0.	0.	0.	2.8721	0.32288	2.8721	0.32288

Delay-Throughput Evaluator, IRI Corp.

Appendix I.

The FDDI Program: Asymmetric Case Example

89/10/04
10:43:02

fddi.3.inp

1

** Simulation of a Timed Token Rotation Protocol (FDDI-I Type)
program: FDDI
Professor Izhak Rubin, UCLA

Enter Select Input Mode

1. from KEYBOARD
2. from DATA FILE

1

Enter the output data file name

fddi.3.out

Enter Feature Selection

1. Symmetric System
2. 2 Classes of Stations
3. Different Loading from Station to Station

3

Enter the statistics collection start time (msec)

400

Enter the stop time (msec)

2000

Enter w,d,x, for computing $P(W>w)$, $P(D>d)$, $P(X>x)$

1 1 1

Enter the number of stations (N)

6

Enter walk times from station to station (msec) ($r(i), i=1..N$)

0.1 0.1 0.1 0.1 0.1 0.1

Enter Target Token Rotation Time (msec; TTRT)

20

Synchronous traffic arrival rates (packets/msec/station; $as(i), i=1..N$)

0.02 0.2 0.02 0.2 0.02 0.2

Mean synchronous packet transmission time (msec; $plens(i), i=1..N$)

0.5 0.5 0.5 0.5 0.5 0.5

Enter the bandwidth times (msec; $<TTRT-walktime$) ($BWT(i), i=1..N$)

3 3 3 3 3 3

Asynchronous Traffic

Enter number of message priority classes per station (N_p)

3

Enter priority-1 threshold for stations 1..N (msec)

($T_{pri}(i,1), i=1..N$)

10 10 10 10 10 10

Enter priority-1 arrival rate for stations 1..N (packets/msec/station)

($aa(i,1), i=1..N$)

0.1 0.1 0.1 0.1 0.1 0.1

Enter priority-1 mean transmission time for stations 1..N (msec)

($plena(i,1), i=1..N$)

0.3 0.3 0.3 0.3 0.3 0.3

Enter priority-2 threshold for stations 1..N (msec)

($T_{pri}(i,2), i=1..N$)

7.65 7.65 7.65 7.65 7.65 7.65

Enter priority-2 arrival rate for stations 1..N (packets/msec/station)

($aa(i,2), i=1..N$)

0.1 0.1 0.1 0.1 0.1 0.1

Enter priority-2 mean transmission time for stations 1..N (msec)

($plena(i,2), i=1..N$)

0.3 0.3 0.3 0.3 0.3 0.3

Enter priority-3 threshold for stations 1..N (msec)

($T_{pri}(i,3), i=1..N$)

5.62 5.62 5.62 5.62 5.62 5.62

Enter priority-3 arrival rate for stations 1..N (packets/msec/station)

($aa(i,3), i=1..N$)

0.1 0.1 0.1 0.1 0.1 0.1

Enter priority-3 mean transmission time for stations 1..N (msec)

($plena(i,3), i=1..N$)

0.3 0.3 0.3 0.3 0.3 0.3

89/10/04
11:04:02

fddi.inp3

1

This is an example of the input file for different loading from station to station.

	output file
ff9.31	ifeat
3	c1
400	c2
2000	W0,D0,X0
1 1 1	N
6	Np
3	TTRT
200	r, BWT
0.1 30	as
0.1	plens
0.5	T_pri(1)
100. 76.5 56.2	aa(1)
0.1 0.1 0.1	plena(1)
0.3 0.3 0.3	r, BWT
0.1 30	as
0.1	plens
0.5	T_pri(2)
100. 76.5 56.2	aa(1)
0.1 0.1 0.1	plena(2)
0.3 0.3 0.3	r, BWT
0.1 30	as
0.1	plens
0.5	T_pri(3)
100. 76.5 56.2	aa(3)
0.1 0.1 0.1	plena(3)
0.3 0.3 0.3	r, BWT
0.1 30	as
0.1	plens
0.5	T_pri(4)
100. 76.5 56.2	aa(4)
0.1 0.1 0.1	plena(4)
0.3 0.3 0.3	r, BWT
0.1 30	as
0.1	plens
0.5	T_pri(5)
100. 76.5 56.2	aa(5)
0.1 0.1 0.1	plena(5)
0.3 0.3 0.3	r, BWT
0.1 30	as
0.1	plens
0.5	T_pri(6)
100. 76.5 56.2	aa(6)
0.1 0.1 0.1	plena(6)
0.3 0.3 0.3	

89/10/04
10:27:42

.. Performance of a Timed Token Rotation Protocol (FDDI-type) Ring Networks ..

Feature Selected: Different Loading from Station to Station

Statistics Start (msec): 400.0
Statistics Stop (msec): 2000.0
TTRT (msec): 20.000
Number of Stations (N): 6
Number of Priorities (Np): 3
Max Throughput (1-walktime/2TTRT): 0.9850
Normalized Throughput (specified): 0.8700
Normalized Throughput (realized): 0.8714
Realized Mean Cycle Time (msec): 4.6646
Realized Mean Dwell Time (msec): 4.0646
Max Cycle Time (msec): 10.6550 at 437th cycle, t= 1406.3

Station 1
Walk Time (r;msec): 0.1000
Bandwidth Time (BWT;msec): 3.0000
Arrival Rate for Synchronous Traffic (packets/msec/station): 0.0200
Mean Packet Length for Synchronous Traffic (msec): 0.5000
Asynchronous Traffic: Priority-j T_pri aa(1,j) plana(1,j)
1 10.0000 0.1000 0.3000
2 7.6500 0.1000 0.3000
3 5.6200 0.1000 0.3000

Station 2
Walk Time (r;msec): 0.1000
Bandwidth Time (BWT;msec): 3.0000
Arrival Rate for Synchronous Traffic (packets/msec/station): 0.2000
Mean Packet Length for Synchronous Traffic (msec): 0.5000
Asynchronous Traffic: Priority-j T_pri aa(2,j) plana(2,j)
1 10.0000 0.1000 0.3000
2 7.6500 0.1000 0.3000
3 5.6200 0.1000 0.3000

Station 3
Walk Time (r;msec): 0.1000
Bandwidth Time (BWT;msec): 3.0000
Arrival Rate for Synchronous Traffic (packets/msec/station): 0.0200
Mean Packet Length for Synchronous Traffic (msec): 0.5000
Asynchronous Traffic: Priority-j T_pri aa(3,j) plana(3,j)
1 10.0000 0.1000 0.3000
2 7.6500 0.1000 0.3000
3 5.6200 0.1000 0.3000

Station 4
Walk Time (r;msec): 0.1000
Bandwidth Time (BWT;msec): 3.0000
Arrival Rate for Synchronous Traffic (packets/msec/station): 0.2000
Mean Packet Length for Synchronous Traffic (msec): 0.5000
Asynchronous Traffic: Priority-j T_pri aa(4,j) plana(4,j)
1 10.0000 0.1000 0.3000
2 7.6500 0.1000 0.3000
3 5.6200 0.1000 0.3000

Station 5
Walk Time (r;msec): 0.1000
Bandwidth Time (BWT;msec): 3.0000
Arrival Rate for Synchronous Traffic (packets/msec/station): 0.0200
Mean Packet Length for Synchronous Traffic (msec): 0.5000
Asynchronous Traffic: Priority-j T_pri aa(5,j) plana(5,j)
1 10.0000 0.1000 0.3000
2 7.6500 0.1000 0.3000

fddi.3.out

Station 6
Walk Time (r;msec): 0.1000
Bandwidth Time (BWT;msec): 3.0000
Arrival Rate for Synchronous Traffic (packets/msec/station): 0.2000
Mean Packet Length for Synchronous Traffic (msec): 0.5000
Asynchronous Traffic: Priority-j T_pri aa(6,j) plana(6,j)
1 10.0000 0.1000 0.3000
2 7.6500 0.1000 0.3000
3 5.6200 0.1000 0.3000

Token goes 545 cycles in simulation

	E(X)	sigma(X)	E(W)	sigma(W)	E(D)	sigma(D)
Sync Traffic:	0.4329	0.7841	2.2001	1.6257	2.6979	1.6397
Async: pri-1	0.4048	0.6553	2.5103	2.0447	2.8118	2.0529
Async: pri-2	0.5311	0.8702	4.1504	5.3985	4.4505	5.4005
Async: pri-3	1.8586	2.6963	20.1408	23.8711	20.4406	23.8720

Station #stat E(X) sigma(X) E(W) sigma(W) E(D) sigma(D)

1	0	30	0.0671	0.2501	2.1202	1.7470	2.5760	1.7813
1	167	0.4286	0.6657	2.4317	1.9576	2.7144	1.9580	
2	184	0.5948	0.8515	3.8284	3.6785	4.1199	3.7049	
3	160	1.0321	1.1960	9.7785	11.1440	10.0673	11.1354	
2	0	312	0.7697	0.9334	2.1673	1.5473	2.6742	1.5496
1	158	0.3965	0.6436	2.5650	2.0375	2.5835	2.0481	
2	149	0.5423	0.9855	5.3539	6.9132	5.6558	6.9095	
3	144	1.7580	2.4309	21.2985	19.5877	21.5955	19.5771	
3	0	30	0.0816	0.2738	2.2715	1.5703	2.7028	1.5583
1	161	0.4344	0.6750	2.4382	1.9358	2.7381	1.9432	
2	156	0.6694	0.7510	3.1543	2.8005	3.4598	2.7971	
3	154	1.2099	1.5395	12.7581	15.7085	13.0345	15.6869	
4	0	324	0.7959	0.9063	2.2862	1.6912	2.7668	1.7181
1	157	0.3586	0.6499	2.4108	1.9640	2.7257	1.9615	
2	146	0.5481	0.9912	5.3262	8.5023	5.6276	8.5025	
3	164	2.5131	3.1915	27.2296	24.5210	27.5347	24.5346	
5	0	36	0.0933	0.2908	2.2322	1.6747	2.8055	1.7465
1	147	0.3994	0.6619	2.4359	1.8619	2.7242	1.8846	
2	164	0.5452	0.8657	3.5669	3.4962	3.8637	3.4874	
3	184	1.2449	1.4801	10.1013	11.0538	10.4213	11.0500	
6	0	329	0.7901	1.0143	2.1445	1.6195	2.6536	1.6260
1	159	0.4111	0.6321	2.7804	2.4346	3.0846	2.4440	
2	168	0.4869	0.7358	3.6729	4.0850	3.9763	4.0925	
3	158	2.3936	4.1544	39.6789	35.2829	39.9901	35.2868	

Station E(Vj) sigma(Vj) Pr(W> 1.000) Pr(D> 1.000) Pr(X> 1)

1	0	0.03986	0.16738	0.66667	0.60000	0.
1	0	0.13763	0.24348	0.68263	0.74850	0.07872
2	0	0.15636	0.29269	0.79348	0.82609	0.13703
3	0	0.13474	0.26572	0.84375	0.86250	0.27114
2	0	0.45945	0.63981	0.85897	0.72436	0.19825
1	0	0.14673	0.25708	0.75949	0.83544	0.06997
2	0	0.13114	0.28979	0.73826	0.79195	0.09913

89/10/04
10:27:42

fddi.3.out

Station	E(V)	sigma(V)	E(C)	sigma(C)	E(C-V)	sigma(C-V)
1	0.4685953	0.4977408	4.6654343	2.0395754	4.1968390	1.6236669
2	0.8636660	0.8079839	4.6645960	2.0380379	3.8009300	1.9750534
3	0.4415433	0.4830993	4.6620234	2.0275840	4.2204800	1.8981629
4	0.8721908	0.7571019	4.6613917	2.0754535	3.7892009	1.7134797
5	0.4973797	0.5005073	4.6629486	2.0389226	4.1655689	1.9125625
6	0.9211771	0.8066889	4.6645523	2.1039462	3.7433751	1.7127622

Station 1

Station	E(V)	sigma(V)	E(C)	sigma(C)	E(C-V)	sigma(C-V)
1	0.4685953	0.4977408	4.6654343	2.0395754	4.1968390	1.6236669
2	0.8636660	0.8079839	4.6645960	2.0380379	3.8009300	1.9750534
3	0.4415433	0.4830993	4.6620234	2.0275840	4.2204800	1.8981629
4	0.8721908	0.7571019	4.6613917	2.0754535	3.7892009	1.7134797
5	0.4973797	0.5005073	4.6629486	2.0389226	4.1655689	1.9125625
6	0.9211771	0.8066889	4.6645523	2.1039462	3.7433751	1.7127622

Station 1

Synchronous Traffic: Normalized throughput (Rho_s) = 0.010000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
0	0.93294	0.96667	0.96667	0.95987	0.3886	0.16667	0.3886	0.03333
1	0.06706	0.03333	0.03333	0.03973	0.7772	0.33333	0.7772	0.13333
2	0.	0.	0.	0.00040	1.1658	0.43333	1.1658	0.33333
3	0.	0.	0.	0.	1.5543	0.66667	1.5543	0.40000
4	0.	0.	0.	0.	1.9429	0.53333	1.9429	0.46667
5	0.	0.	0.	0.	2.3315	0.60000	2.3315	0.53333
6	0.	0.	0.	0.	2.7201	0.63333	2.7201	0.60000
7	0.	0.	0.	0.	3.1087	0.63333	3.1087	0.60000
8	0.	0.	0.	0.	3.4973	0.73333	3.4973	0.66667
9	0.	0.	0.	0.	3.8859	0.76667	3.8859	0.73333
10	0.	0.	0.	0.	4.2744	0.90000	4.2744	0.76667
11	0.	0.	0.	0.	4.6630	0.90000	4.6630	0.83333
12	0.	0.	0.	0.	5.0516	0.90000	5.0516	0.90000
13	0.	0.	0.	0.	5.4402	0.96667	5.4402	0.93333
14	0.	0.	0.	0.	5.8288	0.96667	5.8288	0.93333
15	0.	0.	0.	0.	6.2174	1.00000	6.2174	0.96667
16	0.	0.	0.	0.	6.6060	1.00000	6.6060	1.00000
17	0.	0.	0.	0.	6.9945	1.00000	6.9945	1.00000
18	0.	0.	0.	0.	7.3831	1.00000	7.3831	1.00000
19	0.	0.	0.	0.	7.7717	1.00000	7.7717	1.00000
20	0.	0.	0.	0.	8.1603	1.00000	8.1603	1.00000
21	0.	0.	0.	0.	8.5489	1.00000	8.5489	1.00000
22	0.	0.	0.	0.	8.9375	1.00000	8.9375	1.00000
23	0.	0.	0.	0.	9.3261	1.00000	9.3261	1.00000
24	0.	0.	0.	0.	9.7146	1.00000	9.7146	1.00000
25	0.	0.	0.	0.	10.1032	1.00000	10.1032	1.00000
26	0.	0.	0.	0.	10.4918	1.00000	10.4918	1.00000
27	0.	0.	0.	0.	10.8804	1.00000	10.8804	1.00000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
0	0.65889	0.77246	0.77246	0.76721	0.3886	0.14970	0.3886	0.04192
1	0.26239	0.19760	0.19760	0.18637	0.7772	0.25749	0.7772	0.17365
2	0.07289	0.02395	0.02395	0.04438	1.1658	0.33533	1.1658	0.28743
3	0.00292	0.00599	0.00599	0.00196	1.5543	0.41916	1.5543	0.37126
4	0.00292	0.	0.	0.00007	1.9429	0.49701	1.9429	0.43114
5	0.	0.	0.	0.	2.3315	0.55689	2.3315	0.52695
6	0.	0.	0.	0.	2.7201	0.62275	2.7201	0.56886
7	0.	0.	0.	0.	3.1087	0.66467	3.1087	0.64072
8	0.	0.	0.	0.	3.4973	0.69461	3.4973	0.67066
9	0.	0.	0.	0.	3.8859	0.74850	3.8859	0.72455
10	0.	0.	0.	0.	4.2744	0.79641	4.2744	0.77246
11	0.	0.	0.	0.	4.6630	0.85629	4.6630	0.82635
12	0.	0.	0.	0.	5.0516	0.88024	5.0516	0.85030
13	0.	0.	0.	0.	5.4402	0.92216	5.4402	0.89820
14	0.	0.	0.	0.	5.8288	0.94012	5.8288	0.92216
15	0.	0.	0.	0.	6.2174	0.95210	6.2174	0.94012
16	0.	0.	0.	0.	6.6060	0.97006	6.6060	0.96407
17	0.	0.	0.	0.	6.9945	0.98204	6.9945	0.97605
18	0.	0.	0.	0.	7.3831	0.98802	7.3831	0.98204
19	0.	0.	0.	0.	7.7717	0.99401	7.7717	0.98802
20	0.	0.	0.	0.	8.1603	0.99401	8.1603	0.99401
21	0.	0.	0.	0.	8.5489	0.99401	8.5489	0.99401
22	0.	0.	0.	0.	8.9375	1.00000	8.9375	0.99401
23	0.	0.	0.	0.	9.3261	1.00000	9.3261	0.99401
24	0.	0.	0.	0.	9.7146	1.00000	9.7146	1.00000
25	0.	0.	0.	0.	10.1032	1.00000	10.1032	1.00000
26	0.	0.	0.	0.	10.4918	1.00000	10.4918	1.00000
27	0.	0.	0.	0.	10.8804	1.00000	10.8804	1.00000
28	0.	0.	0.	0.	11.2690	1.00000	11.2690	1.00000
29	0.	0.	0.	0.	11.6576	1.00000	11.6576	1.00000
30	0.	0.	0.	0.	12.0462	1.00000	12.0462	1.00000

Asynch: priority-2 Normalized throughput (Rho_a) = 0.030000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
0	0.58892	0.69565	0.69565	0.65919	0.3886	0.11413	0.3886	0.02717
1	0.27405	0.23913	0.23913	0.24783	0.7772	0.16848	0.7772	0.14130
2	0.10204	0.04891	0.04891	0.06352	1.1658	0.23913	1.1658	0.19022
3	0.02332	0.01630	0.01630	0.01901	1.5543	0.30978	1.5543	0.25000
4	0.01166	0.	0.	0.01045	1.9429	0.36966	1.9429	0.30978
5	0.	0.	0.	0.	2.3315	0.36957	2.3315	0.35326
6	0.	0.	0.	0.	2.7201	0.44022	2.7201	0.40217
7	0.	0.	0.	0.	3.1087	0.52717	3.1087	0.46739
8	0.	0.	0.	0.	3.4973	0.58696	3.4973	0.54891
9	0.	0.	0.	0.	3.8859	0.63043	3.8859	0.58152
10	0.	0.	0.	0.	4.2744	0.67935	4.2744	0.63587
11	0.	0.	0.	0.	4.6630	0.72826	4.6630	0.70652
12	0.	0.	0.	0.	5.0516	0.78804	5.0516	0.73913
13	0.	0.	0.	0.	5.4402	0.81522	5.4402	0.78804
14	0.	0.	0.	0.	5.8288	0.82609	5.8288	0.80978
15	0.	0.	0.	0.	6.2174	0.83696	6.2174	0.83152
16	0.	0.	0.	0.	6.6060	0.84783	6.6060	0.84239
17	0.	0.	0.	0.	6.9945	0.86413	6.9945	0.85326
18	0.	0.	0.	0.	7.3831	0.86957	7.3831	0.86413
19	0.	0.	0.	0.	7.7717	0.86957	7.7717	0.86957
20	0.	0.	0.	0.	8.1603	0.88043	8.1603	0.87500
21	0.	0.	0.	0.	8.5489	0.89130	8.5489	0.88587

Asynch: priority-2 Normalized throughput (Rho_a) = 0.030000

89/10/04
10:27:42

fdi.3.out

3

22	0.	0.	0.	8.9375	0.90217	8.9375	0.89674
23	0.	0.	0.	9.3261	0.91304	9.3261	0.90761
24	0.	0.	0.	9.7146	0.91304	9.7146	0.91304
25	0.	0.	0.	10.1032	0.91848	10.1032	0.91304
26	0.	0.	0.	10.4918	0.91848	10.4918	0.91304
27	0.	0.	0.	10.8804	0.94022	10.8804	0.92391
28	0.	0.	0.	11.2690	0.95652	11.2690	0.94022
29	0.	0.	0.	11.6576	0.96196	11.6576	0.95109
30	0.	0.	0.	12.0462	1.00000	12.0462	1.00000

Async: priority-3 Normalized throughput (Rho_a)= 0.030000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
1	0.41983	0.54088	0.54375	0.43956	0.3886	0.05625	0.3886	0.01875
2	0.30904	0.24528	0.24375	0.28759	0.7772	0.13750	0.7772	0.07500
3	0.15743	0.11950	0.11875	0.15262	1.1658	0.17500	1.1658	0.15000
4	0.06997	0.05660	0.05625	0.08420	1.5543	0.23125	1.5543	0.19375
5	0.02915	0.02516	0.02500	0.02634	1.9429	0.23750	1.9429	0.22500
6	0.00583	0.01258	0.01250	0.00626	2.3315	0.26875	2.3315	0.25000
7	0.00875	0.	0.	0.00344	2.7201	0.28750	2.7201	0.26875
8	0.	0.	0.	0.	3.1087	0.31750	3.1087	0.29375
9	0.	0.	0.	0.	3.4973	0.38125	3.4973	0.35625
10	0.	0.	0.	0.	3.8859	0.40000	3.8859	0.38125
11	0.	0.	0.	0.	4.2744	0.41250	4.2744	0.39375
12	0.	0.	0.	0.	4.6630	0.43125	4.6630	0.41250
13	0.	0.	0.	0.	5.0516	0.45625	5.0516	0.43750
14	0.	0.	0.	0.	5.4402	0.47500	5.4402	0.45625
15	0.	0.	0.	0.	5.8288	0.49375	5.8288	0.47500
16	0.	0.	0.	0.	6.2174	0.53125	6.2174	0.49375
17	0.	0.	0.	0.	6.6060	0.53750	6.6060	0.53750
18	0.	0.	0.	0.	6.9945	0.54375	6.9945	0.53750
19	0.	0.	0.	0.	7.3831	0.55625	7.3831	0.54375
20	0.	0.	0.	0.	7.7717	0.58125	7.7717	0.56250
21	0.	0.	0.	0.	8.1603	0.58750	8.1603	0.56875
22	0.	0.	0.	0.	8.5489	0.60000	8.5489	0.59375
23	0.	0.	0.	0.	8.9375	0.60625	8.9375	0.60000
24	0.	0.	0.	0.	9.3261	0.62500	9.3261	0.61875
25	0.	0.	0.	0.	9.7146	0.65625	9.7146	0.63125
26	0.	0.	0.	0.	10.1032	0.68875	10.1032	0.65625
27	0.	0.	0.	0.	10.4918	0.68750	10.4918	0.67500
28	0.	0.	0.	0.	10.8804	0.70625	10.8804	0.70000
29	0.	0.	0.	0.	11.2690	0.70625	11.2690	0.70625
30	0.	0.	0.	0.	11.6576	0.71875	11.6576	0.70625
31	0.	0.	0.	0.	12.0462	1.00000	12.0462	1.00000

** Station 2

Synchronous Traffic: Normalized throughput (Rho_s)= 0.100000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
1	0.49271	0.61538	0.61538	0.62798	0.3886	0.12500	0.3886	0.01603
2	0.30904	0.27244	0.27244	0.26323	0.7772	0.22115	0.7772	0.08374
3	0.14869	0.08333	0.08333	0.08304	1.1658	0.32051	1.1658	0.19872
4	0.04082	0.02244	0.02244	0.01853	1.5543	0.40385	1.5543	0.27564
5	0.00292	0.00641	0.00641	0.00335	1.9429	0.52244	1.9429	0.38782
6	0.00583	0.	0.	0.00386	2.3315	0.58654	2.3315	0.49359
7	0.	0.	0.	0.	2.7201	0.67949	2.7201	0.57051
8	0.	0.	0.	0.	3.1087	0.73718	3.1087	0.64103
9	0.	0.	0.	0.	3.4973	0.78846	3.4973	0.71154
10	0.	0.	0.	0.	3.8859	0.83654	3.8859	0.76923
11	0.	0.	0.	0.	4.2744	0.87500	4.2744	0.81090
12	0.	0.	0.	0.	4.6630	0.91987	4.6630	0.85897
13	0.	0.	0.	0.	5.0516	0.95833	5.0516	0.91987
14	0.	0.	0.	0.	5.4402	0.98077	5.4402	0.95192

14	0.	0.	0.	5.8288	0.98397	5.8288	0.97436
15	0.	0.	0.	6.2174	0.98718	6.2174	0.98718
16	0.	0.	0.	6.6060	0.99359	6.6060	0.98718
17	0.	0.	0.	6.9945	1.00000	6.9945	0.99038
18	0.	0.	0.	7.3831	1.00000	7.3831	0.99679
19	0.	0.	0.	7.7717	1.00000	7.7717	1.00000
20	0.	0.	0.	8.1603	1.00000	8.1603	1.00000
21	0.	0.	0.	8.5489	1.00000	8.5489	1.00000
22	0.	0.	0.	8.9375	1.00000	8.9375	1.00000
23	0.	0.	0.	9.3261	1.00000	9.3261	1.00000
24	0.	0.	0.	9.7146	1.00000	9.7146	1.00000
25	0.	0.	0.	10.1032	1.00000	10.1032	1.00000
26	0.	0.	0.	10.4918	1.00000	10.4918	1.00000
27	0.	0.	0.	10.8804	1.00000	10.8804	1.00000
28	0.	0.	0.	11.2690	1.00000	11.2690	1.00000
29	0.	0.	0.	11.6576	1.00000	11.6576	1.00000
30	0.	0.	0.	12.0462	1.00000	12.0462	1.00000

Async: priority-1 Normalized throughput (Rho_a)= 0.030000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
1	0.68222	0.78481	0.78481	0.76447	0.3886	0.12025	0.3886	0.01899
2	0.24781	0.18987	0.18987	0.18920	0.7772	0.19620	0.7772	0.12025
3	0.06122	0.02532	0.02532	0.04166	1.1658	0.27215	1.1658	0.20886
4	0.00875	0.	0.	0.00467	1.5543	0.35443	1.5543	0.29747
5	0.	0.	0.	0.	1.9429	0.43038	1.9429	0.36709
6	0.	0.	0.	0.	2.3315	0.53797	2.3315	0.46203
7	0.	0.	0.	0.	2.7201	0.63291	2.7201	0.53165
8	0.	0.	0.	0.	3.1087	0.70253	3.1087	0.63291
9	0.	0.	0.	0.	3.4973	0.74684	3.4973	0.70253
10	0.	0.	0.	0.	3.8859	0.77215	3.8859	0.74684
11	0.	0.	0.	0.	4.2744	0.80810	4.2744	0.79114
12	0.	0.	0.	0.	4.6630	0.86709	4.6630	0.86076
13	0.	0.	0.	0.	5.0516	0.89241	5.0516	0.87342
14	0.	0.	0.	0.	5.4402	0.91139	5.4402	0.89873
15	0.	0.	0.	0.	5.8288	0.93671	5.8288	0.91139
16	0.	0.	0.	0.	6.2174	0.94937	6.2174	0.93671
17	0.	0.	0.	0.	6.6060	0.96835	6.6060	0.95570
18	0.	0.	0.	0.	6.9945	0.96835	6.9945	0.96835
19	0.	0.	0.	0.	7.3831	0.97468	7.3831	0.96835
20	0.	0.	0.	0.	7.7717	0.97468	7.7717	0.97468
21	0.	0.	0.	0.	8.1603	0.97468	8.1603	0.97468
22	0.	0.	0.	0.	8.5489	0.97468	8.5489	0.97468
23	0.	0.	0.	0.	8.9375	0.97468	8.9375	0.97468
24	0.	0.	0.	0.	9.3261	0.98101	9.3261	0.97468
25	0.	0.	0.	0.	9.7146	0.98734	9.7146	0.98101
26	0.	0.	0.	0.	10.1032	0.99367	10.1032	0.98734
27	0.	0.	0.	0.	10.4918	0.99367	10.4918	0.99367
28	0.	0.	0.	0.	10.8804	0.99367	10.8804	0.99367
29	0.	0.	0.	0.	11.2690	0.99367	11.2690	0.99367
30	0.	0.	0.	0.	11.6576	1.00000	11.6576	0.99367
31	0.	0.	0.	0.	12.0462	1.00000	12.0462	1.00000

Async: priority-2 Normalized throughput (Rho_a)= 0.030000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
1	0.64431	0.66443	0.66443	0.66459	0.3886	0.10738	0.3886	0.02013
2	0.25656	0.16779	0.16779	0.23468	0.7772	0.24832	0.7772	0.12081
3	0.05539	0.08725	0.08725	0.05153	1.1658	0.28188	1.1658	0.25503
4	0.02624	0.04027	0.04027	0.03040	1.5543	0.32886	1.5543	0.29530
5	0.00292	0.02013	0.02013	0.00480	1.9429	0.33557	1.9429	0.32886
6	0.00875	0.00671	0.00671	0.00723	2.3315	0.36913	2.3315	0.34228
7	0.00583	0.01342	0.01342	0.00359	2.7201	0.40940	2.7201	0.39597
8	0.	0.	0.	0.	3.1087	0.45638	3.1087	0.40940

89/10/04
10:27:42

fddi.3.out

4

Async: priority-3										Normalized throughput (Rho_a)= 0.030000									
j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)		j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)	
0	0.0	0.0	0.0	0.0	3.4973	0.52349	3.4973	0.47651		0	0.91837	1.00000	1.00000	0.94802	0.3886	0.06667	0.3886	0.03333	
9	0.0	0.0	0.0	0.0	3.8859	0.56376	3.8859	0.52349		1	0.008163	0.0	0.0	0.05198	0.7772	0.20000	0.7772	0.06667	
10	0.0	0.0	0.0	0.0	4.2744	0.57718	4.2744	0.57047		2	0.0	0.0	0.0	0.0	1.1658	0.30000	1.1658	0.16667	
11	0.0	0.0	0.0	0.0	4.6630	0.62416	4.6630	0.59732		3	0.0	0.0	0.0	0.0	1.5543	0.36667	1.5543	0.30000	
12	0.0	0.0	0.0	0.0	5.0516	0.65772	5.0516	0.62416		4	0.0	0.0	0.0	0.0	1.9429	0.46667	1.9429	0.36667	
13	0.0	0.0	0.0	0.0	5.4402	0.69799	5.4402	0.67114		5	0.0	0.0	0.0	0.0	2.3315	0.53333	2.3315	0.46667	
14	0.0	0.0	0.0	0.0	5.8288	0.71812	5.8288	0.69799		6	0.0	0.0	0.0	0.0	2.7201	0.60000	2.7201	0.50000	
15	0.0	0.0	0.0	0.0	6.2174	0.73826	6.2174	0.71812		7	0.0	0.0	0.0	0.0	3.1087	0.73333	3.1087	0.60000	
16	0.0	0.0	0.0	0.0	6.6060	0.77181	6.6060	0.74497		8	0.0	0.0	0.0	0.0	3.4973	0.83333	3.4973	0.73333	
17	0.0	0.0	0.0	0.0	6.9945	0.77852	6.9945	0.77852		9	0.0	0.0	0.0	0.0	3.8859	0.86667	3.8859	0.80000	
18	0.0	0.0	0.0	0.0	7.3831	0.78523	7.3831	0.78523		10	0.0	0.0	0.0	0.0	4.2744	0.90000	4.2744	0.86667	
19	0.0	0.0	0.0	0.0	7.7717	0.79866	7.7717	0.78523		11	0.0	0.0	0.0	0.0	4.6630	0.93333	4.6630	0.86667	
20	0.0	0.0	0.0	0.0	8.1603	0.81879	8.1603	0.81208		12	0.0	0.0	0.0	0.0	5.0516	0.96667	5.0516	0.90000	
21	0.0	0.0	0.0	0.0	8.5489	0.82550	8.5489	0.82550		13	0.0	0.0	0.0	0.0	5.4402	0.96667	5.4402	0.96667	
22	0.0	0.0	0.0	0.0	8.9375	0.83221	8.9375	0.82550		14	0.0	0.0	0.0	0.0	5.8288	0.96667	5.8288	0.96667	
23	0.0	0.0	0.0	0.0	9.3261	0.86577	9.3261	0.83893		15	0.0	0.0	0.0	0.0	6.2174	0.96667	6.2174	0.96667	
24	0.0	0.0	0.0	0.0	9.7146	0.87919	9.7146	0.87248		16	0.0	0.0	0.0	0.0	6.6060	0.96667	6.6060	0.96667	
25	0.0	0.0	0.0	0.0	10.1032	0.88591	10.1032	0.88591		17	0.0	0.0	0.0	0.0	6.9945	0.96667	6.9945	0.96667	
26	0.0	0.0	0.0	0.0	10.4918	0.88591	10.4918	0.88591		18	0.0	0.0	0.0	0.0	7.3831	1.00000	7.3831	1.00000	
27	0.0	0.0	0.0	0.0	10.8804	0.88591	10.8804	0.88591		19	0.0	0.0	0.0	0.0	7.7717	1.00000	7.7717	1.00000	
28	0.0	0.0	0.0	0.0	11.2690	0.88591	11.2690	0.88591		20	0.0	0.0	0.0	0.0	8.1603	1.00000	8.1603	1.00000	
29	0.0	0.0	0.0	0.0	11.6576	0.91275	11.6576	0.90604		21	0.0	0.0	0.0	0.0	8.5489	1.00000	8.5489	1.00000	
30	0.0	0.0	0.0	0.0	12.0462	1.00000	12.0462	1.00000		22	0.0	0.0	0.0	0.0	8.9375	1.00000	8.9375	1.00000	
										23	0.0	0.0	0.0	0.0	9.3261	1.00000	9.3261	1.00000	
										24	0.0	0.0	0.0	0.0	9.7146	1.00000	9.7146	1.00000	
										25	0.0	0.0	0.0	0.0	10.1032	1.00000	10.1032	1.00000	
										26	0.0	0.0	0.0	0.0	10.4918	1.00000	10.4918	1.00000	
										27	0.0	0.0	0.0	0.0	10.8804	1.00000	10.8804	1.00000	
										28	0.0	0.0	0.0	0.0	11.2690	1.00000	11.2690	1.00000	
										29	0.0	0.0	0.0	0.0	11.6576	1.00000	11.6576	1.00000	
										30	0.0	0.0	0.0	0.0	12.0462	1.00000	12.0462	1.00000	

Async: priority-3 Normalized throughput (Rho_a)= 0.030000

Async: priority-1										Normalized throughput (Rho_a)= 0.030000									
j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)		j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)	
0	0.55598	0.79503	0.79503	0.77401	0.3886	0.14286	0.3886	0.06211		0	0.91837	1.00000	1.00000	0.94802	0.3886	0.06667	0.3886	0.03333	
1	0.26822	0.17391	0.17391	0.18161	0.7772	0.2360	0.7772	0.16770		1	0.008163	0.0	0.0	0.05198	0.7772	0.20000	0.7772	0.06667	
2	0.06122	0.03106	0.03106	0.03906	1.1658	0.29814	1.1658	0.23602		2	0.0	0.0	0.0	0.0	1.1658	0.30000	1.1658	0.16667	
3	0.01458	0.0	0.0	0.00532	1.5543	0.37267	1.5543	0.32919		3	0.0	0.0	0.0	0.0	1.5543	0.36667	1.5543	0.30000	
4	0.0	0.0	0.0	0.0	1.9429	0.45963	1.9429	0.38509		4	0.0	0.0	0.0	0.0	1.9429	0.46667	1.9429	0.36667	
5	0.0	0.0	0.0	0.0	2.3315	0.54037	2.3315	0.48447		5	0.0	0.0	0.0	0.0	2.3315	0.53333	2.3315	0.46667	
6	0.0	0.0	0.0	0.0	2.7201	0.62733	2.7201	0.57143		6	0.0	0.0	0.0	0.0	2.7201	0.60000	2.7201	0.50000	
7	0.0	0.0	0.0	0.0	3.1087	0.70781	3.1087	0.62112		7	0.0	0.0	0.0	0.0	3.1087	0.73333	3.1087	0.60000	
8	0.0	0.0	0.0	0.0	3.4973	0.75776	3.4973	0.70186		8	0.0	0.0	0.0	0.0	3.4973	0.83333	3.4973	0.73333	
9	0.0	0.0	0.0	0.0	3.8859	0.79503	3.8859	0.75776		9	0.0	0.0	0.0	0.0	3.8859	0.86667	3.8859	0.80000	
10	0.0	0.0	0.0	0.0	4.2744	0.85093	4.2744	0.79503		10	0.0	0.0	0.0	0.0	4.2744	0.90000	4.2744	0.86667	
11	0.0	0.0	0.0	0.0	4.6630	0.86335	4.6630	0.84472		11	0.0	0.0	0.0	0.0	4.6630	0.93333	4.6630	0.86667	
12	0.0	0.0	0.0	0.0	5.0516	0.90683	5.0516	0.87578		12	0.0	0.0	0.0	0.0	5.0516	0.96667	5.0516	0.90000	
13	0.0	0.0	0.0	0.0	5.4402	0.94410	5.4402	0.91304		13	0.0	0.0	0.0	0.0	5.4402	0.96667	5.4402	0.96667	
14	0.0	0.0	0.0	0.0	5.8288	0.95031	5.8288	0.94410		14	0.0	0.0	0.0	0.0	5.8288	0.96667	5.8288	0.96667	
15	0.0	0.0	0.0	0.0	6.2174	0.96273	6.2174	0.95652		15	0.0	0.0	0.0	0.0	6.2174	0.96667	6.2174	0.96667	
16	0.0	0.0	0.0	0.0	6.6060	0.96894	6.6060	0.96273		16	0.0	0.0	0.0	0.0	6.6060	0.96667	6.6060	0.96667	
17	0.0	0.0	0.0	0.0	6.9945	0.98758	6.9945	0.97516		17	0.0	0.0	0.0	0.0	6.9945	0.96667	6.9945	0.96667	
18	0.0	0.0	0.0	0.0	7.3831	0.98758	7.3831	0.98758		18	0.0	0.0	0.0	0.0	7.3831	1.00000	7.3831	1.00000	
19	0.0	0.0	0.0	0.0	7.7717	0.98758	7.7717	0.98758		19	0.0	0.0	0.0	0.0	7.7717	1.00000	7.7717	1.00000	
20	0.0	0.0	0.0	0.0	8.1603	0.98758	8.1603	0.9758		20	0.0	0.0	0.0	0.0	8.1603	1.00000	8.1603	1.00000	
21	0.0	0.0	0.0	0.0	8.5489	0.99379	8.5489	0.9758		21	0.0	0.0	0.0	0.0	8.5489	1.00000	8.5489	1.00000	
22	0.0	0.0	0.0	0.0	8.9375	0.99379	8.9375	0.9758		22	0.0	0.0	0.0	0.0	8.9375	1.00000	8.9375	1.00000	
23	0.0	0.0	0.0	0.0	9.3261	0.99379	9.3261	0.9758		23	0.0	0.0	0.0	0.0	9.3261	1.00000	9.3261	1.00000	
24	0.0	0.0	0.0	0.0	9.7146	0.99379	9.7146	0.9758		24	0.0	0.0	0.0	0.0	9.7146	1.00000	9.7146	1.00000	
25	0.0	0.0	0.0	0.0	10.1032	0.99379	10.1032	0.9758		25	0.0	0.0	0.0	0.0	10.1032	1.00000	10.1032	1.00000	
26	0.0	0.0	0.0	0.0	10.4918	0.99379	10.4918	0.9758		26	0.0	0.0	0.0	0.0	10.4918	1.00000	10.4918	1.00000	
27	0.0	0.0	0.0	0.0	10.8804	0.99379	10.8804	0.9758		27	0.0	0.0	0.0	0.0	10.8804	1.00000	10.8804	1.00000	
28	0.0	0.0	0.0	0.0	11.2690	0.99379	11.2690	0.9758		28	0.0	0.0	0.0	0.0	11.2690	1.00000	11.2690	1.00000	
29	0.0	0.0	0.0	0.0	11.6576	0.99379	11.6576	0.9758		29	0.0	0.0	0.0	0.0	11.6576	1.00000	11.6576	1.00000	
30	0.0	0.0	0.0	0.0	12.0462	0.99379	12.0462	0.9758		30	0.0	0.0	0.0	0.0	12.0462	1.00000	12.0462	1.00000	

** Station 3

Synchronous Traffic: Normalized throughput (Rho_a)= 0.010000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

fddi.3.out

[illegible]

**** Station 4**

Synchronous Traffic: Normalized throughput (Rho s) = 0.100000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
0	0.46647	0.60802	0.60802	0.59822	0.3886	0.15432	0.3886	0.04012
1	0	0.27778	0.27778	0.27168	0.7772	0.24074	0.7772	0.11728
2	0.32945	0.09568	0.09568	0.10447	1.1768	0.3025	1.1658	0.22531
3	0.15452	0.01852	0.01852	0.02135	1.5543	0.41049	1.5543	0.31173
4	0.04082	0	0	0.00428	1.9429	0.4722	1.9429	0.39506
5	0.00875	0	0	0	2.3315	0.52778	2.3315	0.45679
6	0	0	0	0	2.7201	0.60802	2.7201	0.50309
7	0	0	0	0	3.1087	0.70062	3.1087	0.59259
8	0	0	0	0	3.4973	0.76235	3.4973	0.68519
9	0	0	0	0	3.8859	0.82716	3.8859	0.75000
10	0	0	0	0	4.2744	0.86420	4.2744	0.80864
11	0	0	0	0	4.6630	0.87654	4.6630	0.85185
12	0	0	0	0	5.0516	0.92284	5.0516	0.88272
13	0	0	0	0	5.4402	0.95988	5.4402	0.91358
14	0	0	0	0	5.8288	0.97840	5.8288	0.94136
15	0	0	0	0	6.2174	0.98148	6.2174	0.96914
16	0	0	0	0	6.6060	0.98765	6.6060	0.98148
17	0	0	0	0	6.9945	0.99383	6.9945	0.98457
18	0	0	0	0	7.3831	1.00000	7.3831	0.99074
19	0	0	0	0	7.7717	1.00000	7.7717	1.00000
20	0	0	0	0	8.1603	1.00000	8.1603	1.00000
21	0	0	0	0	8.5489	1.00000	8.5489	1.00000
22	0	0	0	0	8.9375	1.00000	8.9375	1.00000
23	0	0	0	0	9.3261	1.00000	9.3261	1.00000
24	0	0	0	0	9.7146	1.00000	9.7146	1.00000
25	0	0	0	0	10.1032	1.00000	10.1032	1.00000
26	0	0	0	0	10.4918	1.00000	10.4918	1.00000
27	0	0	0	0	10.8804	1.00000	10.8804	1.00000
28	0	0	0	0	11.2690	1.00000	11.2690	1.00000
29	0	0	0	0	11.6576	1.00000	11.6576	1.00000
30	0	0	0	0	12.0462	1.00000	12.0462	1.00000

Async: pr|or|ty-1 Normalized throughput (Rho a) = 0.030000

j	P(X=j)	P(X=a=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)
0	0.72012	0.73885	0.73885	0.77805	0.3866	0.17197	0.3866	0.07006
1	0.21574	0.22293	0.22293	0.18093	0.7772	0.31210	0.7772	0.21656
2	0.05248	0.03185	0.03185	0.03813	1.1658	0.35032	1.1658	0.29299
3	0.00875	0.00637	0.00637	0.00263	1.5543	0.42675	1.5543	0.36306
4	0.00292	0.0	0.0	0.00025	1.9429	0.47134	1.9429	0.42675
5	0.	0.	0.	0.	2.3315	0.53503	2.3315	0.47134
6	0.	0.	0.	0.	2.7201	0.58599	2.7201	0.56688
7	0.	0.	0.	0.	3.1087	0.65605	3.1087	0.59236
8	0.	0.	0.	0.	3.4973	0.72611	3.4973	0.64968
9	0.	0.	0.	0.	3.8859	0.77070	3.8859	0.72611
10	0.	0.	0.	0.	4.2744	0.80255	4.2744	0.77070
11	0.	0.	0.	0.	4.6630	0.83350	4.6630	0.80892
12	0.	0.	0.	0.	5.0516	0.86535	5.0516	0.85987
13	0.	0.	0.	0.	5.4402	0.90466	5.4402	0.87898
14	0.	0.	0.	0.	5.8288	0.93631	5.8288	0.92357

89/10/04
10:27:42

fddi.3.out

6

Async: priority-2 Normalized throughput (Rho_a)= 0.030000									
j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)	
1	0.03557	0.08966	0.09178	0.07069	0.3886	0.08219	0.3886	0.02055	
2	0.02682	0.02759	0.02603	0.02489	0.7772	0.15753	0.7772	0.08904	
3	0.06122	0.04828	0.04795	0.03932	1.1658	0.26712	1.1658	0.19863	
4	0.01458	0.01379	0.01370	0.01208	1.5543	0.34247	1.5543	0.28767	
5	0.00875	0.00690	0.00685	0.00158	1.9429	0.39041	1.9429	0.34932	
6	0.01166	0.0	0.0	0.01231	2.3315	0.43151	2.3315	0.40411	
7	0.0	0.0	0.0	0.01553	2.7201	0.49315	2.7201	0.44521	
8	0.0	0.0	0.0	0.0	3.1087	0.53425	3.1087	0.50685	
9	0.0	0.0	0.0	0.0	3.4973	0.57534	3.4973	0.56164	
10	0.0	0.0	0.0	0.0	3.8859	0.61644	3.8859	0.58904	
11	0.0	0.0	0.0	0.0	4.2744	0.65753	4.2744	0.62329	
12	0.0	0.0	0.0	0.0	4.6630	0.71233	4.6630	0.67808	
13	0.0	0.0	0.0	0.0	5.0516	0.73973	5.0516	0.70548	
14	0.0	0.0	0.0	0.0	5.4402	0.75342	5.4402	0.73342	
15	0.0	0.0	0.0	0.0	5.8288	0.76712	5.8288	0.75342	
16	0.0	0.0	0.0	0.0	6.2174	0.78082	6.2174	0.76712	
17	0.0	0.0	0.0	0.0	6.6060	0.78767	6.6060	0.78082	
18	0.0	0.0	0.0	0.0	6.9945	0.79452	6.9945	0.78767	
19	0.0	0.0	0.0	0.0	7.3831	0.81507	7.3831	0.80137	
20	0.0	0.0	0.0	0.0	7.7717	0.82877	7.7717	0.81507	
21	0.0	0.0	0.0	0.0	8.1603	0.83562	8.1603	0.83562	
22	0.0	0.0	0.0	0.0	8.5489	0.84932	8.5489	0.83562	
23	0.0	0.0	0.0	0.0	8.9375	0.85616	8.9375	0.84932	
24	0.0	0.0	0.0	0.0	9.3261	0.85616	9.3261	0.85616	
25	0.0	0.0	0.0	0.0	9.7146	0.86301	9.7146	0.85616	
26	0.0	0.0	0.0	0.0	10.1032	0.88356	10.1032	0.86301	
27	0.0	0.0	0.0	0.0	10.4918	0.89726	10.4918	0.88356	
28	0.0	0.0	0.0	0.0	10.8804	0.91096	10.8804	0.89726	
29	0.0	0.0	0.0	0.0	11.2690	0.91781	11.2690	0.91781	
30	0.0	0.0	0.0	0.0	11.6576	0.93151	11.6576	0.91781	
31	0.0	0.0	0.0	0.0	12.0462	1.00000	12.0462	1.00000	

Async: priority-3 Normalized throughput (Rho_a)= 0.030000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)	
1	0.037318	0.029878	0.029878	0.032055	0.3886	0.03659	0.3886	0.01220	
2	0.18659	0.13665	0.14024	0.17506	0.7772	0.06098	0.7772	0.04268	
3	0.10204	0.10559	0.10976	0.10221	1.1658	0.07317	1.1658	0.06098	
4	0.04956	0.08632	0.08537	0.06569	1.5543	0.09146	1.5543	0.06707	
5	0.06414	0.05590	0.05488	0.07478	1.9429	0.12805	1.9429	0.09756	
6	0.04956	0.04348	0.04268	0.06719	2.3315	0.13415	2.3315	0.13415	
7	0.02915	0.04969	0.04878	0.03019	2.7201	0.14634	2.7201	0.14024	
8	0.02041	0.04969	0.04878	0.02666	3.1087	0.15854	3.1087	0.15244	
9	0.0	0.0	0.0	0.0	3.4973	0.16463	3.4973	0.16463	

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)	
1	0.03499	0.03106	0.03049	0.03604	0.3886	0.18293	0.3886	0.17073	
2	0.02041	0.01863	0.01829	0.02194	0.7772	0.18902	0.7772	0.18293	
3	0.00583	0.01242	0.01220	0.00649	1.1658	0.21341	1.1658	0.18902	
4	0.00292	0.01242	0.01220	0.00480	1.5543	0.22561	1.5543	0.21341	
5	0.00583	0.01242	0.01220	0.00346	1.9429	0.23171	1.9429	0.22561	
6	0.00292	0.01242	0.01220	0.00310	2.3315	0.25000	2.3315	0.23780	
7	0.00292	0.00621	0.00610	0.00257	2.7201	0.25000	2.7201	0.25000	
8	0.0	0.0	0.0	0.00043	3.1087	0.25000	3.1087	0.25000	
9	0.0	0.0	0.0	0.0	3.4973	0.26220	3.4973	0.25610	
10	0.0	0.0	0.0	0.0	3.8859	0.26220	3.8859	0.25610	
11	0.0	0.0	0.0	0.0	4.2744	0.27439	4.2744	0.26829	
12	0.0	0.0	0.0	0.0	4.6630	0.29268	4.6630	0.28049	
13	0.0	0.0	0.0	0.0	5.0516	0.30488	5.0516	0.29268	
14	0.0	0.0	0.0	0.0	5.4402	0.31707	5.4402	0.31098	
15	0.0	0.0	0.0	0.0	5.8288	0.32317	5.8288	0.31707	
16	0.0	0.0	0.0	0.0	6.2174	0.33537	6.2174	0.32317	
17	0.0	0.0	0.0	0.0	6.6060	0.33537	6.6060	0.33537	
18	0.0	0.0	0.0	0.0	6.9945	0.35366	6.9945	0.34146	
19	0.0	0.0	0.0	0.0	7.3831	0.35366	7.3831	0.35366	
20	0.0	0.0	0.0	0.0	7.7717	0.36585	7.7717	0.35976	
21	0.0	0.0	0.0	0.0	8.1603	0.37195	8.1603	0.37195	
22	0.0	0.0	0.0	0.0	8.5489	0.37805	8.5489	0.37805	
23	0.0	0.0	0.0	0.0	8.9375	0.37805	8.9375	0.37805	
24	0.0	0.0	0.0	0.0	9.3261	0.37805	9.3261	0.37805	
25	0.0	0.0	0.0	0.0	9.7146	1.00000	9.7146	1.00000	
26	0.0	0.0	0.0	0.0	10.1032	1.00000	10.1032	1.00000	
27	0.0	0.0	0.0	0.0	10.4918	1.00000	10.4918	1.00000	
28	0.0	0.0	0.0	0.0	10.8804	1.00000	10.8804	1.00000	
29	0.0	0.0	0.0	0.0	11.2690	1.00000	11.2690	1.00000	
30	0.0	0.0	0.0	0.0	11.6576	1.00000	11.6576	1.00000	
31	0.0	0.0	0.0	0.0	12.0462	1.00000	12.0462	1.00000	

** Station 5

Synchronous Traffic: Normalized throughput (Rho_a)= 0.010000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)	
1	0.90671	0.97222	0.97222	0.93364	0.3886	0.08333	0.3886	0.02778	
2	0.09329	0.02778	0.02778	0.06609	0.7772	0.13889	0.7772	0.03556	
3	0.0	0.0	0.0	0.00027	1.1658	0.27778	1.1658	0.11111	
4	0.0	0.0	0.0	0.0	1.5543	0.44444	1.5543	0.22222	
5	0.0	0.0	0.0	0.0	1.9429	0.55556	1.9429	0.33333	
6	0.0	0.0	0.0	0.0	2.3315	0.61111	2.3315	0.52778	
7	0.0	0.0	0.0	0.0	2.7201	0.66667	2.7201	0.58333	
8	0.0	0.0	0.0	0.0	3.1087	0.77778	3.1087	0.66667	
9	0.0	0.0	0.0	0.0	3.4973	0.83333	3.4973	0.72222	
10	0.0	0.0	0.0	0.0	3.8859	0.86111	3.8859	0.83333	
11	0.0	0.0	0.0	0.0	4.2744	0.86111	4.2744	0.83333	
12	0.0	0.0	0.0	0.0	4.6630	0.86111	4.6630	0.86111	
13	0.0	0.0	0.0	0.0	5.0516	0.88889	5.0516	0.86111	
14	0.0	0.0	0.0	0.0	5.4402	0.91667	5.4402	0.86111	
15	0.0	0.0	0.0	0.0	5.8288	0.97222	5.8288	0.88889	
16	0.0	0.0	0.0	0.0	6.2174	0.97222	6.2174	0.94444	
17	0.0	0.0	0.0	0.0	6.6060	0.97222	6.6060	0.97222	
18	0.0	0.0	0.0	0.0	6.9945	0.97222	6.9945	0.97222	
19	0.0	0.0	0.0	0.0	7.3831	1.00000	7.3831	0.97222	
20	0.0	0.0	0.0	0.0	7.7717	1.00000	7.7717	0.97222	
21	0.0	0.0	0.0	0.0	8.1603	1.00000	8.1603	1.00000	
22	0.0	0.0	0.0	0.0	8.5489	1.00000	8.5489	1.00000	
23	0.0	0.0	0.0	0.0	8.9375	1.00000	8.9375	1.00000	
24	0.0	0.0	0.0	0.0	9.3261	1.00000	9.3261	1.00000	
25	0.0	0.0	0.0	0.0	9.7146	1.00000	9.7146	1.00000	
26	0.0	0.0	0.0	0.0	10.1032	1.00000	10.1032	1.00000	
27	0.0	0.0	0.0	0.0	10.4918	1.00000	10.4918	1.00000	
28	0.0	0.0	0.0	0.0	10.8804	1.00000	10.8804	1.00000	
29	0.0	0.0	0.0	0.0	11.2690	1.00000	11.2690	1.00000	
30	0.0	0.0	0.0	0.0	11.6576	1.00000	11.6576	1.00000	
31	0.0	0.0	0.0	0.0	12.0462	1.00000	12.0462	1.00000	

Async: priority-1 Normalized throughput (Rho_a)= 0.030000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)	
1	0.69096	0.78231	0.78231	0.78496	0.3886	0.12925	0.3886	0.31803	

Async: priority-3										Normalized throughput (Rho_a)= 0.030000										12.0462 1.00000 12.0462 1.00000									
j	P(X=j)	P(Xa=j)	P(XG=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)		j	P(X=j)	P(Xa=j)	P(XG=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)		j	P(X=j)	P(Xa=j)	P(XG=j)	P(Xt=j)	w	P(W<w)	d	P(D<d)	
0	0.39942	0.46196	0.46196	0.41246	0.3806	0.02717	0.3086	0.01087		10	0.090	0.0	0.0	0.0	4.2744	0.36413	4.2744	0.34239		20	0.0	0.0	0.0	0.0	5.4402	0.45109	5.4402	0.42391	
1	0.27988	0.26087	0.26087	0.28743	0.7772	0.07609	0.7772	0.04348		11	0.0	0.0	0.0	0.0	5.0516	0.42391	5.0516	0.39130		21	0.0	0.0	0.0	0.0	6.2174	0.47826	6.2174	0.47826	
2	0.15452	0.13043	0.13043	0.14085	1.1658	0.10326	1.1658	0.09239		12	0.0	0.0	0.0	0.0	5.8288	0.47826	5.8288	0.46196		22	0.0	0.0	0.0	0.0	6.0606	0.49457	6.0606	0.47826	
3	0.09038	0.06522	0.06522	0.08494	1.5543	0.16848	1.5543	0.11413		13	0.0	0.0	0.0	0.0	6.2174	0.47826	6.2174	0.47826		23	0.0	0.0	0.0	0.0	6.9945	0.50543	6.9945	0.49457	
4	0.02624	0.03261	0.03261	0.03064	1.9429	0.19565	1.9429	0.15217		14	0.0	0.0	0.0	0.0	7.3831	0.52174	7.3831	0.51087		24	0.0	0.0	0.0	0.0	7.7717	0.54348	7.7717	0.52177	
5	0.02332	0.03261	0.03261	0.02014	2.3315	0.21196	2.3315	0.20652		15	0.0	0.0	0.0	0.0	8.1603	0.56522	8.1603	0.54891		25	0.0	0.0	0.0	0.0	8.5489	0.58152	8.5489	0.57065	
6	0.02332	0.01087	0.01087	0.02207	2.7201	0.48261	2.7201	0.23670		16	0.0	0.0	0.0	0.0	8.9375	0.63043	8.9375	0.59783		26	0.0	0.0	0.0	0.0	9.3261	0.64674	9.3261	0.62500	
7	0.0	0.00543	0.00543	0.00058	3.1087	0.29891	3.1087	0.28087		17	0.0	0.0	0.0	0.0	9.7146	0.65217	9.7146	0.64130		27	0.0	0.0	0.0	0.0	10.1032	0.66304	10.1032	0.65761	
8	0.00292	0.0	0.0	0.00090	3.4973	0.31522	3.4973	0.30435		18	0.0	0.0	0.0	0.0	10.4918	0.67391	10.4918	0.66304		28	0.0	0.0	0.0	0.0	10.8804	0.69565	10.8804	0.67935	
9	0.0	0.0	0.0	0.0	3.8859	0.33152	3.8859	0.32065		19	0.0	0.0	0.0	0.0	11.2690	0.69565	11.2690	0.69565		29	0.0	0.0	0.0	0.0	11.6576	0.70109	11.6576	0.69565	
10	0.0	0.0	0.0	0.0	4.2744	0.36413	4.2744	0.34239		20	0.0	0.0	0.0	0.0	12.0462	1.00000	12.0462	1.00000		30	0.0	0.0	0.0	0.0	12.0462	1.00000	12.0462	1.00000	

Synchronous Traffic: Normalized throughput (ρ_{ss}) = 0.100000

j	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xt=j)	w	P(w<w)	d	P(D<w)
0	0.50437	0.60790	0.60790	0.61753	0.3886	0.15502	0.3886	0.03951
1	0.29155	0.26140	0.26140	0.26115	0.7772	0.24620	0.7772	0.11854
2	0.14286	0.09119	0.09119	0.08860	1.1658	0.31003	1.1658	0.20365
3	0.04665	0.02128	0.02128	0.02572	1.5543	0.42553	1.5543	0.27356
4	0.00292	0.01520	0.01520	0.00365	1.9429	0.50152	1.9429	0.38298
5	0.00875	0.00304	0.00304	0.00324	2.3315	0.60790	2.3315	0.49544
6	0.00292	0.	0.	0.00011	2.7201	0.68693	2.7201	0.58359
7	0.	0.	0.	0.	3.1087	0.73860	3.1087	0.65653
8	0.	0.	0.	0.	3.4973	0.78723	3.4973	0.71429
9	0.	0.	0.	0.	3.8859	0.83587	3.8859	0.77204
10	0.	0.	0.	0.	4.2744	0.89362	4.2744	0.82979
11	0.	0.	0.	0.	4.6630	0.92705	4.6630	0.88146
12	0.	0.	0.	0.	5.0516	0.94529	5.0516	0.92705
13	0.	0.	0.	0.	5.4402	0.95745	5.4402	0.93313
14	0.	0.	0.	0.	5.8288	0.98176	5.8288	0.96049
15	0.	0.	0.	0.	6.2174	0.98176	6.2174	0.96960
16	0.	0.	0.	0.	6.6060	0.98784	6.6060	0.97569
17	0.	0.	0.	0.	6.9945	0.99392	6.9945	0.98480
18	0.	0.	0.	0.	7.3831	0.99392	7.3831	0.99392
19	0.	0.	0.	0.	7.7717	0.99392	7.7717	0.99392
20	0.	0.	0.	0.	8.1603	1.00000	8.1603	0.99696
21	0.	0.	0.	0.	8.5489	1.00000	8.5489	1.00000

Normalized throughput (Rho_a) = 0.030000

j	P(X=j)	P(X=1)	P(X=2)	P(X=3)	P(X=4)	P(X=5)	W	P(W<w)	d	P(D<=d)
0	0.63048	0.68293	0.68293	0.72749	0.72749	0.72749	0.3886	0.11585	0.3886	0.01829
1	0.23224	0.21341	0.21341	0.17763	0.17763	0.17763	0.7772	0.20122	0.7772	0.12195
2	0.08455	0.07317	0.07317	0.06420	0.06420	0.06420	1.1658	0.31707	1.1658	0.25000
3	0.03207	0.03049	0.03049	0.02599	0.02599	0.02599	1.5543	0.36585	1.5543	0.31707
4	0.01166	0.	0.	0.00469	0.00469	0.00469	1.9429	0.43902	1.9429	0.38415
5	0.	0.	0.	0.	0.	0.	2.3315	0.47561	2.3315	0.43902
6	0.	0.	0.	0.	0.	0.	2.7201	0.53659	2.7201	0.48171
7	0.	0.	0.	0.	0.	0.	3.1087	0.56707	3.1087	0.53049
8	0.	0.	0.	0.	0.	0.	3.4973	0.62805	3.4973	0.61585
9	0.	0.	0.	0.	0.	0.	3.8859	0.66463	3.8859	0.64024
10	0.	0.	0.	0.	0.	0.	4.2744	0.68902	4.2744	0.67073
11	0.	0.	0.	0.	0.	0.	4.6630	0.73780	4.6630	0.69512
12	0.	0.	0.	0.	0.	0.	5.0516	0.76220	5.0516	0.74390
13	0.	0.	0.	0.	0.	0.	5.4402	0.77439	5.4402	0.76829
14	0.	0.	0.	0.	0.	0.	5.8288	0.79878	5.8288	0.77439
15	0.	0.	0.	0.	0.	0.	6.2174	0.82317	6.2174	0.79878
16	0.	0.	0.	0.	0.	0.	6.6060	0.84146	6.6060	0.82927
17	0.	0.	0.	0.	0.	0.	6.9945	0.84756	6.9945	0.83537
18	0.	0.	0.	0.	0.	0.	7.3831	0.84756	7.3831	0.84756
19	0.	0.	0.	0.	0.	0.	7.7717	0.86585	7.7717	0.85976
20	0.	0.	0.	0.	0.	0.	8.1603	0.87805	8.1603	0.87805
21	0.	0.	0.	0.	0.	0.	8.5489	0.89024	8.5489	0.88415
22	0.	0.	0.	0.	0.	0.	8.9375	0.90244	8.9375	0.89024
23	0.	0.	0.	0.	0.	0.	9.3261	0.90854	9.3261	0.90244
24	0.	0.	0.	0.	0.	0.	9.7146	0.91463	9.7146	0.90854
25	0.	0.	0.	0.	0.	0.	10.1032	0.92683	10.1032	0.91463
26	0.	0.	0.	0.	0.	0.	10.4918	0.93902	10.4918	0.92073
27	0.	0.	0.	0.	0.	0.	10.8804	0.95122	10.8804	0.93293
28	0.	0.	0.	0.	0.	0.	11.2690	0.96341	11.2690	0.95122
29	0.	0.	0.	0.	0.	0.	11.6576	0.96341	11.6576	0.95732

89/10/04
10:27:42

fddi.3.out

8

22 0. 0. 8.9375 1.00000 8.9375 1.00000
23 0. 0. 9.3261 1.00000 9.3261 1.00000
24 0. 0. 9.7146 1.00000 9.7146 1.00000
25 0. 0. 10.1032 1.00000 10.1032 1.00000
26 0. 0. 10.4918 1.00000 10.4918 1.00000
27 0. 0. 10.8804 1.00000 10.8804 1.00000
28 0. 0. 11.2690 1.00000 11.2690 1.00000
29 0. 0. 11.6576 1.00000 11.6576 1.00000
30 0. 0. 12.0462 1.00000 12.0462 1.00000

Async: priority-1 Normalized throughput (Rho_a)= 0.030000

J	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xc=j)	W	P(W<w)	d	P(D<d)
0	0.63869	0.80503	0.80503	0.73313	0.3886	0.10063	0.3886	0.03774
1	0.27988	0.15094	0.15094	0.23040	0.7772	0.17610	0.7772	0.11321
2	0.05248	0.03774	0.03774	0.03158	1.1658	0.24528	1.1658	0.16981
3	0.00875	0.00629	0.00629	0.00481	1.5543	0.33333	1.5543	0.25157
4	0.	0.	0.	0.00009	1.9429	0.39623	1.9429	0.32704
5	0.	0.	0.	0.	2.3315	0.49686	2.3315	0.43396
6	0.	0.	0.	0.	2.7201	0.60377	2.7201	0.52201
7	0.	0.	0.	0.	3.1087	0.67925	3.1087	0.63522
8	0.	0.	0.	0.	3.4973	0.73585	3.4973	0.67925
9	0.	0.	0.	0.	3.8859	0.79245	3.8859	0.74843
10	0.	0.	0.	0.	4.2744	0.82390	4.2744	0.79874
11	0.	0.	0.	0.	4.6630	0.86792	4.6630	0.83019
12	0.	0.	0.	0.	5.0516	0.88679	5.0516	0.87421
13	0.	0.	0.	0.	5.4402	0.91824	5.4402	0.90566
14	0.	0.	0.	0.	5.8288	0.91824	5.8288	0.91195
15	0.	0.	0.	0.	6.2174	0.93711	6.2174	0.93082
16	0.	0.	0.	0.	6.6060	0.93711	6.6060	0.93711
17	0.	0.	0.	0.	6.9945	0.94969	6.9945	0.94969
18	0.	0.	0.	0.	7.3831	0.94969	7.3831	0.94969
19	0.	0.	0.	0.	7.7717	0.94969	7.7717	0.94969
20	0.	0.	0.	0.	8.1603	0.96226	8.1603	0.94969
21	0.	0.	0.	0.	8.5489	0.96855	8.5489	0.95597
22	0.	0.	0.	0.	8.9375	0.96855	8.9375	0.96855
23	0.	0.	0.	0.	9.3261	0.97484	9.3261	0.96855
24	0.	0.	0.	0.	9.7146	0.97484	9.7146	0.97484
25	0.	0.	0.	0.	10.1032	0.97484	10.1032	0.97484
26	0.	0.	0.	0.	10.4918	0.98113	10.4918	0.97484
27	0.	0.	0.	0.	10.8804	0.98113	10.8804	0.97484
28	0.	0.	0.	0.	11.2690	0.98113	11.2690	0.98113
29	0.	0.	0.	0.	11.6576	0.98113	11.6576	0.98113
30	0.	0.	0.	0.	12.0462	1.00000	12.0462	1.00000

Async: priority-2 Normalized throughput (Rho_a)= 0.030000

J	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xc=j)	W	P(W<w)	d	P(D<d)
0	0.62391	0.72619	0.72619	0.68207	0.3886	0.16071	0.3886	0.04762
1	0.29155	0.22619	0.22619	0.24307	0.7772	0.22619	0.7772	0.18452
2	0.06706	0.02976	0.02976	0.03718	1.1658	0.29167	1.1658	0.21429
3	0.00875	0.01786	0.01786	0.01063	1.5543	0.34524	1.5543	0.30357
4	0.00875	0.	0.	0.00704	1.9429	0.44048	1.9429	0.36905
5	0.	0.	0.	0.	2.3315	0.48214	2.3315	0.45238
6	0.	0.	0.	0.	2.7201	0.52381	2.7201	0.50000
7	0.	0.	0.	0.	3.1087	0.57143	3.1087	0.52976
8	0.	0.	0.	0.	3.4973	0.60714	3.4973	0.57738
9	0.	0.	0.	0.	3.8859	0.64881	3.8859	0.61905
10	0.	0.	0.	0.	4.2744	0.68452	4.2744	0.66071
11	0.	0.	0.	0.	4.6630	0.73810	4.6630	0.70833
12	0.	0.	0.	0.	5.0516	0.79167	5.0516	0.74405
13	0.	0.	0.	0.	5.4402	0.82738	5.4402	0.78571
14	0.	0.	0.	0.	5.8288	0.84524	5.8288	0.83333
15	0.	0.	0.	0.	6.2174	0.85714	6.2174	0.84524

Async: priority-3 Normalized throughput (Rho_a)= 0.030000

J	P(X=j)	P(Xa=j)	P(Xd=j)	P(Xc=j)	W	P(W<w)	d	P(D<d)
0	0.36152	0.25949	0.25949	0.29469	0.3886	0.01266	0.3886	0.
1	0.12536	0.12025	0.12025	0.11582	0.7772	0.03797	0.7772	0.01899
2	0.09913	0.09494	0.09494	0.10700	1.1658	0.05063	1.1658	0.04430
3	0.06706	0.06329	0.06329	0.06201	1.5543	0.08228	1.5543	0.06329
4	0.04665	0.05696	0.05696	0.06844	1.9429	0.10127	1.9429	0.08861
5	0.05248	0.05696	0.05696	0.06032	2.3315	0.13291	2.3315	0.12025
6	0.02624	0.05063	0.05063	0.02891	2.7201	0.15190	2.7201	0.13924
7	0.03790	0.05696	0.05696	0.04830	3.1087	0.17089	3.1087	0.15823
8	0.01458	0.05063	0.05063	0.01848	3.4973	0.17089	3.4973	0.17089
9	0.05539	0.04430	0.04430	0.07046	3.8859	0.18987	3.8859	0.18354
10	0.03790	0.03797	0.03797	0.03701	4.2744	0.19620	4.2744	0.18987
11	0.02041	0.03797	0.03797	0.02378	4.6630	0.19620	4.6630	0.19620
12	0.01166	0.01899	0.01899	0.01509	5.0516	0.20253	5.0516	0.19620
13	0.01749	0.01266	0.01266	0.01763	5.4402	0.20886	5.4402	0.20253
14	0.00583	0.01899	0.01899	0.01009	5.8288	0.21519	5.8288	0.20886
15	0.00383	0.01266	0.01266	0.00713	6.2174	0.22785	6.2174	0.22152
16	0.00583	0.00633	0.00633	0.00302	6.6060	0.24051	6.6060	0.22785
17	0.00875	0.	0.	0.01184	6.9945	0.24051	6.9945	0.24051
18	0.	0.	0.	0.	7.3831	0.24684	7.3831	0.24051
19	0.	0.	0.	0.	7.7717	0.24684	7.7717	0.24684
20	0.	0.	0.	0.	8.1603	0.24684	8.1603	0.24684
21	0.	0.	0.	0.	8.5489	0.24684	8.5489	0.24684
22	0.	0.	0.	0.	8.9375	0.26582	8.9375	0.25949
23	0.	0.	0.	0.	9.3261	0.26582	9.3261	0.26582
24	0.	0.	0.	0.	9.7146	0.27215	9.7146	0.26582
25	0.	0.	0.	0.	10.1032	0.27215	10.1032	0.27215
26	0.	0.	0.	0.	10.4918	0.27848	10.4918	0.27215
27	0.	0.	0.	0.	10.8804	0.28481	10.8804	0.27848
28	0.	0.	0.	0.	11.2690	0.29114	11.2690	0.28481
29	0.	0.	0.	0.	11.6576	0.29114	11.6576	0.29114
30	0.	0.	0.	0.	12.0462	1.00000	12.0462	1.00000

fddi.f

```
if (i2.lt.m2) goto 2
if (k1.lt.N1) goto 4
if (k2.lt.N2) goto 5
return
end
```

```
subroutine blocks(nq,atail,t2)
implicit real*8 (A-H,O-Z)
logical ofs(20),zf
real*8 tofs(20),cs(20),ys(20),wofs(20)
integer os(20),ss(20),seeds(20)
common /bs/ofs,ss,tofs,cs,os,seeds,ys,wofs /bc/sp,B,zf
```

```
1 temp = atail
s1 = dmod(sp*ss(nq),B)
p1 = -dlog(s1/B)*cs(nq)
temp = temp+max(0.000001,p1)
ss(nq) = s1
if (temp.lt.t2) then
  if (zf) os(nq) = os(nq)+1
  goto 1
end if
tofs(nq) = temp
s2 = dmod(sp*seeds(nq),B)
wofs(nq) = max(0.000001,s2/B*ys(nq))
seeds(nq) = s2
ofs(nq) = .true.
return
end
```

```
subroutine blocka(nq,j,atail,t2)
implicit real*8 (A-H,O-Z)
logical ofa(20,3),zf
real*8 tofa(20,3),ca(20,3),ya(20,3),wofa(20,3)
integer oa(20,3),sa(20,3),seeda(20,3)
common /ba/ofa,sa,tofa,ca,oa,seeda,ya,wofa /bc/sp,B,zf
```

```
1 temp = atail
s1 = dmod(sp*sa(nq,j),B)
p1 = -dlog(s1/B)*ca(nq,j)
temp = temp+max(0.000001,p1)
sa(nq,j) = s1
if (temp.lt.t2) then
  if (zf) oa(nq,j) = oa(nq,j)+1
  goto 1
end if
tofa(nq,j) = temp
s2 = dmod(sp*seeda(nq,j),B)
wofa(nq,j) = max(0.000001,s2/B*ya(nq,j))
seeda(nq,j) = s2
ofa(nq,j) = .true.
return
end
```

```

C>itdma2
Enter the file_name for data
(no more than 12 characters)
(in PC, QUOTE the 'file_name')
'iout1'
What would you like to do ?
  1: Simulation only
  2: Analysis only
  3: Simulation and analysis
Please enter a number:
3
Enter the start time (when statistic starts)
100
Enter the stop time (total simulation time)
1000

Enter the frame duration (m) ( $m \leq 50$ ) and the maximum
number of slots for CS support (n)
( $n \leq m$ ) m and n should be integers
10 5
Enter the session (circuit) arrival rate
0.01
Enter the session transmission rate
0.04

Enter the packet batch arrival rate [mess./slot] p1
0.1
please wait
average number of CS slots used per frame (V): 2.334677242395972
Enter the batch size distribution index
  1: deterministic
  2: geometric
  3: uniform

2
Enter the mean batch size, b ( $m * b * p1 < m - V$ ):
5
To find the probabilities  $P(X > x)$  and  $P(D > d)$ , please
enter x and d (x,  $d \leq 20$ , both should be integers)
10 10
please wait

```

Appendix J.

The FDDI Program: Source Code

Simulation of Timed Token Protocol MAC (FDDI Type)

Version 1 Sep. 20, 1989
Professor Izhak Rubin
Department of Electrical Engineering
University of California
Los Angeles, CA 90024

Description of Input Parameters

```

TTRT      : Target Token Rotation Time (msec)
N          : number of stations
Np        : number of priorities in each station
as(i)     : arrival rate for synchronous traffic at station i
aa(i,j)   : arrival rate for asynchronous traffic at station i, priority j
plens     : mean packet length for synchronous traffic
plena     : mean packet length for asynchronous traffic
Tpri(i,j) : T-Pri, priority threshold for asynchronous traffic
BWT       : synchronous bandwidth time assignment
r(i)      : walk time from station i to station i+1
w,d,x     : specified values for P(W>w), P(D>d), P(X>x)

```

program FDDI

```

implicit real*8 (A-H,O-Z)
character fi*12,fo*12,ch*1
logical gf,zf,sf,af,ofs(20),ofa(20,3)
real azs(20),dzs(20),aza(20,3),dza(20,3),tzs(20),tza(20,3)
* ,tds(20),tws(20),tda(20,3),twa(20,3)
real*8 Aas(20,0:200),works(20,0:200),as(20),plens(20),
* Tpri(20,3),aa(20,3),plena(20,3),Us(20),Xav(20,3),
* Aaa(20,3,0:100),worka(20,3,0:100),Ua(20,3),r(20),res(20),
* TRT,t0(20),BWT(20),v(20),Wa(20,3),Ws(20),Ds(20),Da(20,3)
* ,cs(20),ca(20,3),ys(20),ya(20,3),Xsm(20),Xsv(20),Xam(20,3)
* ,sigws(20),sigds(20),sigwa(20,3),sigda(20,3),tya(20,3)
*,tss(20,0:50),tsa(20,3,0:50),tjumps(20),tjumpa(20,3),tys(20)
*,dwell(20),dwellv(20),ctime(20),ctimev(20),dpar(20),dparv(20)
*,tqs(20),tqa(20,3),tqsv(20),tqav(20,3),tua(3)
*,asc(2),plensc(2),BWTc(2),Tpric(2,8),aac(2,8),plenac(2,8)
*,tofs(20),tofa(20,3),wofs(20),wofa(20,3)
integer tails(20),taila(20,3),ss(20),sa(20,3),seeds(20),
*seeda(20,3),heads(20),heada(20,3),txs(200,0:50),txa(20,3,0:50),
*zcs(20),zca(20,3),os(20),oa(20,3),dxa(20,3,0:50),NN(2)
*,Xs(20),Xa(20,3),axs(20,0:50),dxs(20,0:50),axa(20,3,0:50)
*,pds(20,0:50),pws(20,0:50),pda(20,3,0:50),pwa(20,3,0:50),a(20)
*,PrXs(20),PrXa(20,3),PrWs(20),PrWa(20,3),PrDs(20),PrDa(20,3),X0
common Aas,Aaa,works,worka,Us,Ua,heads,heada,tails,taila,N,Np,
* Xs,Xa,axs,dxs,axa,mind,tss,tsa,tjumps,tjumpa
common /syn/v,res,Ws,Ds,sigws,sigds,scale,pws,pds,zcs,PrWs,PrDs,
* W0,D0 /bs/ofs,ss,tofs,cs,os,seeds,ys,wofs /bc/sp,B,zf
*/ba/ofa,sa,tofa,ca,oa,seeda,ya,wofa

```

mind = 30

```
base for random numbers
sp = db1e(7**5)
yz = 1.
do 101 i = 1,31
```

```
101  yz = yz*2
      B = yz-1

***  Input Section  ***
      write(*,*) '** Simulation of a Timed Token Rotation Protocol'
      *      , ' (FDDI-I Type)'
      write(*,*) 'program: FDDI'
      write(*,*) 'Professor Izhak Rubin, UCLA'
      write(*,*)
      write(*,*) 'Enter Select Input Mode'
      write(*,*) '1. from KEYBOARD'
      write(*,*) '2. from DATA FILE'
      read(*,*) mode
      write(*,*) mode
      if (mode.eq.2) then
        write(*,*) 'state name of data file (<=12 characters)'
        read (*,*) fi
        open(7,file=fi)
        read (7,*) fo
        open(3,file=fo)
        read(7,*) ifeat
        read(7,*) c1
        read(7,*) c2
        read(7,*) W0,D0,X0

        if (ifeat.eq.1) then
          read(7,*) N
          read(7,*) r(1)
          read(7,*) TTRT
          read(7,*) as(1)
          read(7,*) plens(1)
          rhos = N*as(1)*plens(1)
          if (rhos.ge.1.) then
            write(*,*) 'The synchronous traffic normalized throughput'
            write(*,*) 'Rhos = N*as*plens >= 1'
          end if
          read(7,*) BWT(1)
          if (BWT(1).gt.(TTRT-N*r(1))/N) then
            write(*,*) 'BWT > (TTRT-N*r)/N'
          end if
          read(7,*) Np
          rho = rhos
          do 211 j = 1,Np
            read(7,*) Tpri(1,j)
            read(7,*) aa(1,j)
            read(7,*) plena(1,j)
            rho = rho+aa(1,j)*plena(1,j)*N
            if (rho.ge.1) write(*,508) j,rho,j
211      continue
          do 212 i = 2,N
            r(i) = r(1)
            as(i) = as(1)
            plens(i) = plens(1)
            BWT(i) = BWT(1)
            do 212 j = 1,Np
              Tpri(i,j) = Tpri(1,j)
              aa(i,j) = aa(1,j)
212      plena(i,j) = plena(1,j)

          else if (ifeat.eq.2) then
            read(7,*) N1
            read(7,*) N2
            N = N1+N2
            call pmt(N1,N2,N,a)
```

```
read(7,*) rc
rr = rc*N
read(7,*) Np
read(7,*) TTRT
read(7,*) asc(1),asc(2)
read(7,*) plensc(1),plensc(2)
rho = asc(1)*plensc(1)*N1+asc(2)*plensc(2)*N2
if (rho.ge.1.) then
  write(*,*) 'The synchronous traffic normalized throughput'
  write(*,*) 'Rhos = as1*plens1+as2*plens2 >= 1'
end if
read(7,*) BWTc(1),BWTc(2)
if ((BWTc(1)*N1+BWTc(2)*N2+rr).gt.TTRT)
*   write(*,*) 'Synchronous BWT > (TTRT-walk time)'
do 531 j = 1,Np
  read(7,*) Tpric(1,j),Tpric(2,j)
  read(7,*) aac(1,j),aac(2,j)
  read(7,*) plenac(1,j),plenac(2,j)
  rho = rho+aac(1,j)*plenac(1,j)*N1+aac(2,j)*plenac(2,j)*N2
  if (rho.ge.1) write(*,508) j,rho,j
531 continue

NN(1) = N1
NN(2) = N2
do 533 i = 1,N
  r(i) = rc
  as(i) = asc(a(i))
  plens(i) = plensc(a(i))
  BWT(i) = BWTc(a(i))
  do 533 j = 1,Np
    Tpri(i,j) = Tpric(a(i),j)
    aa(i,j) = aac(a(i),j)
533   plena(i,j) = plenac(a(i),j)

else
  read(7,*) N
  read(7,*) Np
  read(7,*) TTRT
  rho = 0.
  temp = 0.
  do 221 i = 1,N
    read(7,*) r(i),BWT(i)
    read(7,*) as(i)
    read(7,*) plens(i)
    rho = rho+as(i)*plens(i)
    temp = temp+BWT(i)+r(i)
    read(7,*) (Tpri(i,j),j=1,Np)
    read(7,*) (aa(i,j),j=1,Np)
    read(7,*) (plena(i,j),j=1,Np)
    do 221 j = 1,Np
      rho = rho+aa(i,j)*plena(i,j)
      if (rho.ge.1.) write(*,*) 'The normalized throughput >= 1'
      if(temp.gt.TTRT) write(*,*) 'Synchronous BWT > TTRT-walktime'
    end if
    goto 99
  end if

write(*,*) 'Enter the output data file name'
read(*,*) fo
open(3,file=fo)

write(*,*) 'Enter Feature Selection'
write(*,*) '1. Symmetric System'
write(*,*) '2. 2 Classes of Stations'
```

fddi.f

```

write(*,*) '3. Different Loading from Station to Station'
read(*,*) ifeat
write(*,*) 'Enter the statistics collection start time (msec)'
read(*,*) c1
write(*,*) 'Enter the stop time (msec)'
read(*,*) c2
write(*,*) 'Enter w,d,x, for computing P(W>w), P(D>d), P(X>x)'
read(*,*) W0,D0,X0

if (ifeat.eq.1) then
  write(*,*) 'Enter the number of stations (N)'
  read(*,*) N
  write(*,502)
502  format(' Enter the walk time from a station to its neighboring'
*    ', station (in msec; r)')
  read(*,*) r(1)
  write(*,*) 'Enter Target Token Rotation Time (msec; TTRT)'
  read(*,*) TTRT
601  write(*,503)
503  format(/,' Synchronous traffic arrival rate',
*    ' (packets/msec/station; as)')
  read(*,*) as(1)
  write(*,*) 'Mean synchronous packet transmission time',
*    ' (msec; plens)'
  read(*,*) plens(1)
  rhos = N*as(1)*plens(1)
  if (rhos.ge.1.) then
    write(*,*) 'The synchronous traffic normalized throughput'
    write(*,*) 'Rhos = N*as*plens >= 1'
    write(*,*) 'Re-select synchronous loading parameters'
    goto 601
  end if

602  write(*,*) 'Enter the bandwidth time (msec; < (TTRT-N*r)/N)'
  write(*,*) ' (the max synchronous message transmission',
*    ' per visit)'
  read(*,*) BWT(1)
  if (BWT(1).gt.(TTRT-N*r(1))/N) then
    write(*,*) 'Re-select:BWT > (TTRT-N*r)/N'
    goto 602
  end if
  write(*,504)
504  format(/,' Enter number of message priority classes',
*    ' per station (Np)')
  read(*,*) Np
  rho = rhos
  do 401 j = 1,Np
    write(*,505) j,j
505  format(' Enter priority-',il,' threshold',
*    ' (msec; T_pri('',i,''))')
    read(*,*) Tpri(1,j)
603  write(*,506) j
506  format(' Enter priority-',il,' arrival rate',
*    ' (aa; packets/msec/station)')
    read(*,*) aa(1,j)
    write(*,507) j
507  format(' Enter priority-',il,
*    ' mean transmission time (msec; plena)')
    read(*,*) plena(1,j)
    ro = rho+aa(1,j)*plena(1,j)*N
    if (ro.ge.1) then
      write(*,508) j,ro,j
508  format(' Total normalized throughput for synchronous and '
*    ',/, asynchronous traffic of priority no lower than ',il,

```

```

*   ' is',f5.2,/, ' Re-select priority-',il,' loading parameters')
      goto 603
      end if
      rho = ro
401   continue
      do 402 i = 2,N
          r(i) = r(1)
          as(i) = as(1)
          plens(i) = plens(1)
          BWT(i) = BWT(1)
          do 402 j = 1,Np
              Tpri(i,j) = Tpri(1,j)
              aa(i,j) = aa(1,j)
402   plena(i,j) = plena(1,j)

      else if (ifeat.eq.2) then
*** ifeat = 2

      write(*,*) 'Enter the number of class-1 stations (N1)'
      read(*,*) N1
      write(*,*) 'Enter the number of class-1 stations (N2)'
      read(*,*) N2
      N = N1+N2
      call pmt(N1,N2,N,a)
      write(*,502)
      read(*,*) rc
      rr = rc*N
      write(*,504)
      read(*,*) Np
      write(*,*) 'Enter Target Token Rotation Time (msec; TTRT)'
      read(*,*) TTRT
811   write(*,813)
813   format(/, ' Synchronous traffic arrival rate for class-1 and ',
*   ' class-2 stations',/, ' (packets/msec/station) (as1, as2)')
      read(*,*) asc(1),asc(2)
      write(*,815)
815   format(' Mean synchronous packet transmission time for ',
*   ' class-1 and class-2 stations',/, ' (msec) (plens1, plens2)')
      read(*,*) plensc(1),plensc(2)
      rhos = asc(1)*plensc(1)*N1+asc(2)*plensc(2)*N2
      if (rhos.ge.1.) then
          write(*,*) 'The synchronous traffic normalized throughput'
          write(*,*) 'Rhos = as1*plens1+as2*plens2 >= 1'
          write(*,*) 'Re-select synchronous loading parameters'
          goto 811
      end if
812   write(*,816)
816   format(' Enter the BandWidth Time for class-1 and class-2 ',
*   ' stations (msec)',/, ' (BWT1*N1+BWT2*N2 < (TTRT-walk time)',/,
*   ' (the max synchronous message transmission per visit)')
      read(*,*) BWTc(1),BWTc(2)
      if ((BWTc(1)*N1+BWTc(2)*N2+rr).gt.TTRT) then
          write(*,*) 'Re-select: synchronous BWT > (TTRT-walk time)'
          goto 812
      end if
      write(*,*) 'Asynchronous Traffic'
      rho = rhos
      do 431 j = 1,Np
          write(*,805) j,j,j
805   format(' Enter priority-',il,' threshold for both classes',
*   ' (msec; T_pri(class1,',il,',) T_pri(class2,',il,',))')
          read(*,*) Tpric(1,j),Tpric(2,j)
803   write(*,806) j
806   format(' Enter priority-',il,' arrival rate for both classes'

```

fddi.f

```
*      , ' (packets/msec/station) (aa1 aa2) ' )
      read(*,*) aac(1,j),aac(2,j)
      write(*,807) j
807    format(' Enter priority-',i1,
*      ' mean transmission time (msec) (plena1 plena2) ' )
      read(*,*) plenac(1,j),plenac(2,j)
      ro = rho+aac(1,j)*plenac(1,j)*N1+aac(2,j)*plenac(2,j)*N2
      if (ro.ge.1) then
        write(*,508) j,ro,j
        goto 803
      end if
431    rho = ro

      NN(1) = N1
      NN(2) = N2
      do 433 i = 1,N
        r(i) = rc
        as(i) = asc(a(i))
        plens(i) = plensc(a(i))
        BWT(i) = BWTc(a(i))
        do 433 j = 1,Np
          Tpri(i,j) = Tpric(a(i),j)
          aa(i,j) = aac(a(i),j)
433    plena(i,j) = plenac(a(i),j)

      else
*** ifeat = 3

      write(*,*) 'Enter the number of stations (N)'
      read(*,*) N
      write(*,*) 'Enter walk times from station to station',
*      ' (msec) (r(i),i=1..N)'
      read(*,*) (r(i),i=1,N)
      rr = 0.
      do 921 i = 1,N
921    rr = rr+r(i)
      write(*,*) 'Enter Target Token Rotation Time (msec; TTRT)'
      read(*,*) TTRT
933    write(*,*) 'Synchronous traffic arrival rates',
*      ' (packets/msec/station; as(i),i=1..N)'
      read(*,*) (as(i),i=1,N)
      write(*,*) 'Mean synchronous packet transmission time',
*      ' (msec; plens(i),i=1..N)'
      read(*,*) (plens(i),i=1,N)
      rhos = 0.
      do 923 i = 1,N
923    rhos = rhos+as(i)*plens(i)
      if (rhos.ge.1.) then
        write(*,*) 'The synchronous traffic normalized throughput'
        write(*,*) 'Rhos = sum of s(i)*plens(i) >= 1'
        goto 933
      end if
932    temp = rr
      write(*,*) 'Enter the bandwidth times (msec; ',
*      '<TTRT-walktime) (BWT(i),i=1..N)'
      read(*,*) (BWT(i),i=1,N)
      do 924 i = 1,N
924    temp = temp+BWT(i)
      if (temp.gt.TTRT) then
        write(*,*) 'synchronous BWT > (TTRT-walk time)'
        goto 932
      end if
      write(*,*) 'Asynchronous Traffic'
      write(*,504)
```

```
      read(*,*) Np
      rho = rhos
      do 931 j = 1,Np
        write(*,915) j,j
915      format(' Enter priority-',il,' threshold for stations 1'
        *      ',...N (msec)',/,', (T_pri(i,',il,'),i=1..N)')
        read(*,*) (Tpri(i,j),i=1,N)
903      write(*,906) j,j
906      format(' Enter priority-',il,' arrival rate for stations',
        *      ' 1,...N (packets/msec/station)',/,', (aa(i,',il,'),i=1..N)')
        read(*,*) (aa(i,j),i=1,N)
942      write(*,907) j,j
907      format(' Enter priority-',il,' mean transmission time for',
        *      ' stations 1..N (msec)',/,', (plena(i,',il,'),i=1..N)')
        read(*,*) (plena(i,j),i=1,N)
        ro = rho
        do 941 i = 1,N
941      ro = ro+aa(i,j)*plena(i,j)
        if (ro.ge.1) then
          write(*,508) j,ro,j
          goto 942
        end if
931      rho = ro

      end if
      ***** End of Input Section *****

99      do 403 i = 1,N
        k = i*32
        ss(i) = k
        seeds(i) = k+1
        do 403 j = 1,Np
          sa(i,j) = k+2+j
403      seeda(i,j) = k+10+j

ccc = c2/10.
do 21 nq = 1,N
  os(nq) = 0
  cs(nq) = 1./as(nq)
  ys(nq) = plens(nq)*2.
  s1 = dmod( sp*ss(nq), B)
  s2 = dmod( sp*seeds(nq), B)
  p1 = -dlog(s1/B)*cs(nq)
  Aas(nq,0) = max(0.000001,p1)
  works(nq,0) = max(0.000001,s2/B*ys(nq))
  ss(nq) = s1
  seeds(nq) = s2
  do 21 kp = 1,Np
    oa(nq,kp) = 0
    ca(nq,kp) = 1./aa(nq,kp)
    ya(nq,kp) = plena(nq,kp)*2.
    s1 = dmod( sp*sa(nq,kp), B)
    s2 = dmod( sp*seeda(nq,kp), B)
    p1 = -dlog(s1/B)*ca(nq,kp)
    Aaa(nq,kp,0) = max(0.000001,p1)
    worka(nq,kp,0) = max(0.000001,s2/B*ya(nq,kp))
    sa(nq,kp) = s1
21  seeda(nq,kp) = s2

*** 1st cycle ***
*** no station is allowed to transmit during 1st cycle
Rs = 0.
zf = .false.
TRTmax = 0.
```

```
do 5 i = 1,N
  tails(i) = 0
  heads(i) = 0
  res(i) = works(i,0)
  zcs(i) = 0
  Ds(i) = 0.
  do 6 j = 1,Np
    zca(i,j) = 0
    Da(i,j) = 0.
    heada(i,j) = 0
    taila(i,j) = 0
    Ua(i,j) = 0.
6    continue
    Us(i) = 0.
    Rs = Rs+r(i)
5    continue

  if (rho.lt.0.999) then
    scale = Rs*rho/(1-rho)*3./(mind+1.)
  else
    scale = TTRT*2.0/(mind+1.0)
  end if

*** 2nd cycle ***
*** only synchronous traffic is allowed during 2nd cycle
  t = Rs
  do 10 nq = 1,N
    call unfw(nq,t)
    t0(nq) = t
    TRT = t-t0(nq)
    if (TRT.gt.TRTmax) then
      TRTmax = TRT
      nmax = it
      tmax = t
    end if

    v(nq) = 0.
    y = 0.
    gf = .false.
8    if (Us(nq).eq.0.0.or.gf) goto 10
    diff = BWT(nq)-y
    if (Us(nq).ge.diff) then
      call qwus(nq,t,diff)
      gf = .true.
    else
      temp = Us(nq)
      call qwus(nq,t,temp)
      y = y+temp
      goto 8
    end if
10   t = t+r(nq)

*** afterwards- to the end ***
*** both synchronous and asynchronous messages are allowed
*** assume: exhausted service,
***         synchronous message is served first
***         priority 1 is the highest priority
  ck = ccc
  it = 2
30  if (t.le.c2) then

    if (t.gt.c2) goto 77
    it = it+1
```

```
      if (.not.zf) then
        if (t.ge.cl) then
          nstart = it
          tstart = t
          zf = .true.
        end if
      end if

***      display in progress
      if (t.gt.ck) then
        ck = ck+ccc
        write(*,71) t,t/c2*100.
71      format(' [Simulation Time]',f12.5,' msec',3x,f6.2,'%')
      end if

      do 31 nq = 1,N
        TRT = t-t0(nq)
        if (TRT.gt.TRTmax) then
          TRTmax = TRT
          nmax = it
          tmax = t
        end if
        vtemp = TRT-v(nq)

        call unfw(nq,t)
        t0(nq) = t

        if (zf) then
          if (Xs(nq).gt.X0) PrXs(nq) = PrXs(nq)+1
          if (Xs(nq).gt.mind) then
            txs(nq,mind) = txs(nq,mind)+1
          else
            txs(nq,Xs(nq)) = txs(nq,Xs(nq))+1
          end if
          do 38 i = 1,Np
            if (Xa(nq,i).gt.X0) PrXa(nq,i) = PrXa(nq,i)+1
            if (Xa(nq,i).gt.mind) then
              txa(nq,i,mind) = txa(nq,i,mind)+1
            else
              txa(nq,i,Xa(nq,i)) = txa(nq,i,Xa(nq,i))+1
            end if
38          continue
        end if

        do 496 i = 1,Np
496      tua(i) = 0.
        v(nq) = 0.
        y = 0.
        gf = .false.
32      if (Us(nq).lt.0.0000001.or.gf) goto 34
33      diff = BWT(nq)-y
        if (diff.lt.0.0000001) then
          gf = .true.
          goto 34
        end if
        if (Us(nq).ge.diff) then
          call qwus(nq,t,diff)
          gf = .true.
        else
          temp = Us(nq)
          call qwus(nq,t,temp)
          y = y+temp
          goto 32
        end if
```

```

34      do 35 j = 1,Np
        if ((TRT+v(nq)).ge.Tpri(nq,j)) then
          if (gf.or.Us(nq).eq.0.) goto 35
          goto 33
        else if (Ua(nq,j).gt.0.0000001) then
          ww = worka(nq,j,heada(nq,j))
          if (zf) tua(j) = tua(j)+ww

          t2 = t+ww
          if (ofs(nq)) then
            if (tofs(nq).lt.t2) call blocks(nq,tofs(nq),t2)
            goto 54
          end if
          ttl = Aas(nq,tails(nq))
52      if (ttl.lt.t2) then
            Us(nq) = Us(nq)+works(nq,tails(nq))
            k = mod(tails(nq)+1,201)
            if (k.eq.heads(nq)) then
              call blocks(nq,ttl,t2)
              goto 54
            end if
            if (zf) then
              if (Xs(nq).gt.mind) then
                axs(nq,mind) = axs(nq,mind)+1
                tss(nq,mind)=tss(nq,mind)+ttl-tjumps(nq)
              else
                axs(nq,Xs(nq)) = axs(nq,Xs(nq))+1
                tss(nq,Xs(nq))=tss(nq,Xs(nq))+ttl-tjumps(nq)
              end if
            end if
            Xs(nq) = Xs(nq)+1
            tjumps(nq) = ttl
            s1 = dmod(sp*ss(nq),B)
            p1 = -dlog(s1/B)*cs(nq)
            ttl = ttl+max(0.000001,p1)
            tails(nq) = k
            ss(nq) = s1
            Aas(nq,k) = ttl
            s2 = dmod(sp*seeds(nq),B)
            works(nq,k) = max(0.000001,s2/B*ys(nq))
            seeds(nq) = s2
            goto 52
          end if
54      do 51 i = 1,Np
        if (ofa(nq,i)) then
          if (tofa(nq,i).lt.t2) call blocka(nq,i,tofa(nq,i),t2)
          goto 51
        end if
53      ttl = Aaa(nq,i,taila(nq,i))
        if (ttl.lt.t2) then
          Ua(nq,i) = Ua(nq,i)+worka(nq,i,taila(nq,i))
          k = mod(taila(nq,i)+1,101)
          if (k.eq.heada(nq,i)) then
            call blocka(nq,i,ttl,t2)
            goto 51
          end if
          if (zf) then
            if (Xa(nq,i).gt.mind) then
              axa(nq,i,mind) = axa(nq,i,mind)+1
              tsa(nq,i,mind)=tsa(nq,i,mind)+ttl-tjumpa(nq,i)
            else
              axa(nq,i,Xa(nq,i)) = axa(nq,i,Xa(nq,i))+1
              tsa(nq,i,Xa(nq,i)) = tsa(nq,i,Xa(nq,i))+ttl-
                tjumpa(nq,i)
            end if
          end if
        end if

```

```

        end if
        end if
        Xa(nq,i) = Xa(nq,i)+1
        tjumpa(nq,i) = ttl
        s1 = dmod(sp*sa(nq,i),B)
        s2 = dmod(sp*seeda(nq,i),B)
        p1 = -dlog(s1/B)*ca(nq,i)
        Aaa(nq,i,k) = ttl+max(0.000001,p1)
        worka(nq,i,k) = max(0.000001,s2/B*ya(nq,i))
        sa(nq,i) = s1
        seeda(nq,i) = s2
        taila(nq,i) = k
        goto 53
    end if
    continue

Xa(nq,j) = Xa(nq,j)-1
if (ofa(nq,j)) then
    ofa(nq,j) = .false.
    k = mod(taila(nq,j)+1,101)
    Aaa(nq,j,k) = tofa(nq,j)
    worka(nq,j,k) = wofa(nq,j)
    taila(nq,j) = k
end if

if (zf) then
    temp = t-Aaa(nq,j,heada(nq,j))
    if (temp.gt.W0) PrWa(nq,j) = PrWa(nq,j)+1
    if ((temp+ww).gt.D0) PrDa(nq,j) = PrDa(nq,j)+1
    sigwa(nq,j) = sigwa(nq,j)+temp*temp
    sigda(nq,j) = sigda(nq,j)+(temp+ww)*(temp+ww)
    kd = min((temp+ww)/scale,mind)
    kw = min(temp/scale,mind)
    pda(nq,j,kd) = pda(nq,j,kd)+1
    pwa(nq,j,kw) = pwa(nq,j,kw)+1
    Wa(nq,j) = Wa(nq,j)+temp
    Da(nq,j) = Da(nq,j)+temp+ww
    if (Xa(nq,j).ge.mind) then
        dxa(nq,j,mind) = dxa(nq,j,mind)+1
        tsa(nq,j,mind) = tsa(nq,j,mind)+t2-tjumpa(nq,j)
    else
        dxa(nq,j,Xa(nq,j)) = dxa(nq,j,Xa(nq,j))+1
        tsa(nq,j,Xa(nq,j)+1) = tsa(nq,j,Xa(nq,j)+1)+t2-
            tjumpa(nq,j)
    end if
    zca(nq,j) = zca(nq,j)+1
end if
tjumpa(nq,j) = t2
Ua(nq,j) = Ua(nq,j)-ww
v(nq) = v(nq)+ww
t = t2
heada(nq,j) = mod(heada(nq,j)+1,101)
goto 32
end if

```

35

continue

```

***-----
*** The token leaves station nq, takes walk time r(nq)
*** and goes to station (nq+1)
***-----

```

```

if (zf) then
    tqz(nq) = tqz(nq)+y
    tqsv(nq) = tqsv(nq)+y*y
do 499 i = 1,Np

```

```

499      tqa(nq,i) = tqa(nq,i)+tua(i)
      tqav(nq,i) = tqav(nq,i)+tua(i)*tua(i)
      dwell(nq) = dwell(nq)+v(nq)
      dwellv(nq) = dwellv(nq)+v(nq)*v(nq)
      ctime(nq) = ctime(nq)+vtemp+v(nq)
      ctimev(nq) = ctimev(nq)+(vtemp+v(nq))*(vtemp+v(nq))
      dpar(nq) = dpar(nq)+vtemp
      dparv(nq) = dparv(nq)+vtemp*vtemp
      end if
31      t = t+r(nq)
      goto 30
      end if

***      Output Section      ***
77      write(3,701)
701      format(/,' ** Performance of a Timed Token Rotation'
*, ' Protocol (FDDI-type) Ring Networka **')

      cyclen = it-nstart+1
      dtr = 0.
      do 555 i = 1,N
555      dtr = dtr+dwell(i)
      thn = dtr/(t-tstart)
      dtr = dtr/cyclen
      sf = .false.
      af = .false.
      do 577 i = 1,N
          if (BWT(i).gt.0.) sf = .true.
          do 577 j = 1,Np
              if (Tpri(i,j).gt.0.) af = .true.
577      continue

      if (ifeat.eq.1) then
          write(3,702) 'Symmetric Systems'
      else if (ifeat.eq.2) then
          write(3,702) '2 Classes of Stations'
      else
          write(3,702) 'Different Loading from Station to Station'
      end if
702      format(/,' Feature Selected: ',a41)
      write(3,703) cl,c2,TTRT,N,Np,1.0-Rs/2.0/TTRT,rho,thn,
*, (t-tstart)/cyclen,dtr,TRTmax,nmax,tmax
703      format(' Statistics Start (msec):',f9.1,
*, '/', ' Statistics Stop (msec):',f9.1,
*, '/', ' TTRT (msec):',f9.3,
*, '/', ' Number of Stations (N):',i9,
*, '/', ' Number of Priorities (Np):',i9,
*, '/', ' Max Throughput (1-walktime/2TTRT):',f9.4,
*, '/', ' Normalized Throughput (specified):',f9.4,
*, '/', ' Normalized Throughput (realized):',f9.4,
*, '/', ' Realized Mean Cycle Time (msec):',f9.4,
*, '/', ' Realized Mean Dwell Time (msec):',f9.4,
*, '/', ' Max Cycle Time (msec):',f9.4,
*, ' at',i6,'th cycle, t=',f9.1)
      if (ifeat.eq.1) then
          write(3,704) r(1)
          if (sf) write(3,758) BWT(1),as(1),plens(1)
          if (af) then
              write(3,759) 'i','i'
              write(3,705) (i,Tpri(1,i),aa(1,i),plena(1,i),i=1,Np)
          end if
      else if (ifeat.eq.2) then
          do 778 j = 1,2
              write(3,776) j

```

```
      ch = char(48+j)
      write(3,704) r(j)
      if (sf) write(3,758) BWT(j),as(j),plens(j)
      if (af) then
        write(3,759) ch,ch
        write(3,705) (i,Tpri(j,i),aa(j,i),plena(j,i),i=1,Np)
      end if
778  continue
      write(3,747)
      k = mod(N,20)
      if (mod(N,20).eq.0) then
        k = N/20
      else
        k = N/20+1
      end if
      do 742 i = 1,k
        j = min(i*20,N)
        write(3,749) (ii,ii=(20*(i-1)+1),j)
742  write(3,746) (a(ii),ii=(20*(i-1)+1),j)
747  format(/,' System Configuration')
749  format(/,' station',20i4)
746  format(' ' class: ',20i4)
      else
        do 777 j = 1,N
          write(3,776) j
          ch = char(48+j)
          write(3,704) r(j)
          if (sf) write(3,758) BWT(j),as(j),plens(j)
          if (af) then
            write(3,759) ch,ch
            write(3,705) (i,Tpri(j,i),aa(j,i),plena(j,i),i=1,Np)
          end if
777  continue
776  format(/,' Station',i3)
      end if
704  format(' Walk Time          (r;msec):',f10.4)
758  format(' Bandwidth Time (BWT;msec):',f10.4,
*      /,' Arrival Rate for Synchronous Traffic (packets/msec',
*      '/station):',f9.4,
*      /,' Mean Packet Length for Synchronous Traffic      ',
*      ' (msec):',f9.4)
759  format(' Asynchronous Traffic: Prioriry-j',7x,'T_pri      aa('
*      ,Al,',j) plena(',Al,',j)')
705  format(33x,i1,3f12.4)
      write(3,65) it
65  format(/,' Token goes',i7,' cycles in simulation')

      kk = 1
999  dm = 0.
      wm = 0.
      xm = 0.
      dv = 0.
      wv = 0.
      xv = 0.
      am = 0.
      do 721 i = 1,N
        if (ifeat.eq.2.and.a(i).ne.kk) goto 721
        if (sf) then
          if (zcs(i).eq.0) then
            Ws(i) = -1
            Ds(i) = -1
          else
            Ws(i) = Ws(i)/zcs(i)
            Ds(i) = Ds(i)/zcs(i)
```

```

end if

tqs(i) = tqs(i)/cyclen
tqsv(i) = sqrt(tqsv(i)/cyclen-tqs(i)*tqs(i))

X1 = 0.
X2 = 0.
do 67 j = 0,mind
  X1 = X1+j*txs(i,j)
  X2 = X2+j*j*txs(i,j)
  tds(i) = tds(i)+pds(i,j)
  tws(i) = tws(i)+pws(i,j)
  tzs(i) = tzs(i)+txs(i,j)
  tys(i) = tys(i)+tss(i,j)
  azs(i) = azs(i)+axs(i,j)
67  dzs(i) = dzs(i)+dxs(i,j)
  Xsm(i) = X1/tzs(i)
  Xsv(i) = sqrt(X2/tzs(i)-Xsm(i)*Xsm(i))
  am = am+as(i)
  dm = dm+Ds(i)*as(i)
  wm = wm+Ws(i)*as(i)
  xm = xm+Xsm(i)
  xv = xv+X2/tzs(i)
  if (zcs(i).gt.0) then
    dv = dv+sigds(i)/zcs(i)*as(i)
    wv = wv+sigws(i)/zcs(i)*as(i)
    sigws(i) = sqrt(sigws(i)/zcs(i)-Ws(i)*Ws(i))
    sigds(i) = sqrt(sigds(i)/zcs(i)-Ds(i)*Ds(i))
  end if
end if
721 continue

if (ifeat.eq.2) then
  m = NN(kk)
  write(3,725) ' ** Class-'//char(48+kk)
else
  m = N
  write(3,725) '          '
end if
dm = dm/am
wm = wm/am
xm = xm/m
725  format(/,a11,9x,'E(X)  sigma(X)',6x,'E(W)  sigma(W)',6x,
*      'E(D)  sigma(D)')
if (sf) write(3,722) xm,sqrt(xv/m-xm*xm),wm,sqrt(wv/am-wm*wm),
*      dm,sqrt(dv/am-dm*dm)
722  format(' Sync Traffic:',6f10.4)

do 723 j = 1,Np
  am = 0.
  dm = 0.
  wm = 0.
  xm = 0.
  dv = 0.
  wv = 0.
  xv = 0.
do 727 i = 1,N
  if (ifeat.eq.2.and.a(i).ne.kk) goto 727
  tqa(i,j) = tqa(i,j)/cyclen
  tqav(i,j) = sqrt(tqav(i,j)/cyclen-tqa(i,j)*tqa(i,j))
  if (zca(i,j).eq.0) then
    Wa(i,j) = -1
    Da(i,j) = -1
  else

```

```

      Wa(i,j) = Wa(i,j)/zca(i,j)
      Da(i,j) = Da(i,j)/zca(i,j)
    end if
    X1 = 0.
    X2 = 0.
    do 764 k = 0,mind
      X1 = X1+k*txa(i,j,k)
      X2 = X2+k*k*txa(i,j,k)
      tda(i,j) = tda(i,j)+pda(i,j,k)
      twa(i,j) = twa(i,j)+pwa(i,j,k)
      tza(i,j) = tza(i,j)+txa(i,j,k)
      tya(i,j) = tya(i,j)+tsa(i,j,k)
      aza(i,j) = aza(i,j)+axa(i,j,k)
764    dza(i,j) = dza(i,j)+dxa(i,j,k)
      Xam(i,j) = X1/tza(i,j)
      Xav(i,j) = sqrt(X2/tza(i,j)-Xam(i,j)*Xam(i,j))
      am = am+aa(i,j)
      dm = dm+Da(i,j)*aa(i,j)
      wm = wm+Wa(i,j)*aa(i,j)
      xm = xm+Xam(i,j)
      if (zca(i,j).gt.0) then
        dv = dv+sigda(i,j)/zca(i,j)*aa(i,j)
        wv = wv+sigwa(i,j)/zca(i,j)*aa(i,j)
        sigwa(i,j) = sqrt(sigwa(i,j)/zca(i,j)-Wa(i,j)*Wa(i,j))
        sigda(i,j) = sqrt(sigda(i,j)/zca(i,j)-Da(i,j)*Da(i,j))
      end if
      xv = xv+X2/tza(i,j)
727    continue

      dm = dm/am
      wm = wm/am
      xm = xm/m
      write(3,724) j,xm,sqrt(xv/m-xm*xm),wm,sqrt(wv/am-wm*wm),
*          dm,sqrt(dv/am-dm*dm)
724    format(' Async: pri-',i1,6f10.4)
723    continue

      if (ifeat.eq.2.and.kk.eq.1) then
        kk = kk+1
        goto 999
      end if

      write(3,70)
70    format(/,' Station #stat      E(X)  sigma(X)      E(W)  ',
* 'sigma(W)      E(D)  sigma(D)',/,75('*'))

      do 68 i = 1,N

        dwell(i) = dwell(i)/cyclen
        dwellv(i) = sqrt(dwellv(i)/cyclen-dwell(i)*dwell(i))
        ctime(i) = ctime(i)/cyclen
        ctimev(i) = sqrt(ctimev(i)/cyclen-ctime(i)*ctime(i))
        dpar(i) = dpar(i)/cyclen
        dparv(i) = sqrt(dparv(i)/cyclen-dpar(i)*dpar(i))

        if (sf) then
          write(3,66) i,0,zcs(i),Xsm(i),Xsv(i),Ws(i),sigws(i),Ds(i),
*          sigds(i)
66        format(/,i4,i2,i8,6f10.4)
        end if

        if (.not.af) goto 68
        do 968 j = 1,Np
          if (sf.or.j.gt.1) then

```

fddi.f

```
      write(3,69) j,zca(i,j),Xam(i,j),Xav(i,j),Wa(i,j),
*      sigwa(i,j),Da(i,j),sigda(i,j)
      else
      write(3,66) i,j,zca(i,j),Xam(i,j),Xav(i,j),Wa(i,j),
*      sigwa(i,j),Da(i,j),sigda(i,j)
      end if
69      format(5x,i1,i8,6f10.4)
968      continue

68      continue

      write(3,78) W0,D0,X0
78      format(/,' Station      E(Vj)      sigma(Vj)      Pr(W>',f7.3,
*      'Pr(D>',f7.3,')      Pr(X>',i2,'')',/,73('**'))
      do 290 i = 1,N
      if (sf) write(3,79) i,0,tqs(i),tqsv(i),1.0*PrWs(i)/zcs(i),
*      1.0*PrDs(i)/zcs(i),PrXs(i)/tzs(i)
      if (.not.af) goto 290
      do 291 j = 1,Np
      if (sf.or.j.gt.1) then
      write(3,279) j,tqa(i,j),tqav(i,j),1.0*PrWa(i,j)/zca(i,j),
*      1.0*PrDa(i,j)/zca(i,j),PrXa(i,j)/tza(i,j)
      else
      write(3,79) i,j,tqa(i,j),tqav(i,j),1.0*PrWa(i,j)/zca(i,j),
*      1.0*PrDa(i,j)/zca(i,j),PrXa(i,j)/tza(i,j)
      end if
291      continue
79      format(/,i4,i2,2f12.5,3f14.5)
279      format(i6,2f12.5,3f14.5)
290      continue

      write(3,765)
765      format(/,' Station',8x,'E(V)      sigma(V)',8x,'E(C)      sigma(C)',
*      6x,'E(C-V)      sigma(C-V)',/,80('**'))
      do 768 i = 1,N
768      write(3,596) i,dwell(i),dwellv(i),ctime(i),ctimev(i),
*      dpar(i),dparv(i)
596      format(i4,4x,6f12.7)

      do 91 i = 1,N
      write(3,571) i
571      format(/,' ** Station',i4)
      if (sf) then
      write(3,706) as(i)*plens(i)
706      format(/,' Synchronous Traffic:',
*      ' Normalized throughput (Rho_s)=' ,f9.6)
      write(3,708)
708      format(/,' j      P(X=j)      P(Xa=j)      P(Xd=j)      P(Xt=j)',6x,
*      'w      P(W<=w)',6x,'d      P(D<=d)')
      g = 0.
      g1 = 0.
      g2 = 0.
      do 92 j = 0,mind
      g = g+scale
      g1 = g1+pws(i,j)/tws(i)
      g2 = g2+pds(i,j)/tds(i)
92      write(3,93) j,txs(i,j)/tzs(i),axs(i,j)/azs(i),dxs(i,j)/dzs(i),
*      tss(i,j)/tys(i),g,g1,g,g2
93      format(i3,4f9.5,2(f9.4,f9.5))
      end if

      if (.not.af) goto 91
      do 791 j = 1,Np
      write(3,726) j,aa(i,j)*plena(i,j)
```

```
726  format(/,' Async: priority-',i1,5x,  
*      ' Normalized throughput (Rho_a)=' ,f9.6)  
      write(3,708)  
      g = 0.  
      g1 = 0.  
      g2 = 0.  
      do 791 k = 0,mind  
        g = g+scale  
        g1 = g1+pwa(i,j,k)/twa(i,j)  
        g2 = g2+pda(i,j,k)/tda(i,j)  
791    write(3,93) k,txa(i,j,k)/tza(i,j),axa(i,j,k)/aza(i,j),  
*      dxa(i,j,k)/dza(i,j),tsa(i,j,k)/tya(i,j),g,g1,g,g2  
  
91    continue  
  
end  
  
subroutine qwus(nq,t,dif)  
implicit real*8 (A-H,O-Z)  
logical zf,ofs(20),ofa(20,3)  
real*8 Aas(20,0:200),works(20,0:200),Us(20),  
*      Aaa(20,3,0:100),worka(20,3,0:100),Ua(20,3),Ws(20),Ds(20),  
*      cs(20),ca(20,3),ys(20),ya(20,3),res(20),v(20)  
*      ,sigws(20),sigds(20),tofs(20),tofa(20,3),wofs(20),wofa(20,3)  
*      ,tss(20,0:50),tsa(20,3,0:50),tjumps(20),tjumpa(20,3)  
integer tails(20),taila(20,3),ss(20),sa(20,3),zcs(20)  
*      ,heads(20),heada(20,3),seeds(20),seeda(20,3),os(20),oa(20,3)  
*      ,Xs(20),Xa(20,3),axs(20,0:50),dxs(20,0:50),axa(20,3,0:50)  
*      ,pds(20,0:50),pws(20,0:50),PrWs(20),PrDs(20)  
common Aas,Aaa,works,worka,Us,Ua,heads,heada,tails,taila,N,Np,  
*      Xs,Xa,axs,dxs,axa,mind,tss,tsa,tjumps,tjumpa  
common /syn/v,res,Ws,Ds,sigws,sigds,scale,pws,pds,zcs,PrWs,PrDs,  
*      W0,D0 /bs/ofs,ss,tofs,cs,os,seeds,ys,wofs /bc/sp,B,zf  
*      /ba/ofa,sa,tofa,ca,oa,seeda,ya,wofa  
v(nq) = v(nq)+dif  
t2 = t+dif  
ttl = Aas(nq,tails(nq))  
tt2 = t+res(nq)  
2  if ((ttl).ge.t2.and.  
*      tt2.ge.(0.0000001+t2)) goto 4  
if (ttl.lt.tt2) then  
  if (zf) then  
    if (Xs(nq).gt.mind) then  
      axs(nq,mind) = axs(nq,mind)+1  
      tss(nq,mind) = tss(nq,mind)+ttl-tjumps(nq)  
    else  
      axs(nq,Xs(nq)) = axs(nq,Xs(nq))+1  
      tss(nq,Xs(nq)) = tss(nq,Xs(nq))+ttl-tjumps(nq)  
    end if  
  end if  
  Xs(nq) = Xs(nq)+1  
  tjumps(nq) = ttl  
  j = mod(tails(nq)+1,201)  
  if (j.eq.heads(nq)) then  
    temp = Aas(nq,tails(nq))  
6    if (zf) os(nq) = os(nq)+1  
    s1 = dmod(sp*ss(nq),B)  
    p1 = -dlog(s1/B)*cs(nq)  
    temp = temp+max(0.000001,p1)  
    ss(nq) = s1  
    if (temp.lt.t2) goto 6  
    goto 4
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302 and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1 AGENCY USE ONLY (Leave blank)		2 REPORT DATE September 1990		3 REPORT TYPE AND DATES COVERED Final	
4 TITLE AND SUBTITLE DELAY-THROUGHPUT PERFORMANCE EVALUATOR FOR DISTRIBUTED SYSTEMS TDMA and Token Ring Schemes (Version 1)				5 FUNDING NUMBERS C: N66001-89-M-BY18 Mod: P00001 PE: 0602721N WU: DN 088524	
6 AUTHOR(S)				8 PERFORMING ORGANIZATION REPORT NUMBER IRI-NOSC-8902	
7 PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) IRI Corporation 4544 Totana Dr Tarzana, CA 91356				10 SPONSORING/MONITORING AGENCY REPORT NUMBER NOSC TD 1931	
9 SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Naval Ocean Systems Center San Diego, CA 92152-5000					
11 SUPPLEMENTARY NOTES					
12a DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.				12b DISTRIBUTION CODE	
13 ABSTRACT (Maximum 200 words) This work involves the development of a delay-throughput performance evaluator for distributed systems. Currently planned and future Navy distributed integrated computer and communications systems involve the extensive use of medium access control procedures for sharing distributed communications, processing, and computing resources among distributed stations. This work contributes to the development of methods and tools for carrying out modeling, performance evaluation, analysis, and design of such systems.					
14 SUBJECT TERMS distributed systems				15 NUMBER OF PAGES 160	
				16 PRICE CODE	
17 SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18 SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19 SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20 LIMITATION OF ABSTRACT SAME AS REPORT		