NISTIR 4530

Validation of an OSI Transport Class 4 Simulator

Okhee Kim
Sharon Heatley
Bob Bishop

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
Computer Systems Laboratory
Gaithersburg, MD 20899

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

NIST

NISTIR 4530

Validation of an OSI Transport Class 4 Simulator

Okhee Kim
Sharon Heatley
Bob Bishop

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
Computer Systems Laboratory
Gaithersburg, MD 20899

May 1991

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

VALIDATION OF AN OSI TRANSPORT CLASS 4 SIMULATOR

Okhee Kim
Sharon Heatley
Bob Bishop

National Institute of Standards and Technology
Gaithersburg, Maryland

1.0 INTRODUCTION

For the last six years, the National Institute of Standards and Technology (NIST) has pursued a program in network protocol performance.[1] This program studied the performance of OSI Transport Class 4 [1] over a satellite link [2,3,4]. More recent work studied OSI Transport Class 4 over local area networks [5,6,7] IEEE 802.3 (CSMA/CD) [8] and IEEE 802.4 (token bus) [9].

NIST is now studying scheduling algorithms for OSI Transport Class 4. Most work on scheduling in communications protocols has considered only the MAC layer. Token bus, token ring[10], and FDDI [11] all have priority mechanisms. But mechanisms for scheduling must be extended into the upper OSI layers to be effective.

The Mills-Franx paper [6] referenced above investigates the usefulness of Transport Class 4 for real-time factory applications. This paper concludes that "OSI protocol standards, as now specified, do not provide adequate mechanisms for guaranteeing real-time performance for selected connections or messages."

The Transport service standard provides for Quality of Service (QOS) parameters allowing the Transport users to communicate to the Transport layer their performance needs for a particular connection. These QOS parameters include desired throughput, desired delay, priority of connection, and residual error rate. QOS parameters are negotiated between the two Transport users and the Transport service provider at connection establishment time. However, no guidance is given in the Transport standards as to how to implement QOS. In addition, in current implementations, no use is made of the QOS parameters. In order to give a particular connection a higher QOS, such as shorter delay or more throughput than other connections, it is necessary to give the connection more access to resources such as CPU time and channel capacity via scheduling algorithms. QOS quantities (throughput, delay) should be measured on an ongoing basis. Resources would then be reassigned dynamically to meet quality of service goals.

In one interesting study of how to implement QOS in higher protocol layers, Jacquet and Sedillot [12] investigated real-time message scheduling for a general layered protocol architecture such as OSI. In their scheme, each message is given a priority and a deadline. There is a central scheduler which allocates the CPU to a message based on a function of each message's priority and deadline. If a message misses its deadline, it is discarded to

[1] Certain commercial equipment is indentified in this paper in order to adequately specify the experimental procedure. Such indentification does not imply recommendation or endorsement by NIST, nor does it imply that the equipment identified is necessarily the best available for the purpose.

avoid congesting the network with old messages.

NIST has chosen to study possible scheduling mechanisms for Transport Class 4 by first developing a detailed Transport Class 4 simulator. This discrete event simulator is written in Simscript II.5 and simulates the Intel iNA960 implementation [13] of Transport Class 4 running on a 186/51 front-end board with a 286/10 host. Processing times such as the time required for communication between the host and the front-end board and the time required to send a data (DT) Transport Protocol Data Unit (TPDU) were measured by instrumenting the iNA960 code. These processing times are inputs into the model. A set of experiments were performed on both the iNA960 software and the Simscript model and the results were compared in order to validate the model.

In this paper, the model is described, the processing times which serve as inputs into the model are specified and the validation experiments are documented.

As a continuation of the work, NIST plans to implement various scheduling algorithms for implementing QOS in the Simscript Transport Class 4 model. Since the model has been carefully validated, the results of experiments with different algorithms using the model should be similar to the results obtained by actually changing the iNA960 to implement these algorithms. Changing the model instead of the iNA960 implementation is attractive because far fewer resources are required to implement model changes. Running model experiments is also less time-consuming than running live experiments.

In section 2, the OSI protocols with which we are working are described. The live testbed, hardware and software, is described in section 3. The simulator is discussed in section 4, including a list of the processing times which are part of the input to the simulator. Section 5 contains the validation experiments. The report is summarized in section 6.

2.0 PROTOCOL LAYERS

2.1 TERMINOLOGY

In this paper, the terms Transport Service Data Unit (TSDU) and Transport Protocol Data Unit (TPDU) are frequently used. The Transport user passes data to Transport as TSDUs. These TSDUs are also referred to as user messages. A Transport entity communicates with another Transport entity via TPDUs. The Transport layer uses Network layer services to transmit TPDUs. Each TSDU is transferred as one or more Data Transfer (DT) TPDUs.

2.2 TRANSPORT

Transport Class 4 offers virtual circuits, error detection and recovery, flow control, segmentation of messages, and in-order delivery of messages. The error detection, recovery, and flow control functions are accomplished using AK (acknowledge) TPDUs transmitted by the receiving station. Each DT TPDU has a field containing a sequence number. Every AK TPDU contains the acknowledgment of receipt of all DT TPDUs with lower sequence numbers. The AK also contains a credit field. The transmitting station is allowed to send DT

TPDUs with sequence number K where

AK seq no <= K < (AK seq no + AK credit)

This range of permitted sequence numbers is known as the window. There are retransmission timers on the transmitting station which are started when DTs are sent and restarted when AKs are received. If no AK is received for a period of time and the retransmission timer expires, then unacknowledged DT TPDUs are retransmitted.

In order to understand the results of the validation experiments, it is important to understand segmentation. The user may wish to send TSDUs that are too large to send over the communications network in a single packet. An IEEE 802.3 LAN, for example, limits the maximum Data Link frame size to 1518 octets. One of the services offered by Transport Class 4 is to segment TSDUs that are too large to fit into network packets into multiple TPDUs where each TPDU fits into a single network packet. Thus the user of Transport Class 4 can pass arbitrarily large TSDUs to Transport without knowledge of the network packet size.

2.3 NETWORK

The Network protocol is the Connectionless Network Protocol (CLNP) [14]. The CLNP has the capability to segment messages but, for the validation experiments, the non-segmenting subset was used.

2.4 DATA LINK

The Data Link layer is divided into two sublayers. The top sublayer is Logical Link Control (LLC) [15] Class 1 Type 1 which provides a datagram service. The bottom sublayer is the Medium Access Control (MAC) layer. For the validation experiments, the IEEE 802.3 Carrier Sense Multiple Access with Collision Detection (CSMA/CD) MAC protocol was used.

3.0 LIVE EXPERIMENTAL ENVIRONMENT

3.1 HARDWARE OVERVIEW

The Intel 310 system, as shown in Figure 1, consists of a 286/10 host processor board, a 1 megabyte Multibus memory board, and a 186/51 front-end communications board. A 286/10 host board has a 6 MHz 80286 CPU and a 186/51 front-end board has a 6 MHz 80186 CPU. The Multibus memory is accessible by both the 286/10 host and the 186/51 front-end communication boards. The 286/10 host board can access the Multibus memory over the local bus extension (LBX) and the 186/51 front-end board can access the Multibus memory via the Multibus.

The 186/51 board also includes 256 kilobytes of local memory, an 82586 local area network coprocessor, and an 82501 Ethernet serial interface. The 286/10 host board runs a traffic generation program which produces and consumes the traffic using communication services of the front-end board. The 186/51 front-end board runs the iNA960 software which is explained in detail in the next section.

The 82586 coprocessor and the 82501 together implement the IEEE 802.3 medium access control sublayer. The 82586 LAN coprocessor manages the processes of transmitting and receiving frames over a network. The 82586 receives and transmits frames from the local memory by direct memory access (DMA).

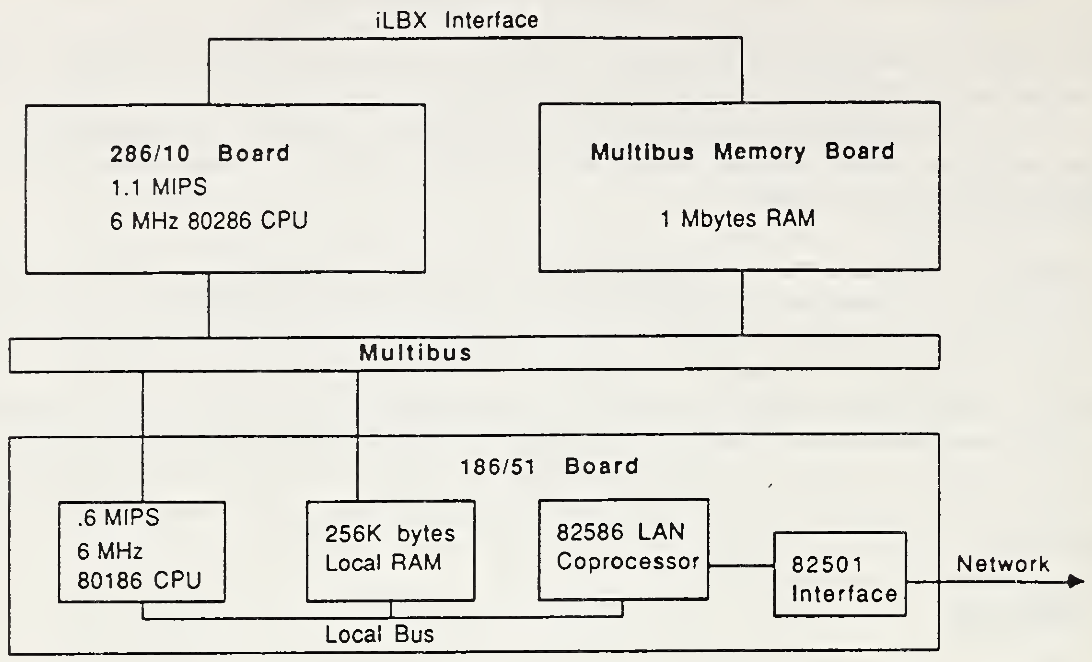

Figure 1. Simple Block Diagram of the System 310

Each system includes 3 processors running independently: the host board, the 186/51 front-end board, and the 82586 coprocessor. MIPS in Figure 1 is the acronym for "million instructions per second".

3.2 SOFTWARE OVERVIEW (iNA960 RELEASE 2.0)

The iNA960 Release 2.0 is the OSI Transport software package developed by Intel. In our validation experiments, the iNA960 Release 2.0 is configured with OSI Transport Class 4, Connectionless Network Protocol, LLC Type 1, Class 1 and the IEEE 802.3 medium access control. The iNA960 implementation offers both Transport Class 4 (TP 4) virtual circuit services (normal and expedited data) and Transport datagram (DG) services as shown in Figure 2. Transport datagram service is used to transfer data between users without setting up a connection and does not guarantee delivery.

The Network layer may be configured with one of the two protocol options: a Connectionless Network Protocol or a Null Network layer. If CLNP is chosen, nodes on interconnected subnets can be addressed. If a Null Network layer is chosen, only nodes on the same subnet may be addressed. The iNA960 offers external data link services (a direct user access to the Data Link layer).

The Network Management Facility (NMF) in the iNA960 software provides the network management services for the Transport layer, Network layer, and Data Link layer. The NMF enables the user to monitor the network operations either local or remote and tune the network parameters for better performance (e.g., set new values for the retransmission timer and maximum window size).

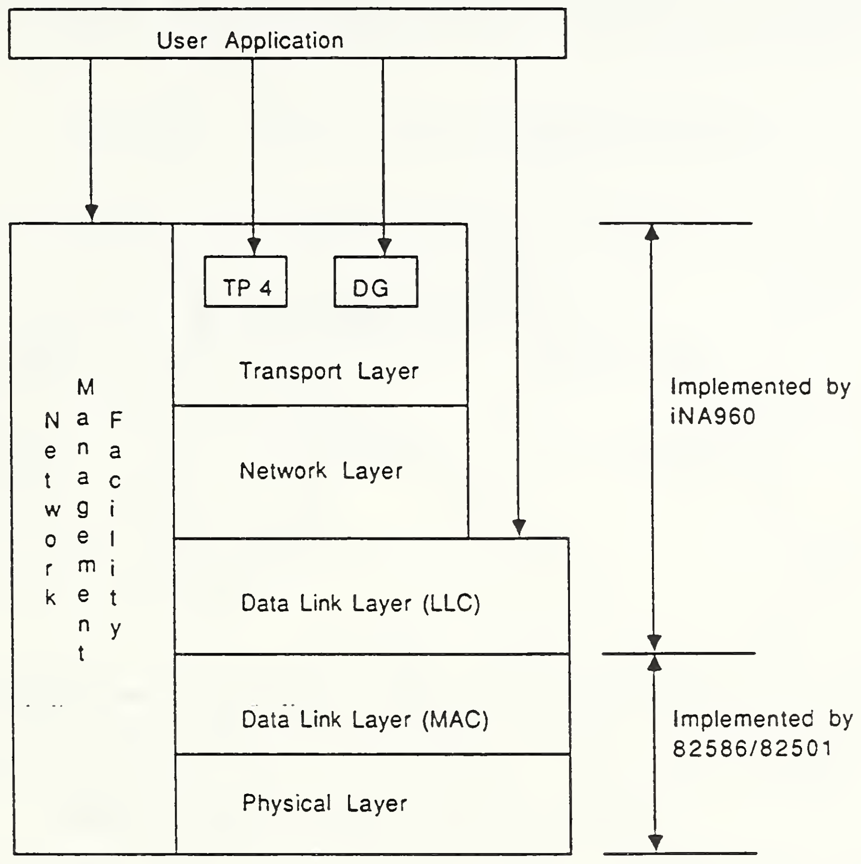

Figure 2. iNA960 Services

In our simulator, OSI Transport Class 4 services for normal data are modeled in detail. The model does not simulate: Transport services for expedited data; Transport datagram services; the external data link services; and the network management facility. For the validation experiments, the simulator is configured with OSI Transport Class 4 with CLNP and the CSMA/CD local area network.

3.2.1 iNA960 USER INTERFACE

A user process running on the 286/10 host interfaces with an iNA960 communications software running on the 186/51 front-end board via request blocks (RB). The RB contains information about the services being requested. When the user process requests the iNA960 to transmit or receive data, the RB contains a pointer to an associated transmit or receive data buffer. When the services are completed, the RB is returned to the user with a response code by the iNA960. The returned RB also contains a pointer to the user buffer.

The communications between the host board and the front-end board is done via Multibus Interprocessor Protocol (MIP) [16]. Figure 3 shows the one-way message transfer from the host to the communications (comm) board via MIP. The host MIP send task delivers the RB to the comm MIP receive task by writing the buffer into Multibus memory and then generating a channel attention interrupt at the comm board. The comm MIP receive task passes the buffer to the destination process on the comm board and then sends an acknowledgement to the host MIP receive task. The message transfer in the reverse direction from the comm board to the host board is done in the same way.

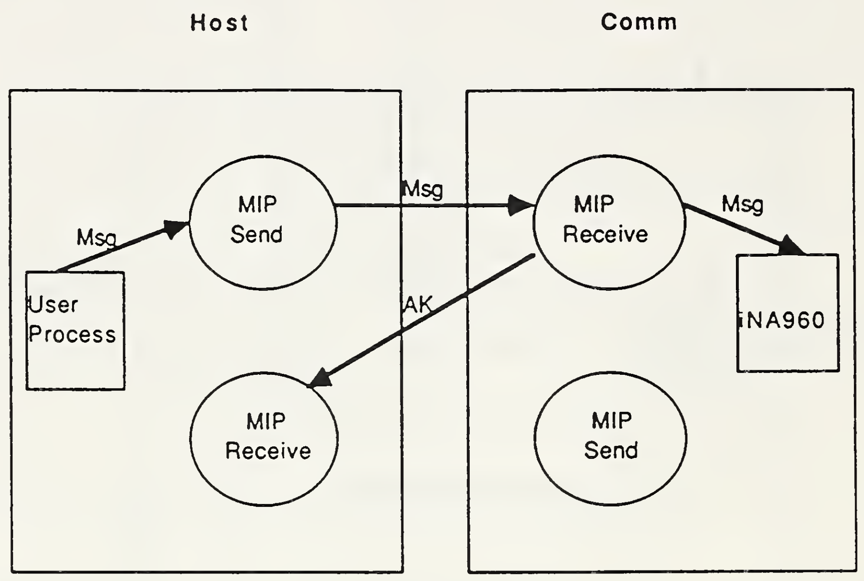

Figure 3. One-way Message Transfer via MIP

3.2.2 iNA960 BUFFER MANAGEMENT

The iNA960 implementation reserves a number of Data Link buffers in the 256 kilobytes local memory on the 186/51 board. The size of each transmit buffer is 1500 octets, large enough to store the memory image of a full IEEE 802.3 frame. Receive buffers are 256 octets each; these buffers are chained together to store the memory image of a received Data Link frame.

When the user sends iNA960 a request to transmit data, as explained in section 3.2.1, the request contains a pointer to an associated Multibus data buffer which contains the TSDU (or user message). The data in the user data buffer is copied by the iNA960 software into one or more Data Link transmit buffers on the 186/51 board. It is copied into more than one Data Link transmit buffer if Transport Class 4 segments the data into more than one TPDU. Then the data is sent out on the LAN by the 82586 LAN coprocessor. On the receiving side, the same set of events happens in reverse. The 82586 LAN coprocessor receives the data and stores it in Data Link receive buffers. The data in the Data Link receive buffers is copied into user receive data buffers previously posted by the host user process. When the buffers are full or when they contain a complete TSDU, they are returned to the user on the host. Note that the data is copied twice during the end-to-end transmission, from

Multibus memory to local memory at the transmit side and from local memory to Multibus memory at the receive side.

3.2.3 iNA960 CONFIGURATION PARAMETERS

The default values for the important configuration parameters in the iNA960 implementation for the validation are shown in Table 1.

Table 1. System Configuration Parameters

Number of data link transmit buffers	5
Size of data link transmit buffers (bytes)	1500
Number of receive packet descriptors	200
Number of data link receive buffers	255
Size of data link receive buffer (bytes)	256
Inactivity timeout (secs)	30
Abort timeout (secs)	240
Transport retransmission time (milliseconds)	500
Maximum Transport window	15
Minimum Transport credit	1
Maximum TPDU size (octets)	2048
Transport checksum	OFF

The credit returned in AKs in the iNA960 implementation is based on the amount of receive buffer space available at the Transport level. In addition, the credit may not be larger than the maximum Transport window parameter or less than the minimum Transport credit parameter. Since the minimum credit is one, each AK always contains permission to send at least one DT TPDU.

The actual TPDU size used is the minimum of the maximum TPDU size configuration parameter and the network packet size (minus lower layer headers). Unless otherwise noted, parameters are set to their default values in the validation experiments reported.

3.3 TRAFFIC GENERATION PROGRAM

A traffic generator running on the 286/10 host board was used to get the live results. The traffic generator generates the traffic in accordance with the following input variables: 1) Message delay (time between two successive user messages); 2) Duplex or simplex data flow; 3) User message size (TSDU size); 4) Total data sent for all connections; 5) Number of connections; 6) Number of user transmit (TX) and receive (RX) buffers for each connection. Other variables such as retransmission timer and maximum window size can be set on a connection-by-connection basis using network management services.

An external global clock system with 100 microseconds accuracy is connected to the 286/10 host board in all systems and makes it possible to get accurate measurements for all experiments, in particular accurate end-to-end delay measurements.

The traffic generation program provides throughput and delay measurements. The delay metric in this measurement program includes three measurements: request accept delay, request return delay, and one-way delay. Throughput is the total amount of user data transferred divided by the time required to send them. The time is measured from when the first TSDU is sent by the sending Transport user until the last TSDU is received by the receiving Transport user. Each delay measurement is described below.

An application task running in the host sends a request for communication services to the iNA960 program via a subroutine call. The subroutine call returns after the host sends the message (request block) containing the request for communication services to the comm board and receives an acknowledgement which indicates that the comm board has received the request (but has not necessarily processed it). Request accept delay is the time required for this subroutine call. When iNA960 has completed the requested service, the request block is returned to the user task. Request return delay is the delay from the time the request block is sent to the comm board by a user to the time the request block is returned to the user. One-way delay is the time required to send a message from a user on the host in the sending station to a user on the host in the receiving station.

4.0 SIMULATION MODEL

4.1 MODEL OVERVIEW

This section discusses the model which simulates the Intel 310 hardware system and the iNA960 communications software. The applications modeled include: simplex and duplex file transfers, database query, periodic status report, data entry, request response, and virtual terminal. Definitions of these applications are documented in the Experiment Plan [17]. For the validation experiments simplex and duplex file transfers, and periodic status report applications are used.

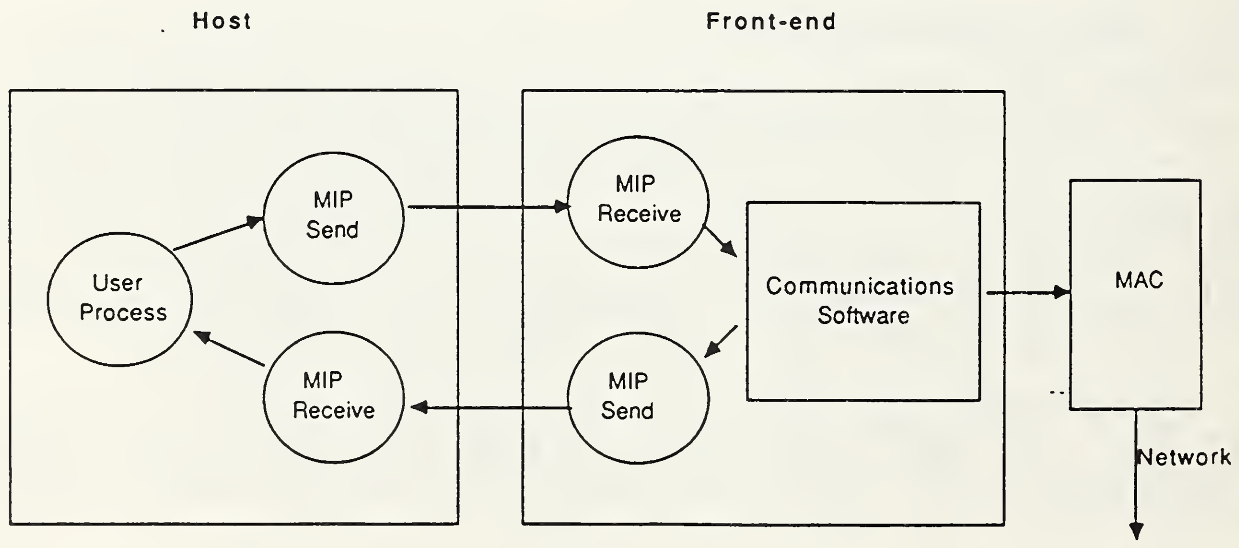

Figure 4. Main Tasks of the Simulated System

As shown in Figure 4, each simulated process maps directly to a process in the Intel 310 system. The simulated host processor contains the host MIP processes and the user processes which produce the traffic for Transport on the communications board. The user process controls generation of the traffic in accordance with the data input parameters provided by the experimenter for each simulation run. The host and the comm MIP processes, and the communications between the host board and the comm board are simulated in detail.

The simulated communications software provides the OSI Transport Class 4 services for normal data and performs in detail Transport Class 4 mechanisms for segmenting, flow control, recovery, reassembly, and error detection. As in the iNA960, Transport segments TSDUs which are bigger than network packets into multiple TPDUs where each TPDU fits into a network packet. The maximum size of the user data field of Data Link transmit buffer with CLNP is 1401 octets. The remaining octets are for the Transport and the lower layer headers (CLNP, LLC). The iNA960 buffer management is also modeled in detail. Retransmission timers are simulated. If an AK is not received before the retransmission timer expires, then retransmission of unacknowledged data occurs. The MAC networks modeled are IEEE 802.3 CSMA/CD and IEEE 802.4 token bus.

In our simulator, the connection establishment and release phases are not modeled. Connections are open at the beginning of the experiment and are not closed during the experiment. The time spent context switching between software processes on the communications board is also not simulated in the model. The time required for the context switching has been roughly estimated by counting the instructions in the code. The time estimated is .1 milliseconds (ms). Since the overhead is small compared to the variation seen in the measurements of the processing times in the iNA960 software (see Figure 5 below), the context switching is ignored in the model. Transport services for expedited data, Transport datagram services, the external Data Link services, and the network management facility are not modeled.

The model used for the validation runs on a Digital Equipment Corporation Micro-Vax II under the Ultrix operating system version 1.1. The model can be executed in one of two modes: interactive or batch. For the interactive mode, the experimenter can monitor performance parameters during the experiment. This is especially useful for analysis of unexpected results. For the batch mode, all experiments can be done in the background or overnight.

A typical model run for one of the throughput (simplex) validation experiments where 300 messages of 10000 bytes are transmitted from one station to a second station takes 304 CPU seconds.

4.2 PROCESSING TIME FOR EACH JOB

The simulation model contains a set of jobs such as posting a TX buffer and receiving a DT TPDU as does the iNA960 software. The processing time for each job in the iNA960 software has been measured by instrumentation of the source code using a hardware timer on the 186/51 board. The timer's granularity is .67 microseconds. Processing times for jobs on the host board have been estimated from the results of the live experiments. The estimation method for these jobs is explained in the following section.

4.2.1 MEASURED PROCESSING TIMES

The input parameters used for the measurement and the estimation experiments using the traffic generation program were as follows: 1) inter-message delay - 100 milliseconds, 2) TSDU size - 100 bytes, 3) total data sent - 10000 bytes, 4) number of connections - 1, 5) number of the user buffers - 1 TX buffer and 1 RX buffer.

The starting time and the ending time of each job were saved and 100 measurements were made. The minimum processing time, the maximum processing time, and the average processing time were computed when the experiment was completed.

Figure 5 shows the measured CPU time of each job obtained from the experiments. The average processing time for each job is used as input to the model. A brief description of the parameters is in Appendix A.

Figure 5. Measured CPU Time For Each Job

	Mininum (ms)	Maximum (ms)	Average (ms)
Comm MIP Send (send a message)	.921	1.271	.956
Comm MIP Send (time taken for the buffer to be acknowledged)	2.371	6.398	2.802
Comm MIP Receive (receive a message)	1.585	1.661	1.635
Comm MIP Receive (receive an AK)	.897	.927	.921
Post TX Buffer	1.361	1.703	1.378
Post RX Buffer	.861	.872	.865
Sent DT	5.499	5.854	5.575
Send DT Overhead	.388	.405	.395
Send DT Fail	.421	.636	.494
Send AK (includes overhead)	5.877	7.773	7.210
Receive DT	5.057	5.408	5.181
Receive AK	3.954	4.303	4.071
DL TX Interrupt	.096	.337	.329
DL RX Interrupt	.181	.193	.187

4.2.2 ESTIMATED PROCESSING TIMES

The "host MIP send" processing time was estimated in the following way. As explained in section 3.3, an application task interfaces to the iNA960 program via a subroutine call. The time measured across this subroutine call, which is "request accept delay", is 4.1 milliseconds. The time measured in the "comm MIP receive" task, to receive the message and acknowledge it, is 1.635 milliseconds. The difference between these two numbers is 2.47 milliseconds. Thus, 2.47 milliseconds should be the amount of processing time taken in the host by the "host MIP send" processing plus the software interface between the applications task and the "host MIP send" task. 1.257 milliseconds of this is attributed to "host MIP send" (send a message) and 1.211 milliseconds is attributed to "host MIP receive" (receive an AK). This is the same proportion as the .956 milliseconds that the comm board requires to send a message and the .921 milliseconds that the comm board requires to receive an AK.

The "host MIP receive" processing was also estimated from another measurement. The amount of time spent from transmission of a message in the "comm MIP send" task to acknowledgement of the message was measured at 2.8 milliseconds. If the amount of processing required for the comm board to receive an AK, .921 milliseconds, is subtracted from 2.8 milliseconds, the difference is 1.879 milliseconds. This is approximately the amount of processing required to receive a message and send an AK in the host board.

The "host recycle RX buffer" time, the time required in the traffic generation program to process a RX buffer and return it to the comm board, and the "host recycle TX buffer" time, the time required in the traffic generation program to process a TX buffer and return it to the comm board, are obtained from the traffic generation program. For the "host recycle RX buffer" time the traffic generation task receives an RX buffer from the iNA960 program on the comm board after a request for communication services has been processed. The traffic generation task recycles the RX buffer and returns it to the iNA960 program. The time measured in the total host recycle RX process is 4.49 milliseconds. When the traffic generation task returns the RX buffer for communication services to the iNA960 program, the interface (subroutine call) which is described in section 3.3 is called. The time required for the subroutine call, 4.1 milliseconds, is included in the total host recycle RX process. The "host recycle RX buffer" time is the difference between these two numbers, which is .39 milliseconds. The "host recycle TX buffer" time is estimated in the same way. The "host recycle TX buffer" time obtained from the experiment is .43 milliseconds. Of this, 0.21 milliseconds is attributed to the "host generate TX buffer" time and 0.22 milliseconds is attributed to the "host consume returned TX buffer" time.

In order to get the "copy byte Multibus (M) to local (L)" and the "copy byte L to M" times we used data obtained from a delay vs. TSDU size experiment. Parameters used for the experiment were as follows: inter-message delay - 100 milliseconds, number of user buffers - 1 TX buffer and 2 RX buffers, number of connections - 1, and TSDU size - 100 bytes, 500 bytes, 1000 bytes, and 1400 bytes. As shown in Figure 6 the best fit (least squares) line has the equation:

$$y = 3.84E - 03 * x + 17.29$$

The variable x represents TSDU size in bytes while y is the delay in milliseconds. The processing time that depends on the number of bytes, i.e., the time required for the "copy byte from M to L" on the sending station, the "copy byte from L to M" on the receiving station plus the transmission time should be approximately 3.84 microseconds. The maximum transmission speed for communication is 10 megabits per second and a byte is 8 bits. Therefore the minimum transmission time is .8 microseconds per byte. If the transmission time, 0.8 microseconds, is subtracted from 3.84 microseconds, the difference is 3.04 microseconds. This should be the amount of time for the "copy byte from M to L" and the "copy byte from L to M". Thus, assuming that the two copies take the same amount of time, it takes approximately 1.52 microseconds to do either one of the "copy byte L to M" or the "copy byte M to L". The estimated CPU time for jobs are shown in Table 2.

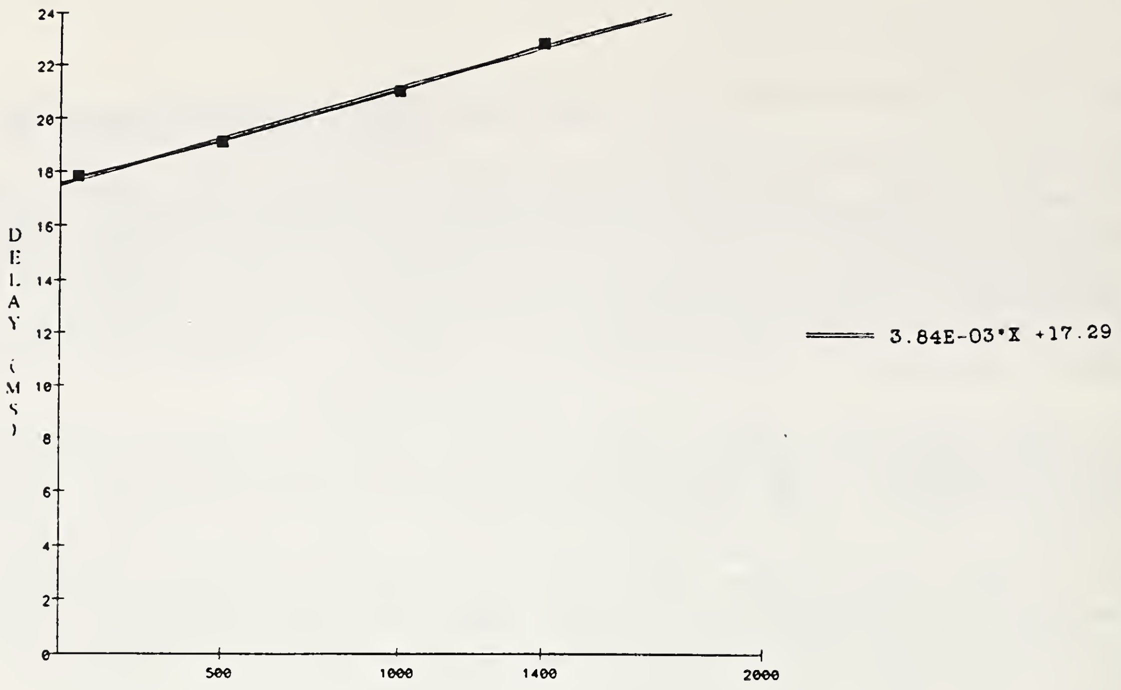

Figure 6 Graph Used For Calculation Of The Copy Byte Times

Table 2. Estimated CPU Time For Each Job

Host MIP send (send a message)	1.257(ms)
Host MIP receive (receive an AK)	1.211(ms)
Host MIP receive (receive a message)	1.879(ms)
Host recycle RX buffer	.39(ms)
Host generate TX buffer	.21(ms)
Host consume returned TX buffer	.22(ms)
Copy byte M to L	1.52(microsecs)
Copy byte L to M	1.52(microsecs)

4.3 MODEL INPUT FILES

There are three input files for the Simscript II.5 model: the system configuration file, the network description file, and the connection data file. The system configuration file provides the system configuration parameters and the processing time for each job. The characteristics for the network modeled are provided in the network description file. Each system is initialized in accordance with the system configuration file and the network description file. It is possible to have different system configuration files for different stations in order to simulate a network where stations are not all homogeneous, e.g., some stations are faster than others. The default values for the system configuration parameters in the model for the validation are the same as those in the iNA960 shown in Table 1 in section 3.2.3. Table 3 shows the characteristics for the CSMA/CD network modeled.

Table 3. Simulated CSMA/CD Network Characteristics

Link Speed	10 Mbps
Average Propagation Delay	2 microsecs
Slot Time	51.2 microsecs
Collision Overhead	0 microsecs
Jam Time	3.2 microsecs
Interpacket Delay	9.6 microsecs
Preamble (octets)	7 octets
LLC and MAC Header (octets)	24 octets
Bit Error Rate	0

(Note: The bit error rate for CSMA/DC networks is low enough that the probability of error in the simulator is set to 0).

The connection data file provides the connection data parameters on a per connection basis. The experimenter may provide as many stations and connections as desired. Each connection may choose one of the application types explained in section 4.1. The source and the destination stations of the connection are given by the experimenter. The experimenter also provides TSDU size distributions, TSDU arrival time distributions, and a probability for each type of message (or TSDU) on a connection, based on the application type. Parameters such as the maximum window, the minimum credit, and the size of buffers are specified for each connection.

4.4 MODEL OUTPUT FILE

The output file contains the performance statistics for each run. The model provides program execution statistics such as actual system elapsed time, CPU time used, and the simulated time used for a run. The following performance metrics are measured and provided for each run. For the validation experiments, throughput and delay measurements are only used.

4.4.1 THROUGHPUT

User throughput is defined as the total amount of user data transferred divided by the time required to send them. The time is measured from when the first TSDU is sent by the sending Transport user until the last TSDU is received by the receiving Transport user. In the validation experiments this measurement is of interest in both the simplex file transfer and duplex file transfer. User throughput is measured for each connection and for each direction of data flow on a connection. The sum of user throughput for all connections per Transport entity is also reported.

4.4.2 DELAY

One-way end-to-end delay is the time a TSDU takes to be transferred from a sending Transport user to a receiving Transport user. Average, minimum, maximum, and the standard deviation are reported on a per connection basis and each direction of data flow on a connection for each Transport entity. The average one-way delay measurement is used in the validation.

Two-way delay is measured from when the first byte of a request TSDU is transmitted by the initiating Transport user until the last byte of the corresponding response TSDU is received by the initiating Transport user. Average, minimum, maximum, and standard deviation are reported on a per connection basis and each direction of data flow on a connection for each Transport entity. The two-way delay is applicable to applications such as request response, database query, data entry, and virtual terminal.

4.4.3 TRANSPORT TRANSMIT AND RECEIVE DELAY

Transport transmit delay is measured from the time a TSDU is accepted by Transport from the user to the time the end of the TSDU is sent into the network. In the same way, Transport receive delay is the elapsed time between the time a TSDU is accepted by Transport from the network and the time the complete TSDU is sent to the user.
Average, minimum, maximum, and standard deviation are reported for each end station in each connection. The measure of the delay metric includes Transport processing time and internal queueing delay.

4.4.4 PROTOCOL EFFICIENCY

The protocol efficiency is defined as the ratio of the Transport user data sent to the total data sent (including AKs and headers) by the Transport entity. The measurement is reported for each end station in each connection.

4.4.5 RETRANSMISSIONS

The measures of retransmission include the total number of retransmissions, the average number of retransmissions per DT TPDU, and the distribution of retransmissions. The measurement is reported for each end station in each connection.

4.4.6 RESOURCE UTILIZATION

The model reports the utilization of resources. The resources in the model include the transmission channel, communications CPU, user buffers, and Data Link buffers.

The channel utilization is the total throughput sent on the link (including both user data and overhead) divided by the link speed.

The measurement of communications CPU utilization is reported on a per Transport entity basis and also reported on a per connection basis supported by the Transport entity. The measurement of CPU utilization for each connection is broken up into the CPU utilization for each process in the communications system.

The model provides the utilization of the RX and TX buffers for both the user and the Data Link. The average amount of memory space used is measured for the user RX buffers. The average fraction of buffers used is measured for the user TX buffers and the Data Link RX and TX buffers. The measurements for both the user and the Data Link memory are reported on a per connection basis.

5.0 VALIDATION EXPERIMENTS

A number of different types of experiments were conducted for both the live measurements and the simulation measurements using the same experimental parameters. The experimental environment for both measurements was Transport Class 4 with CLNP configured in and the CSMA/CD LAN network. The network background traffic was assumed to be zero. Two metrics were measured for the comparison: one-way delay and throughput.

The experiments divide into three sets: 1) one-way delay, 2) throughput, and 3) multi-application. Each of these is explained in detail below.

5.1 <u>ONE-WAY DELAY</u>

Experiment: One-way Delay vs. TSDU Size

Parameters:

TSDU interarrival time (constant)	- 100 milliseconds
Duplex or simplex data flow	- Simplex
TSDU size (constant distribution)	- varied between 100 and 17000 octets
Total data sent	- 10 megabyte (live) 100 TSDUs (simulation)
Number of connections	- 1
Number of user buffers	- 1 TX and 2 RX buffers
Maximum window	- 15
Minimum credit	- 1

Live and Simulation Results:

One-way delay was measured from the initiating user process to responding user process for both the live and the simulation experiments. Figure 7 presents the one-way delay measurements as a function of the message size (TSDU size) for both the live and the simulation experiments.

Remember that if a TSDU is bigger than the size of the user data field of the network packet, then the TSDU is segmented into one or more TPDUs. Since the size of the user data field of Data Link transmit buffer with CLNP is 1401 octets, 1402 octets or more of Transport user data will be segmented into two or more TPDUs.

The curves in Figure 7 show the same shape. As the message size increases the delay increases. Of particular interest are those user message sizes at the boundary where segmentation into an additional TPDU is required; those messages having (1401 * k) + 1 octets, where k = 1, 2, 3, .. e.g., 1402, 2803, 4204, exhibit a sharp increase in delay. It is reasonable to infer from these results that the time taken to segment the data is the cause of the sharp jump. The delay curves are higher in the simulation measurement.

Figure 8 shows two curves for the DT-to-AK ratio as a function of the TSDU size for both the live measurement and the simulation measurement. As can be seen from Figure 8, not every DT TPDU provides the return of an AK TPDU. The ratio increases as the message size increases. For the TSDU size of 16813 octets, the DT to AK ratio was 6.5:1 for the live measurement and the DT to AK ratio was 2.2:1 for the simulation measurement. Thus AKs are sent more often for the simulation experiment. This is probably the cause of the higher delay of the simulation results. Our hypothesis for the higher DT to AK ratio in the live experiments as opposed to the simulation experiments follows.

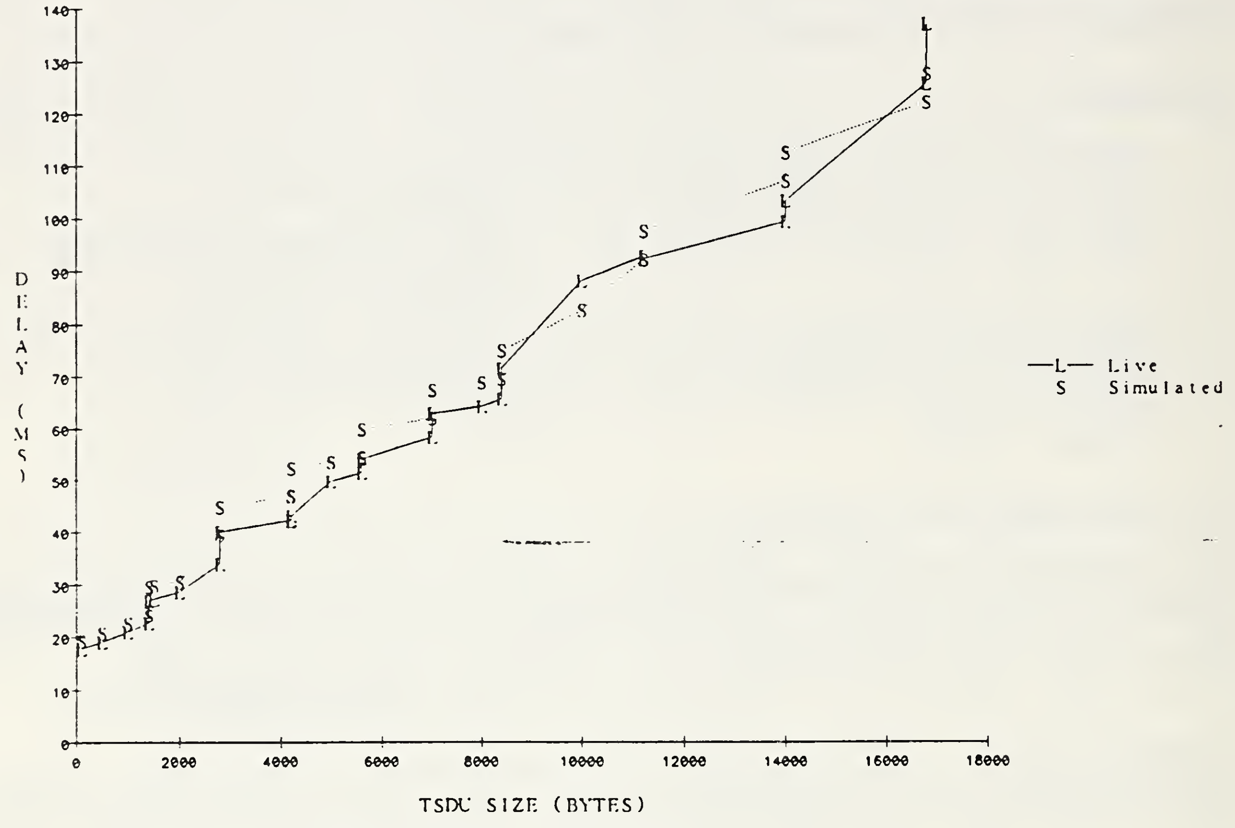

Figure 7 DELAY VS. TSDU SIZE

It is clear from the results of the live experiments that at the destination the "send AK" task is a lower priority than the "receive DT" task. Otherwise every DT TPDU would be acknowledged. If tasks of different priorities, such as "receive DT" and "send AK", are being scheduled then the ratio of DTs received to AKs sent depends on a number of factors: how often DTs are received, the processing time of "receive DT" and the processing time of "send AK". For example, if more DTs are received in a given amount of time, the processing times being unchanged, then fewer AKs will be sent and the DT to AK ratio will rise. Or if the arrival rate of DTs is unchanged but the processing time of AKs is increased, then fewer AKs will be sent and the DT to AK ratio will be higher. This is true for preemptive or nonpreemptive scheduling.

The arrival rate of DTs at the destination node is affected by the priority and processing time of the "send DT" task at the origination node. The priority and processing times of other tasks at both the destination and origination node may also have an effect on the DT to AK ratio.

The simulation model uses constant processing times for each task. However, if figure 5 is examined it is seen that in the live model, there is a variation in processing times. The biggest variation is in "send AK" where the minimum is

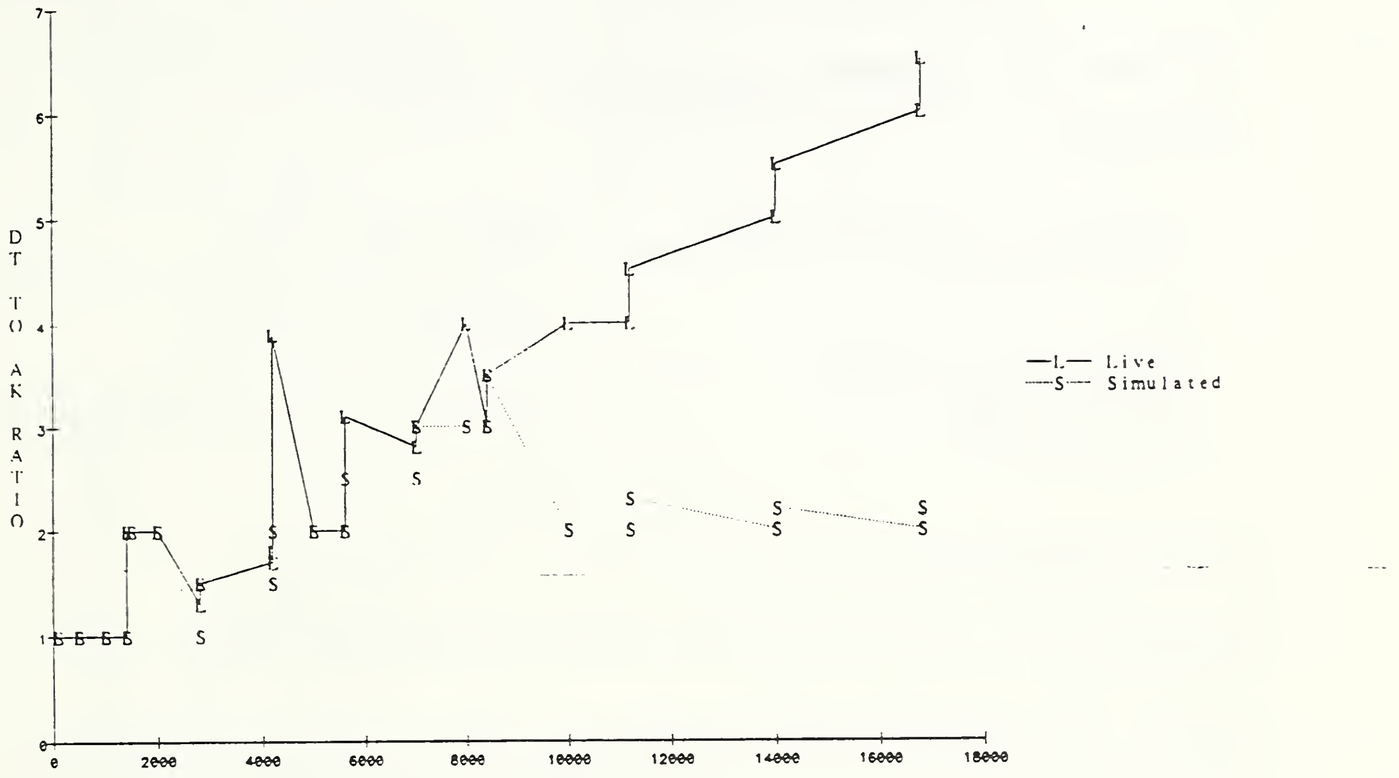

Figure 8 DT TO AK RATIO FOR ONE-WAY DELAY

5.877 ms, the maximum is 7.773 ms and the average, which is used in the model, is 7.210 ms. In addition the experiment which was used to obtain the processing times is described in section 4.2.1. It is possible that in other experimental situations, the processing times will be different for some tasks.

Undoubtedly, the reason for the different processing DT to AK ratios in the live experiments and simulation experiments is the inaccuracy of the processing times in the model. The important question is whether,. with these inaccuracies, the model is usable for our purposes. This will be addressed in Section 6, Conclusions.

5.2 <u>THROUGHPUT</u>

For the throughput measurements, bulk data are continuously transferred memory-to-memory from a transport user on one system to a transport user on the other system in one direction (simplex) and both directions (duplex). For both the live and the simulation experiments, all receive buffers are posted before the experiment starts. On the transmitting station, the user process sends all transmit buffers containing user data to the communications board before waiting for any acknowledged buffer to be returned at the beginning of the experiment. As soon as all the data in a user buffer are acknowledged, the user buffer is returned to the user and recycled for the next transmission of data. On the receiving station, a receive buffer is returned to the user when a buffer is full or the end of TSDU is received at the Transport. The user process then reposts the receive buffer.

Experiment #1: Throughput vs. TSDU Size

Parameters:

Duplex or simplex data flow	- Simplex and duplex
TSDU size (constant distribution)	- varied between 100 and 25000 octets
Total data sent	- 10 megabyte (live) 300 TSDUs (simulation)
Number of connections	- 1
Number of user buffers	- 10 TX buffers and 10 RX buffers
Maximum window	- 15
Minimum credit	- 1

Live and Simulation Results:

This experiment is divided into two subsets: simplex and duplex file transfers.

Figures 9 and 10 show simplex and duplex throughput versus message size for both the live and the simulation measurements. Both graphs exhibit the same shape. Throughput increases rapidly as the user messages increase up to 5600 octets for both the live and the simulation measurements. Beyond this figure the rate of throughput increases more slowly for both the live and the simulation measurements. As pointed out in the delay case, throughput drops down sharply at the point where the user messages are segmented into an additional TPDU, resulting in additional processing for each user message.

The DT-to-AK ratio of the simplex experiment for both the live and the simulation are shown in Figure 11. The DT-to-AK ratio for both the live and the simulation are almost the same for message sizes up to 2000 octets. As shown in Figure 11, the difference in DT-to-AK ratio between the live and the simulation measurements increases as the user messages increase. For the user message size of 16813 octets, the DT to AK ratio was 5.2:1 for the live measurement and the DT to AK ratio was 2.8:1 for the simulation measurement. More AKs are sent for the simulation as in the delay experiment.

It seems probable that the time taken to send additional AKs is the cause of the lower throughput of the simulation measurements compared to the live measurements. This may be the reason for the lower throughput of the simulation measurement for all remaining experiments done for the validation.

Experiment #2: Throughput vs. Number of Receive Buffers

Parameters:

Duplex or simplex data flow	- Simplex
TSDU size	- 10000 bytes
Total data sent	- 1 megabyte (live) 300 TSDUs (simulation)
Number of connections	- 1
Number of user buffers	- TX buffers: 10 and RX buffers: 1, 2, 3, 4, 6, 8 and 10
Maximum window	- 15
Minimum credit	- 1

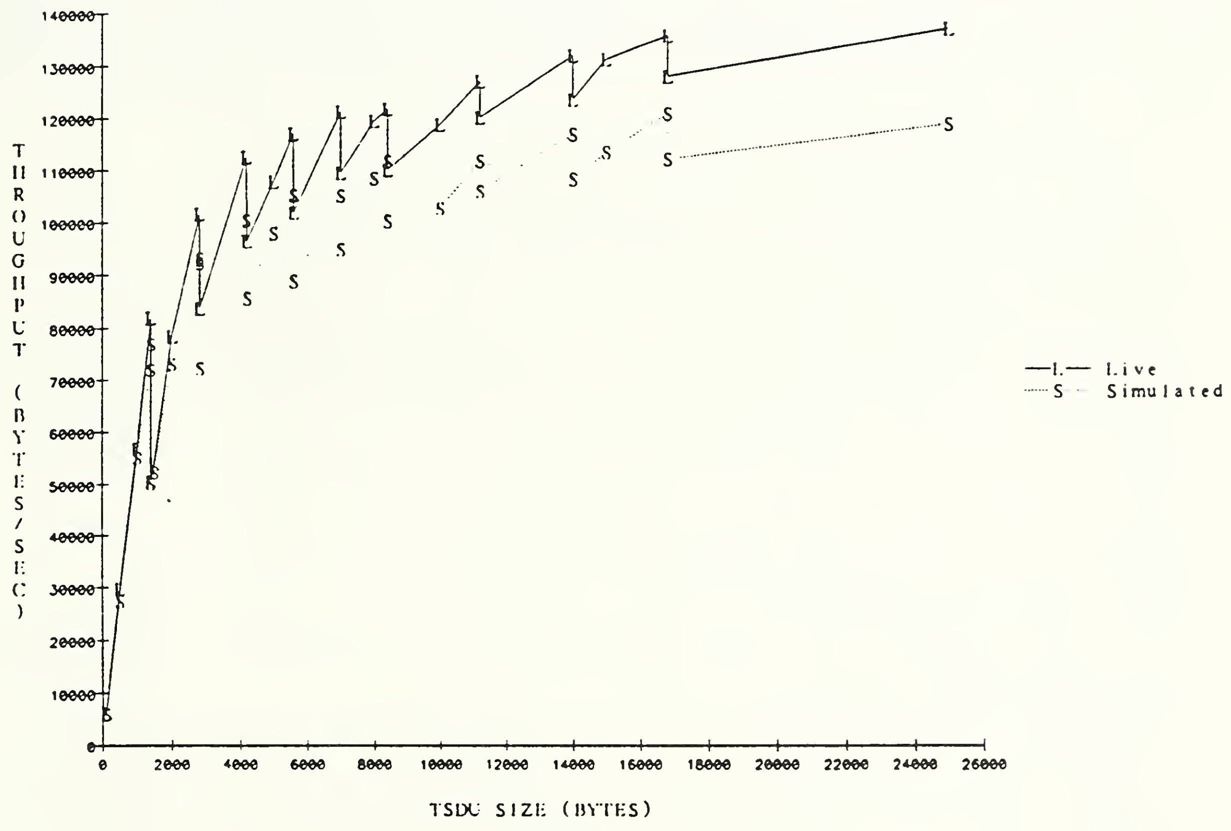

Figure 9 THROUGHPUT (SIMPLEX) VS. TSDU SIZE

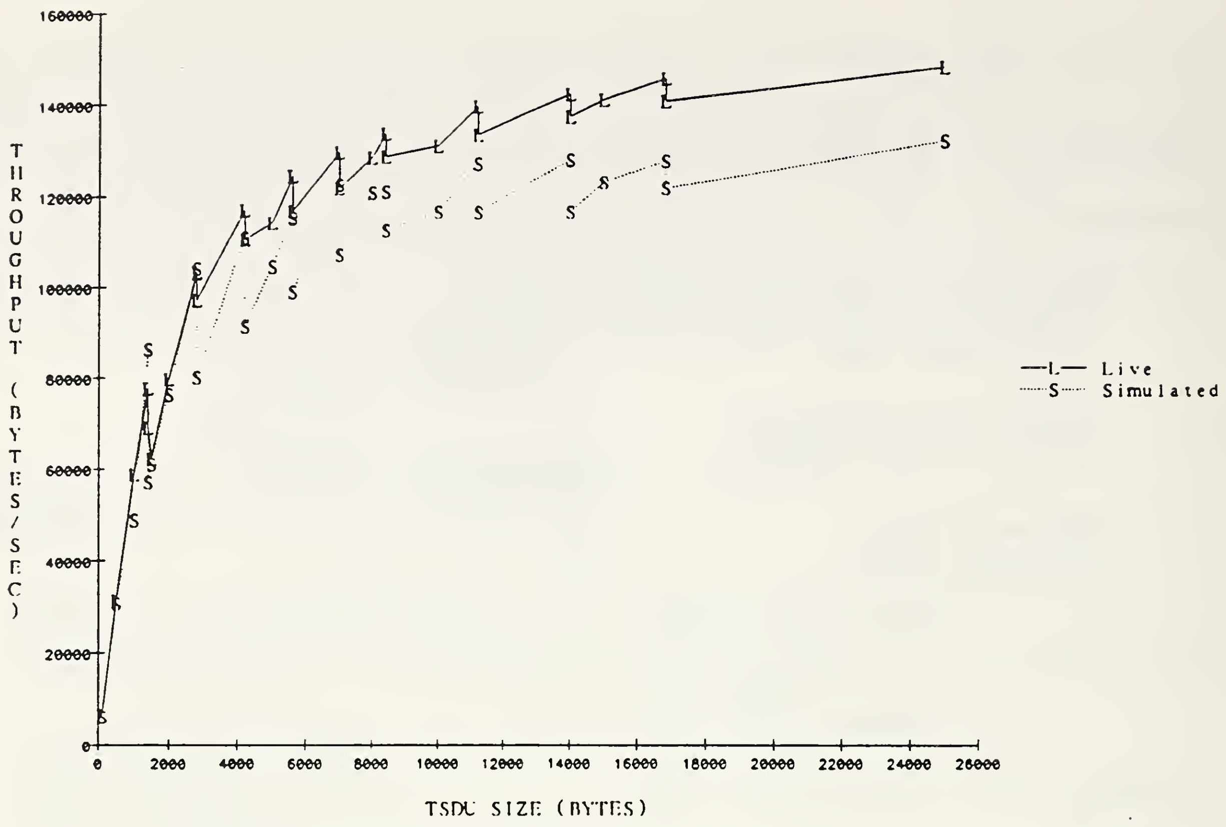

Figure 10 THROUGHPUT (DUPLEX) VS. TSDU SIZE

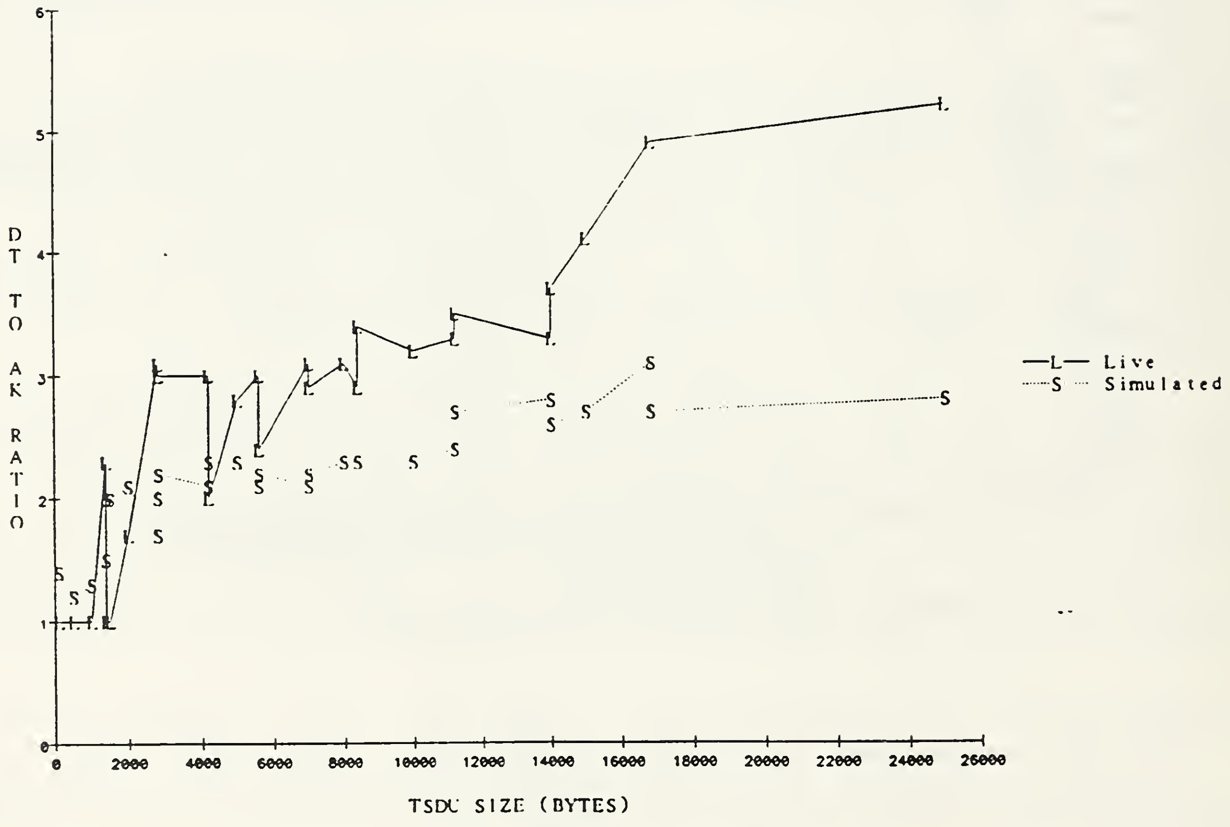

Figure 11 DT TO AK RATIO FOR SIMPLEX THROUGHPUT

Live and Simulation Results:

For the experiment where the number of receive buffers was varied, the number of transmit buffers was held constant at 10 for the live and the simulation measurements. Figure 12 shows two curves for the throughput as a function of the number of receive buffers.

For the live measurements, when two user receive buffers are made available, measured throughput is increased by 18k bytes per second. There is no change in throughput when three or more user receive buffers are made available. Thus, at least three receive buffers are necessary for the maximum throughput.

For the simulation measurement, when two user receive buffers are made available, measured throughput is increased by 20k bytes per second. There is no further change in throughput when two or more user receive buffers are made available. Thus, at least two receive buffers are necessary for the maximum throughput since the receive buffer is always available at the front-end level. When there is only one receive buffer, the front-end board on the receiving station sends a receive buffer filled with the received data back to the host board and waits for it to be reposted for data to be received. This causes the transmitting Transport to run out of credit, resulting in reduced throughput. The maximum throughput for live measurements is about 119K bytes/sec and the maximum throughput for simulation measurements is about 103K bytes/sec.

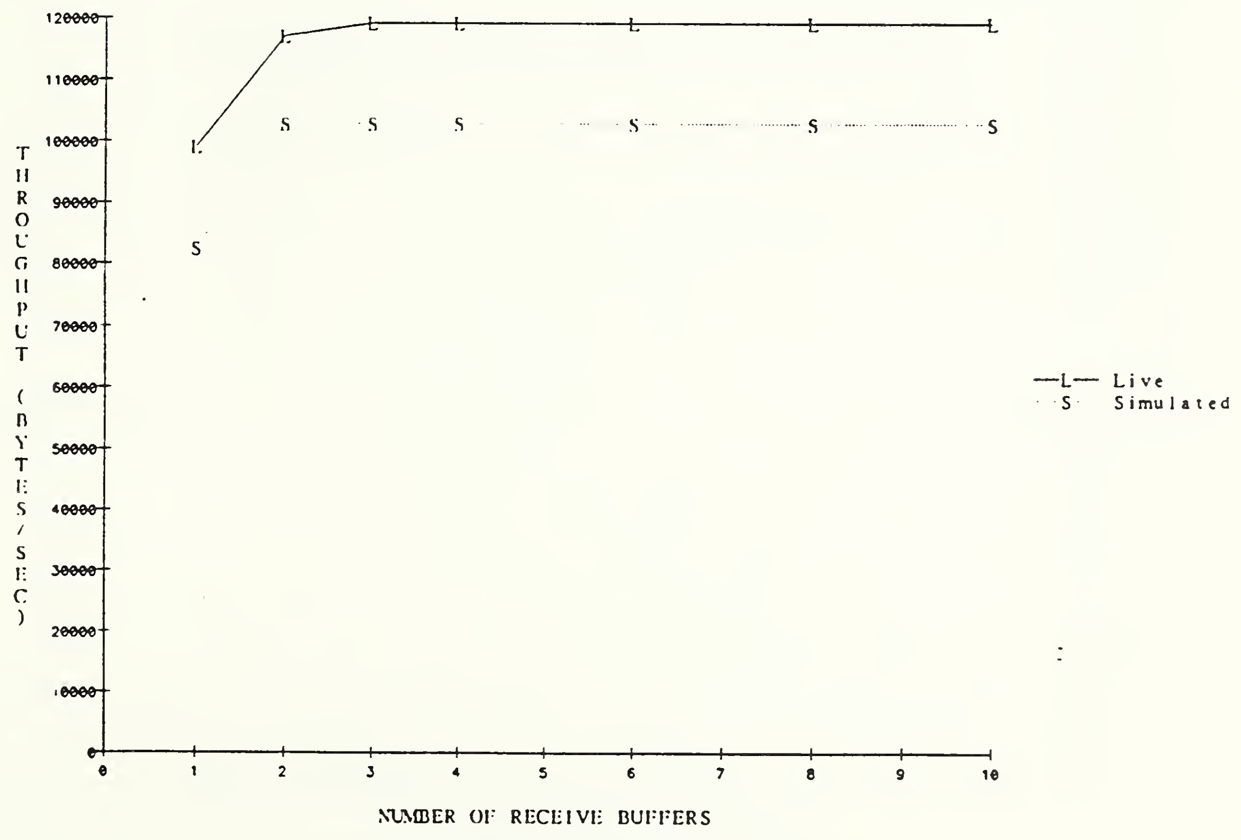

Figure 12 THROUGHPUT VS. NUMBER OF RECEIVE BUFFERS

Experiment #3: Throughput vs. Number of Transmit Buffers

Parameters:

Duplex or simplex data flow	- Simplex
TSDU size	- 10000 bytes
Total data sent	- 1 megabyte (live) 300 TSDUs (simulation)
Number of connections	- 1
Number of user buffers	- RX buffers: 10 and TX buffers: 1,2,3,4,6,8, and 10
Maximum window	- 15
Minimum credit	- 1

Live and Simulation Results:

Measurements for throughput versus number of transmit buffers are graphed in Figure 13. The number of receive buffers was held constant at 10 while the number of transmit buffers was varied.

As shown in Figure 13, the two curves for the live and the simulation measurements exhibit the same shape, and the measured throughput for both remains about the same (119k bytes/sec for the live and 103k bytes/sec for the simulation) for two or more user transmit buffers. Thus, the optimal number of transmit buffers is two for both measurements. When there is only one TX buffer

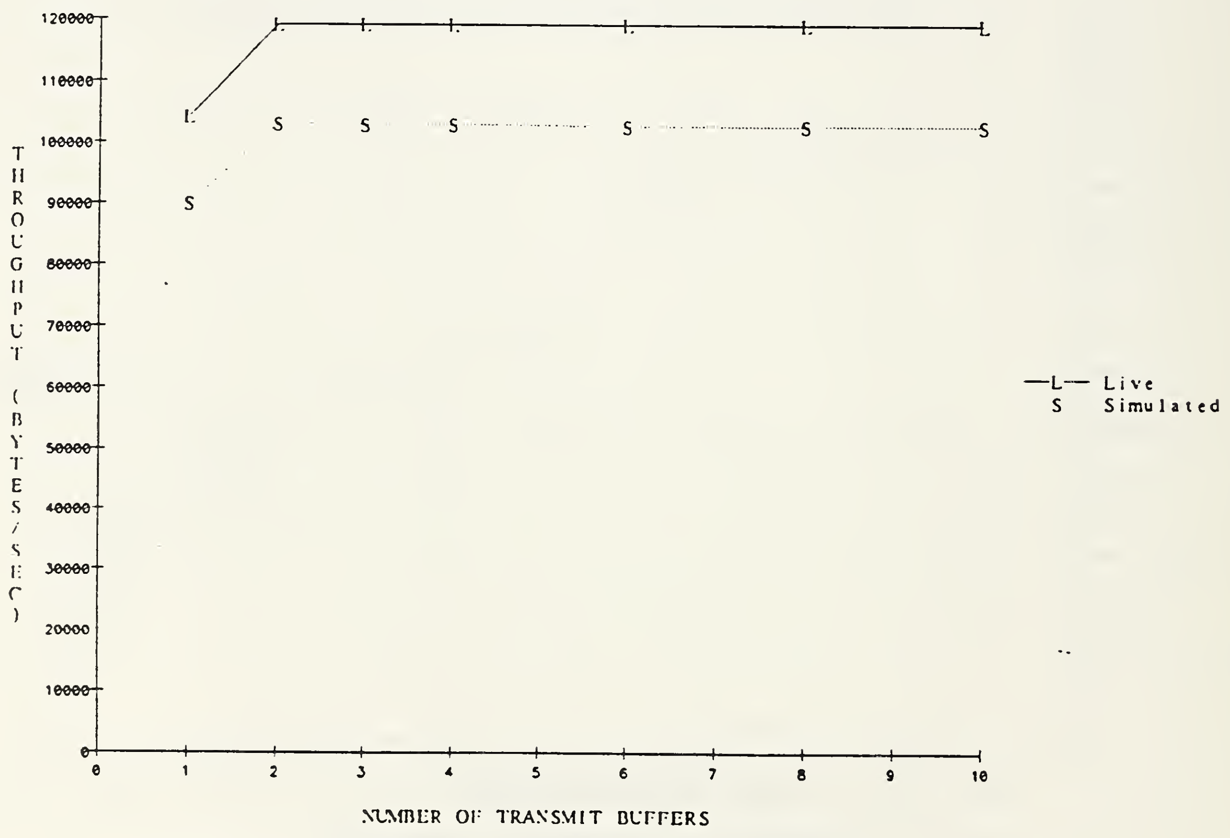

Figure 13 THROUGHPUT VS. NUMBER OF TRANSMIT BUFFERS

available, the front-end board on the sending station sends a TX buffer back to the user as soon as the data in the buffer is acknowledged and waits for it to be reposted for the next data to be sent. This results in lower throughput since data to be transmitted is not available all the time.

Experiment #4: Throughput vs. Maximum Window

Parameters:

Duplex or simplex data flow	- Simplex
TSDU size	- 10000 bytes
Total data sent	- 10 megabyte (live) 300 TSDUs (simulation)
Number of connections	- 1
Number of user buffers	- 10 RX and 10 TX buffers
Maximum window	- 1,2,3,4,5,6,8,12 and 15
Minimum credit	- 1

Live and Simulation Results:

Throughput as a function of maximum window is graphed in Figure 14. For the live measurement, throughput increases sharply as the maximum number of window increases up to 4. Above this figure throughput levels off and grows much more slowly. For the simulation measurement, throughput increases sharply up to a maximum window of 3 and more slowly after that.

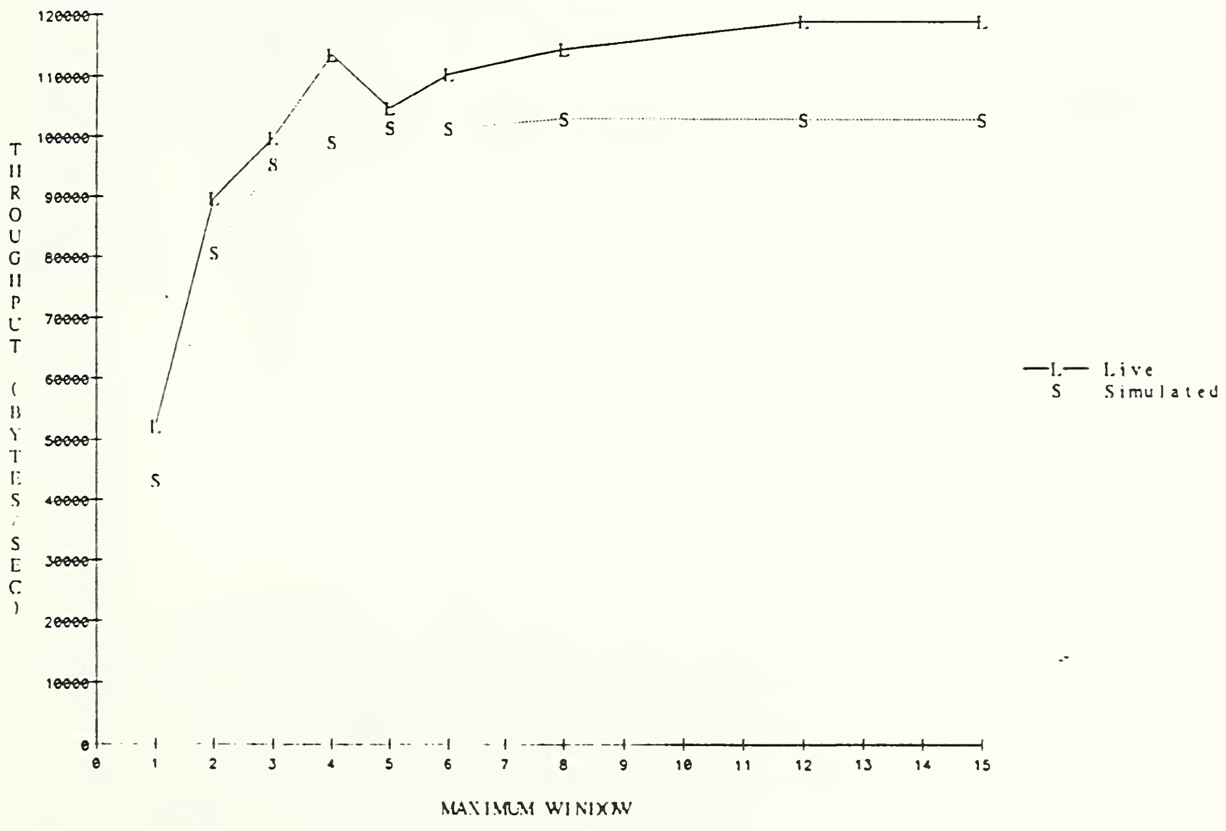

Figure 14 THROUGHPUT VS. MAXIMUM WINDOW

Maximum throughput was achieved when a maximum window was set equal to 12 for both the live and the simulation experiments. Since an AK is not necessarily returned for each DT TPDU, as seen in Figures 8 and 11, the maximum window size has to be large enough to allow several DTs to be received before an AK is sent, to decrease AK processing and thereby increase throughput.

Experiment #5: Throughput vs. Retransmission Timer

Parameters:

Duplex or simplex data flow	- Simplex
TSDU size	- 10000 bytes
Total data sent	- 10 megabyte (live) 100 TSDUs (simulation)
Number of connections	- 1
Number of user buffers	- 10 TX and 10 RX buffers
Retransmission timer	- 50,100,250 and 500 milliseconds
Maximum window	- 15
Minimum credit	- 1

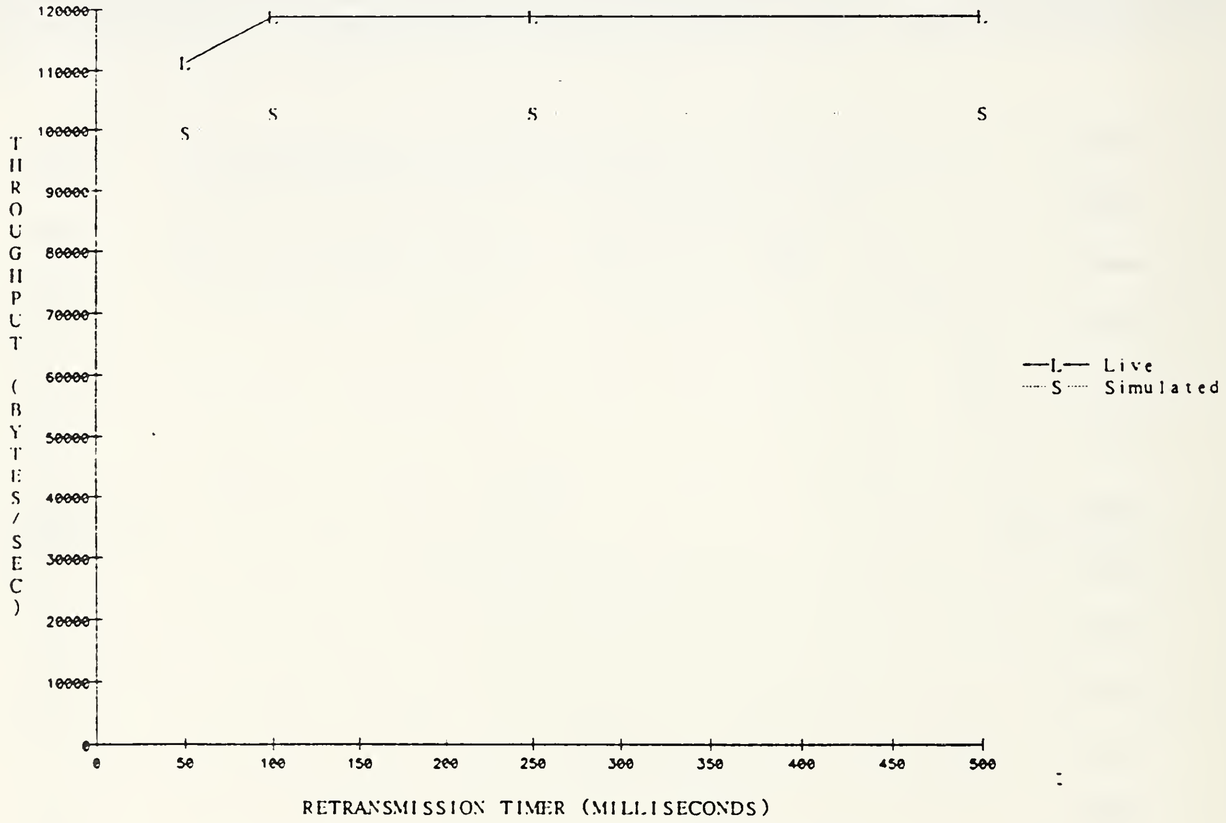

Figure 15 THROUGHPUT VS. RETRANSMISSION TIMER

Experiment #6: Throughput vs. Number of Connections

Live and Simulation Results:

Figure 15 shows the throughput versus retransmission timer. For both measurements, throughput remains the same independent of the timer values of 100 milliseconds or above, because no retransmission occurs. When the retransmission timer is set equal to 50 milliseconds, the retransmission timeouts occur since the retransmission timeout interval is too short. If no AK is received and the retransmission timer expires, all unacknowledged data are retransmitted, leading to reduced throughput for both the live and the simulation measurements.

Parameters:

Duplex or simplex data flow	- Simplex
TSDU size	- 10000 bytes
Total data sent	- 10 megabyte (live) 100 TSDUs (simulation)
Number of connections	- 1, 2, 3, 4, 5, 6, 8 and 10
Number of user buffers for each connection	- 10 TX and 10 RX buffers
Maximum window	- 15
Minimum credit	- 1

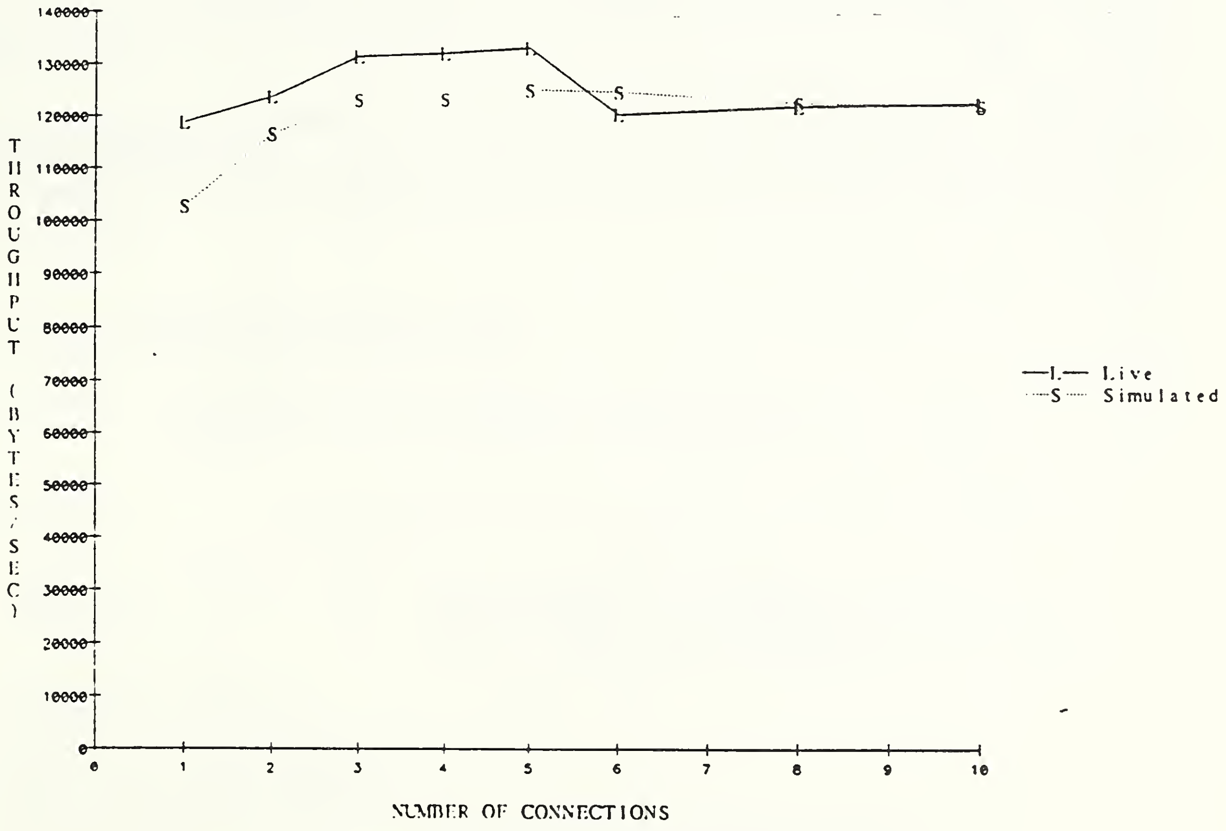

Figure 16 THROUGHPUT VS. NUMBER OF CONNECTIONS

Figure 16 presents throughput versus number of connections for both the live measurement and the simulation measurement. For the live measurement, throughput measured increases sharply up to 3 connections and more slowly up to 5 connections. Throughput drops sharply after which it increases slowly. The optimal number is 5. This suggests that the amount of overhead increases in maintaining multiple connections more than 5.

For the simulation measurement, throughput measured increases sharply up to 3 connections and more slowly up to 5 connections. For 6 or more connections, throughput drops slightly. The optimal number is 5 which is the same as the live measurement.

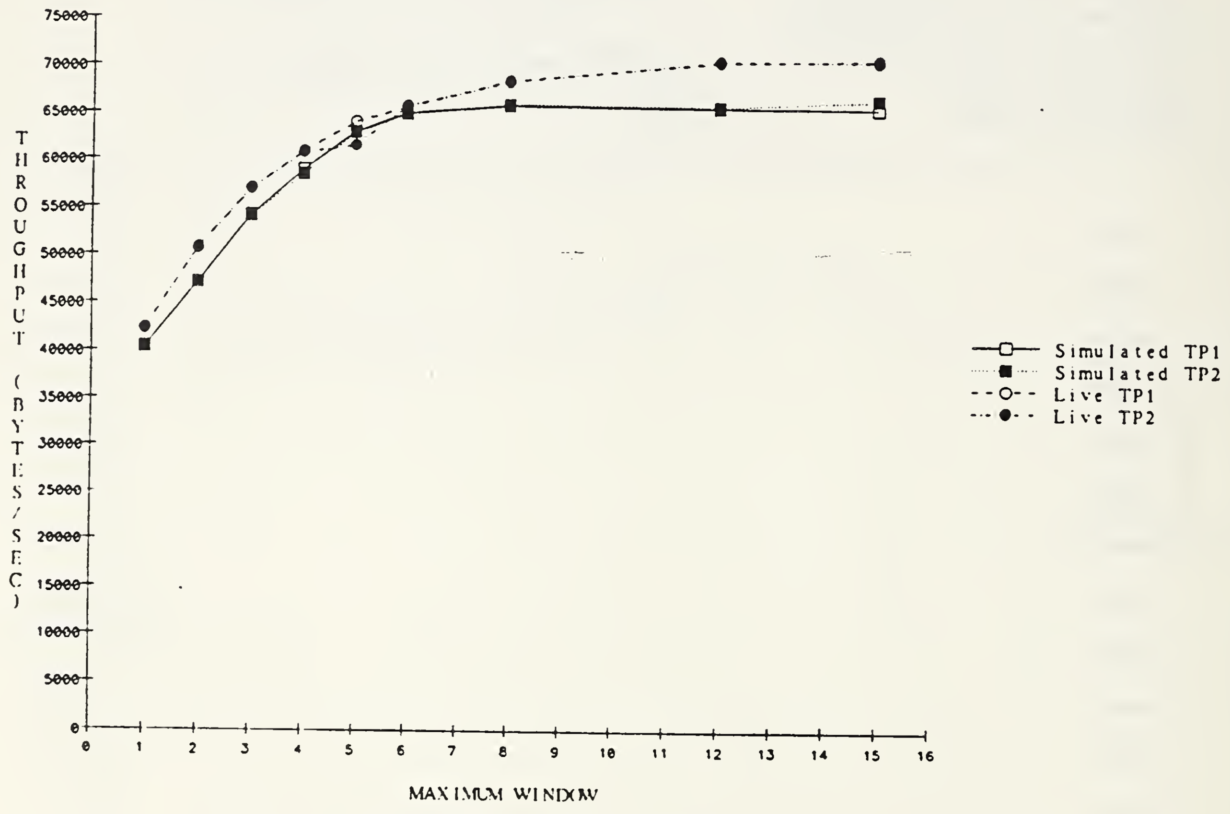

Figure 17 THROUGHPUT VS. MAXIMUM WINDOW
TWO STATIONS TRANSMIT TO ONE STATION

Experiment #7: Throughput vs. Maximum Window (Two Stations Transmit to One Station)

Live and Simulation Results:

Parameters:

Duplex or simplex data flow	- Simplex
TSDU size	- 10000 bytes
Total data sent	- 10 megabyte (live) 100 TSDUs (simulation)
Number of stations	- 3
Number of connections	- 2
Number of user buffers for each connection	- 10 TX and 10 RX buffers
Maximum window	- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, and 15
Minimum credit	- 1

Live and Simulation Results:

Two senders are continuously sending memory to memory bulk data to one receiver. Figure 17 shows throughput of Transport 1 and Transport 2 for both the live and the simulation experiments while the maximum window is varied. As the maximum window increases, throughput of both senders increases respectively toward 66K bytes for the simulation and 70K bytes for the live measurements.

For the live measurement, the throughputs seen at Transport 1 and Transport 2 seem to be fair since the two connections have achieved almost the same throughput. The same fairness is also observed for the simulation experiments.

5.3 Multi-Application Experiments

Experiment: Two Connections with Two Applications (Simplex File Transfer and Periodic Status Report)

Parameters:

Duplex or simplex data flow	- Simplex
TSDU size	- 10000 bytes(simplex file transfer) 100 bytes(periodic status report)
Total data sent	- 100 TSDUs
Number of stations	- 2
Number of connections	- 2
Number of user buffers for each connection	- 10 TX and 10 RX buffers
Maximum window	- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 and 15
Minimum credit	- 1

Live and Simulation Results:

The experiments involve a pair of applications between two stations. For the first application, the simplex file transfer generates bulk data traffic while in the second application, a periodic status report submits messages at a constant rate of 100 milliseconds. The data flow for both connections is in the same direction.

One-way delay vs. maximum window for both the live and the simulation experiments is shown in Figure 18. For the simulation experiments, one-way delay increases sharply as the maximum window increases up to 5. After this point one-way delay levels off and grows sharply to the maximum window of 7. One-way delay remains the same for the maximum window of 8 and more. For the live experiments, one-way delay increases sharply as the maximum window size increases up to 7 and more slowly after that.

Figure 19 shows the throughput vs. maximum window for both the live and the simulation measurements. The two curves have the same shape. Throughput increases sharply as the maximum window size increases up to 5 for both measurements. Above this figure throughput increases more slowly for the live measurement. For the simulation measurement, throughput remains the same above window size 5.

6.0 Conclusions

No model, whether a mathematical model or a simulation, is an exact duplicate of the system being modeled. The model will not produce the same outputs as the real system in all situations. The model used in this study exhibited throughput and delay differences of 6% to 20% compared to the real system. Thus, the model would not be a good predictor of the performance of a real system.

What the transport simulation model does a good job of predicting is which of two scheduling algorithms for implementing QOS at the transport layer would work better in a real local area network. This is illustrated by the fact that the output curves of the live system and the simulator follow one another. It is, therefore, felt that the simulator would predict relative performance of QOS algorithms. If it accomplishes this, then for the purposes of this study it is considered a successful simulator.

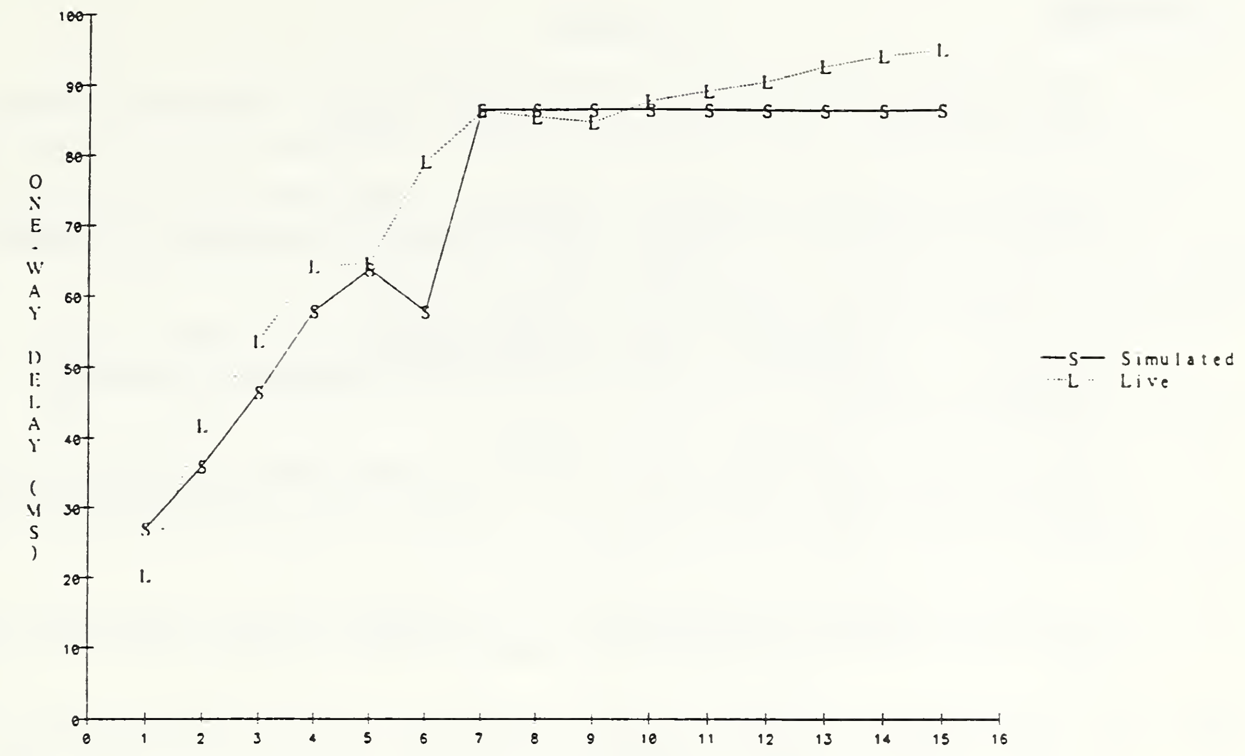

Figure 18 ONE-WAY DELAY VS. MAXIMUM WINDOW
(TWO TRAFFIC APPLICATIONS)

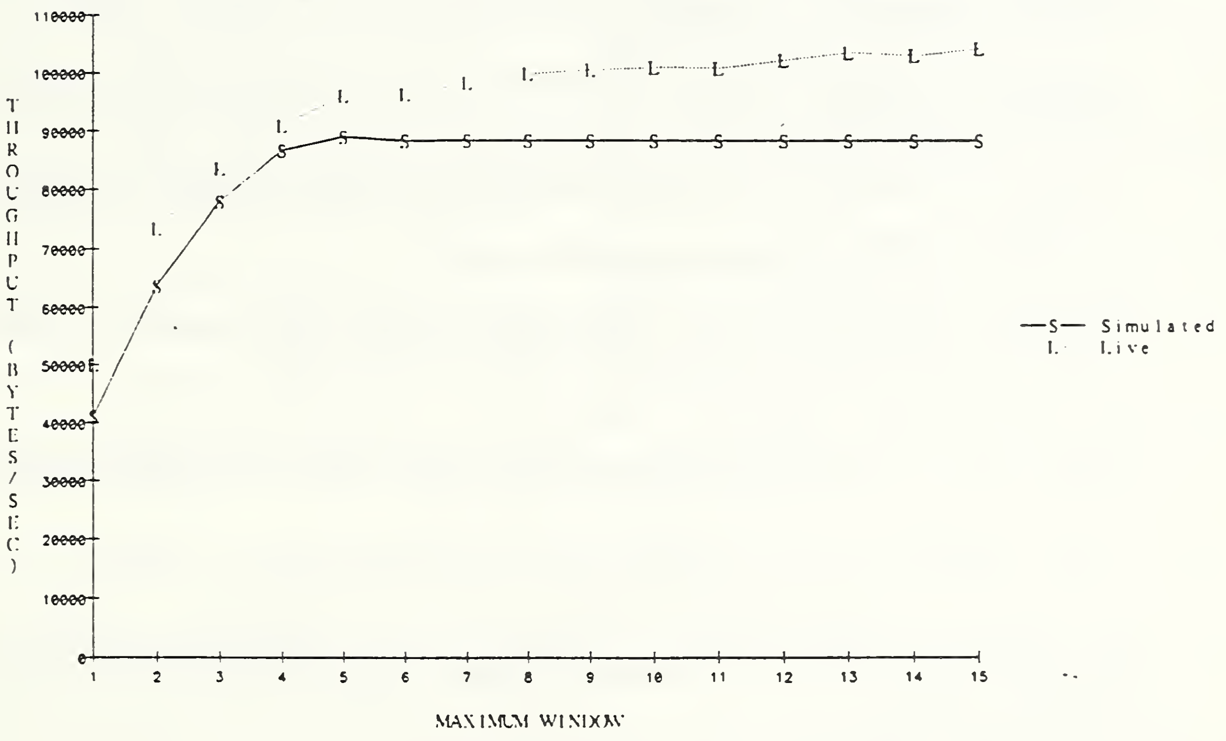

Figure 19 THROUGHPUT VS. MAXIMUM WINDOW
(TWO TRAFFIC APPLICATIONS)

REFERENCES

[1] International Organization for Standardization, Transport Protocol Specification, ISO/TC 97/SC 16, IS 8073, 1986.

[2] K. Mills, M. Wheatley, and R. Colella, "Simulation of an International Standard Transport Protocol," 1985 CMG XVI Proceedings, December, 1985.

[3] R. Colella, R. Aronoff, and K. Mills, "Performance Improvements for ISO Transport," Proceedings: Ninth Data Communications Symposium, September, 1985.

[4] R. Colella, M. Wheatley, and K. Mills, COMSAT/NBS Experiment Plan For Transport Protocol, NBSIR 85-3141, National Bureau of Standards, March, 1985.

[5] K. Mills, M. Wheatley, and S. Heatley, "Prediction of Transport Protocol Performance Through Simulation," Proceedings: SIGCOMM 86, ACM, August, 1986.

[6] K. Mills, and C. Franx, "Open System Interconnection for Real-Time Factory Communications: Performance Results," Workshop on Factory Communications, NBSIR 87-3516, March, 1987.

[7] D. Stokesberry and S. Heatley, "Measurements of a Transport Implementation Running Over an IEEE 802.3 Local Area Network," Proceedings: Computer Networking Symposium, Washington, D. C., April, 1988.

[8] IEEE 802.3 Standard For Carrier Sense Multiple Access With Collison Detection, December, 1984.

[9] IEEE 802.4 Standard For Token-Passing Bus Access Method, February, 1985.

[10] IEEE 802.5 Token Ring Access Method, 1985

[11] American National Standard, Fiber Distributed Data Interface (FDDI) Token Ring Media Access Control (MAC), X3.139-1987.

[12] P. Jacquet and S. Sedillot, "Simulation of Real-Time Message Scheduling Algorithms," The Advanced Seminar On Real-time Local Area Networks, Bandol, France in April 1986, pp. 315-325.

[13] Intel, iNA960 Release 2 Programmer's Reference Manual, January, 1987.

[14] International Organization for Standardization, Protocol for Providing the Connectionless-mode Netowrk Service, ISO/TC 97/SC 16, IS 8473, February, 1987.

[15] Logical Link Control, IEEE Standard 802.2, 1984.

[16] Intel, iNA960 Programmer's Reference Manual, Appendix E, 1984.

[17] S. Heatley and R. Colella, Experiment Plan: ISO Transport Over An IEEE 802.3 Local Area Network, February, 1986.

APPENDIX A

Description of Parameters

Host Recycle RX Buffer:	Length of time between the time the traffic generator receives a RX buffer or RX Interface Data Unit (IDU) from iNA960 and recycles the RX buffer and returns it to iNA960.
Host Generate TX Buffer:	Time it takes the traffic generator to generate and send a TX buffer or IDU to iNA960.
Host Consume Returned TX Buffer:	Time it takes the traffic generator to receive a returned TX buffer from iNA960.
Host MIP Send (send a message):	Time it takes the host MIP send process to send a message.
Host MIP Receive (receive an AK):	Time it takes the host MIP receive task to receive an AK.
Host MIP Receive (receive a message):	Time required by the host MIP receive task to receive a message and acknowledge it.
Comm MIP Send (send a message):	Time taken by the comm MIP send task to send a message.
Comm MIP Send (time taken for the buffer to be acknowledged):	The elapsed time between transmission of a message in the comm MIP send task and an AK of the message.
Comm MIP Receive (receive a msg):	Time it takes the comm MIP receive task to receive a message and acknowledge it.
Comm MIP Receive (receive an AK):	Time taken by the comm MIP receive task to receive an AK.
Post TX Buffer:	Time taken to post a TX buffer.
Post RX Buffer:	Time taken to post a RX buffer.
Send DT:	Time taken to send a DT TPDU successfully.
Send DT Overhead:	Overhead time to send a DT TPDU on a specific connection.
Send DT Fail:	Time taken to discover that a DT TPDU cannot be sent. The send DT may fail because there is no network layer buffer available.

Send AK:	Time taken to send an AK TPDU successfully. This time includes the overhead time.
Receive DT:	Time taken to receive a DT TPDU.
Receive AK:	Time taken to receive an AK TPDU.
DL TX Interrupt:	Time it takes the Data Link transmit interrupt service routine to be completed.
DL RX Interrupt:	The time it takes the Data Link receive interrupt routine to be completed.
Copy Byte M to L:	Time a byte takes to be copied from Multibus memory to local memory.
Copy Byte L to M:	Time a byte takes to be copied from local memory to Multibus memory.

NIST-114A (REV. 3-90)	U.S. DEPARTMENT OF COMMERCE NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY	1. PUBLICATION OR REPORT NUMBER NISTIR 4530
	BIBLIOGRAPHIC DATA SHEET	2. PERFORMING ORGANIZATION REPORT NUMBER
		3. PUBLICATION DATE APRIL 1991

4. TITLE AND SUBTITLE

VALIDATION OF AN OSI TRANSPORT CLASS 4 SIMULATOR

5. AUTHOR(S)

Okhee Kim Sharon Heatley, and Bob Bishop

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS) U.S. DEPARTMENT OF COMMERCE NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY GAITHERSBURG, MD 20899	7. CONTRACT/GRANT NUMBER
	8. TYPE OF REPORT AND PERIOD COVERED

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)

10. SUPPLEMENTARY NOTES

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

Most work on scheduling in communications protocols has considered only the MAC layer. Token bus, token ring, and FDDI all have priority mechanisms. But mechanisms for scheduling must be extended into the upper OSI layers to be effective. The National Institute of Standards and Technology has chosen to study possible scheduling mechanisms for Transport Class 4 by first developing a detailed Transport Class 4 simulator. This discrete event simulator is written in Simscript II.5 and simulates the Intel iNA960 implementation of Transport Class 4 running on a 186/51 front-end board with a 286/10 host. In this paper, the model is described, the processing times which serve as inputs into the model are specified and the validation experiments are documented.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)

Communication protocols; open systems; OSI Transport; protocol performance; protocols; Quality of Service, QOS; Transport Class 4; Transport Class 4 Simulator; Transport Class 4 Validation.

13. AVAILABILITY		14. NUMBER OF PRINTED PAGES
X	UNLIMITED	35
	FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).	
	ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.	15. PRICE
x	ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.	A03

ELECTRONIC FORM