

TRANSPORT LAYER PERFORMANCE TOOLS AND MEASUREMENT

R. Aronoff
K. Mills
M. Wheatley

Over the past three years the National Bureau of Standards (NBS) has investigated Open System Interconnection (OSI) protocol performance issues. To support this investigation the NBS has developed several transport layer performance tools and has made many performance measurements. This article describes the function and design of a software system and the implementation of a testbed comprising the software and hardware required to evaluate transport layer performance over a local area network.* Further, some performance results are given to demonstrate the utility of the testbed. Other transport layer performance tools, developed by, and measurement projects, conducted at, the NBS are not recounted here, but the appropriate references are included [1,2,6,8,9].

Transport Layer Performance Tools

This section discusses the function and design of software developed at the NBS to serve as a Transport Experiment Control System (TECS). The implementation of the TECS within a local area network testbed is also described. In addition, some limitations of TECS are discussed. The TECS is designed and implemented to overcome difficulties inherent in computer network performance measurement. In particular, the distributed nature of computer networks, an appealing operational characteristic, presents a major obstacle to performance measurement. The physical separation of nodes leads to difficulties with respect to experiment setup and configuration, traffic generation and consumption, synchronization of timing measurements between nodes, and collection and analysis of data.

Function

The TECS performs two major functions—control and simulation. The control function enables the experimenter to configure the testbed nodes appropriately to test a particular hypothesis. The communications software at each network node must be configured according to experiment requirements, transport connections

*Certain commercial equipment is identified in this paper in order to adequately specify the experimental procedure. Such identification does not imply recommendation or endorsement by the National Bureau of Standards, nor does it imply that the equipment identified is necessarily the best available for the purpose.

must be established between appropriate nodes, and experiment start and stop times must be coordinated throughout the testbed. Performance measurements must be taken at all nodes in the testbed during an experiment in a distributed but coordinated fashion. This is crucial in the case of delay measurements of transmissions between nodes. Finally, the control function must enable collation and analysis of data collected from a set of nodes.

The simulation function of the TECS generates and consumes transport user traffic. Application behaviors, representing a broad spectrum of transport users, are simulated.

Design

For centralized control of an experiment, one node is designated as control station for a given experiment session and all other nodes are considered to be remote stations. Connection establishment is centrally managed by the control station even when the control station itself is not a node in the experiment. Figure 1 shows a control station and three remote stations configured with five transport connections (TC). Transport connections are permitted between any combination of stations.

The TECS software, illustrated in Fig. 2, is identical on each node. This provides the ability to use each node in the same way and to easily add and subtract nodes from the testbed. Each major software component—user interface, experiment setup, time control, and application simulation—is discussed in detail below.

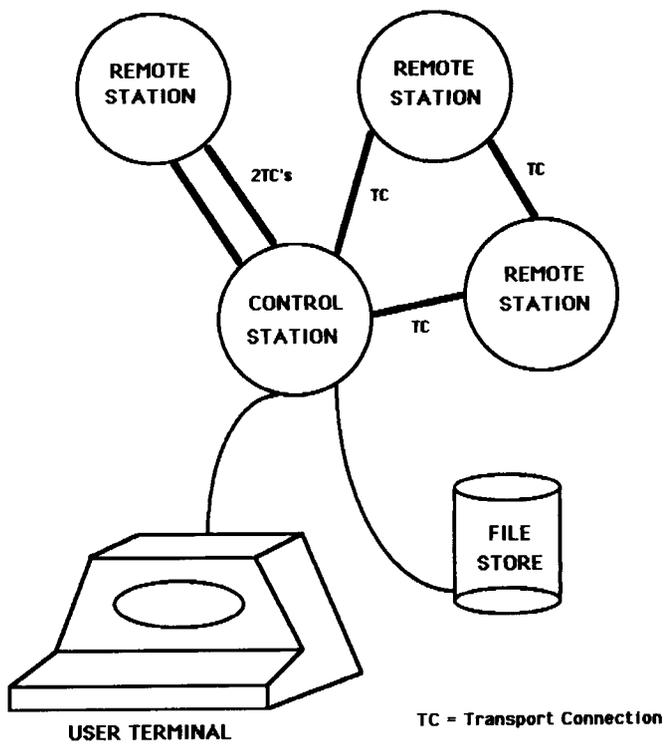


Fig. 1. Simple Performance Test Configuration.

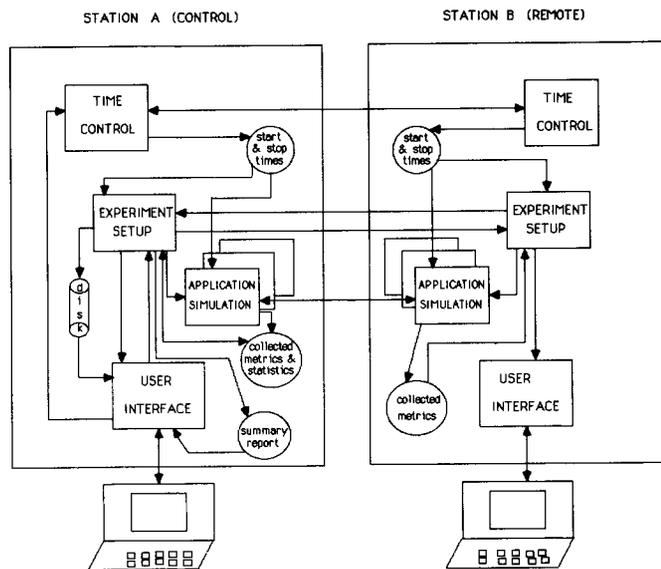


Fig. 2. The TECS Software Architecture.

user interface

The menu-driven user interface provides the user communication with the TECS. Through selection of appropriate menu choices, the user designates one of the nodes as control station; the rest are designated as remote stations by default. At the control station, user

specified input and output file names are passed to experiment setup so that experiment parameters can be read from a file and experiment results can be saved to file. The use of parameter files allows easy and repeatable experiment configuration. A sample input file, which could be used for a uni-directional file transfer experiment between two stations, is presented in Fig. 3.

```

INPUT FILE NAME is : input.uni
|
|   types of application traffic
|
|   1 = uni-directional file transfer sender
|   2 = uni-directional file transfer receiver
|   3 = bi-directional file transfer
|   4 = periodic status report sender
|   5 = periodic status report receiver
|   6 = request for status requester
|   7 = request for status responder
|   8 = data base query requester
|   9 = data base query responder
|  10 = data entry requester
|  11 = data entry responder
|  12 = virtual terminal requester
|  13 = virtual terminal responder
|
|   types of distributions
|
|   C = constant
|   U = uniform
|   N = normal
|   P = poisson
|   E = exponential
|   G = gamma
|
| LOGICAL CONNECTION NUMBER      MAX TPDU SIZE
|
|      1                          2048
|
|   CONNECTION INITIATOR INFORMATION
|
| station applic traf task conn retrans send  rcv  send  rtr
| number  type  priority (0=highest) prio  timer  idu  idu  idu  buf
|         |         |         |         | (msec) (octets)(octets)(octets)(octet
|         |         |         |         |         |         |         |         |
| 4       1       230       3         800       1024  1024  10240  1024
|
|                                     TSU 1
|
| probability of sending  size distribution  P1  P2  arrival distribution  P1  P2
| 1.0                    C                 1024  C
|
|   CONNECTION RECEIVER INFORMATION
|
| station applic traf task conn retrans send  rcv  send  rtr
| number  type  priority (0=highest) prio  timer  idu  idu  idu  buf
|         |         |         |         | (msec) (octets)(octets)(octets)(octet
|         |         |         |         |         |         |         |         |
| 2       2       230       3         700       1024  1024  10240  1024

```

Fig. 3. Sample TECS Input File

The user interface of the control station also accepts the desired experiment duration defined by start and stop times. These times are sent to time control for conversion to system times and for distribution to all nodes via the network. During the experiment, the user may monitor the progress of the experiment or terminate it prematurely. Upon completion of an experiment, menu selections offer the opportunity to view brief statistical summary report, run another experiment or terminate the session. The user interface on the remote stations merely permits monitoring of experiment progress and terminates the session when finished.

experiment setup

Experiment setup starts the experiment, retrieves the stored results from throughout the network, performs the statistical analyses, and saves the results in an output file and writes a summary report. Experiment setup at the control station reads the experiment configuration parameters from the designated input file and, on a connection-by-connection basis, passes the information to experiment setup on those stations specified as connection initiators. Configuration information for the connection recipient stations is also sent to experiment setup on the station which initiates the connection.

that this information can, in turn, be forwarded to the station receiving the connection. This relaying of experiment setup information is accomplished using normal transport layer services.

Once all information is received on the appropriate stations for the desired connections, experiment setup on these stations: 1) establishes the requested type of application simulation task for each connection end-point, 2) maps these tasks to transport connections, 3) passes along any experiment configuration parameter values needed to properly configure the tasks, 4) uses the network management facility to configure the transport connections according to the experiment parameters, and 5) then waits for the experiment start time to be reached.

When the start time of the experiment is reached, experiment setup on each station signals the local application simulation to begin and suspends until all application simulation is complete. Then experiment setup on the control station collects experiment measurements from the remote stations, performs statistical analyses, writes the results to the output file, creates a results summary display, and signals the user interface. Experiment setup on the remote stations signal their user interface after sending experiment results to the control station. Figure 4 shows a sample output record which might be generated by TECS for an experiment involving uni-directional file transfer between two stations.

```

EXPERIMENT NAME is name
EXPERIMENT DESCRIPTION :

Experiment start time: 119:15:38.5658
Experiment stop time: 119:15: 4, 110
Duration: 0: 0:34.5548

Logical connection number: 1
  Station # 2 Application type: uni-directional file transfer receiver
    no user throughput cps and bps
    TSDU type received: #1
      total number of TSDUs received: 30
      mean one_way delays: 0.14145 average TSDU size: 1024
      maximum 1_way delays: 0.59970 TSDU size: 1024
      minimum 1_way delays: 0.10020 TSDU size: 1024
      standard deviations: 0.12655
    no 2_way delay measures
  Station # 4 Application type: uni-directional file transfer sender
    user throughput cps: 903.52941
    user throughput bps: 7228.23529
    there are no one_way measures
    no two_way measures

TOTAL THROUGHPUT PER STATION
station # 2 user throughput cps: 0.00000
            user throughput bps: 0.00000
station # 3 user throughput cps: 0.00000
            user throughput bps: 0.00000
station # 4 user throughput cps: 903.52941
            user throughput bps: 7228.23529
station # 5 user throughput cps: 0.00000
            user throughput bps: 0.00000

AGGREGATE USER THROUGHPUT
user throughput cps: 903.52941
user throughput bps: 7228.23529

```

Fig. 4. Sample TECS Output File.

time control

Time control on the control station receives experiment start and stop times from the user interface, con-

verts these times to standard system-wide time, and distributes them to all stations in the experiment. The task then returns to a receiving mode to do time conversion and distribution of time changes. Time control on remote stations waits to receive the experiment start and stop times from the control station.

application simulation

The TECS generates transport user traffic simulating a variety of applications, including: 1) one-way file transfer, 2) bi-directional file transfer, 3) query/response, 4) periodic status reporting, 5) data entry, and 6) virtual terminal. The experimenter describes an application by specifying message size and interarrival time distributions and probability of generating each message type supported by the application.

Experiment setup on each station initiates appropriate application simulation for each connection end-point. Data messages are time-stamped and measurements such as one-way delay, two-way delay, and throughput are collected at each node. The end of the experiment is sensed on each transport connection when a time-stamp exceeds the experiment stop time. Upon reaching the stop time, application simulation ceases, the transport connection is closed, and experiment setup is notified of completion.

Implementation

The TECS design is implemented in "C", under the iRMX operating system, to run on a local area network testbed at the NBS, as illustrated in Figure 5. Four Intel

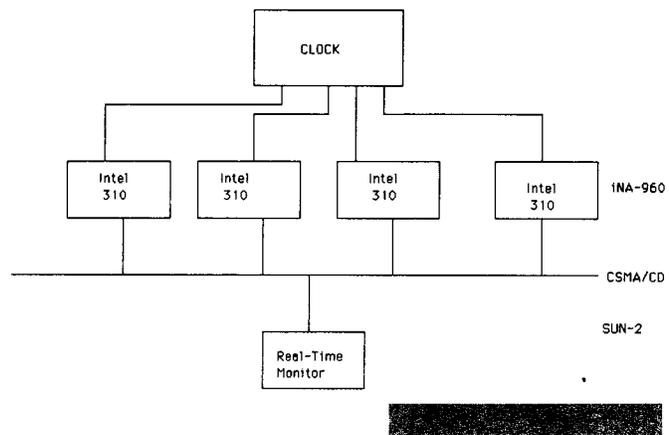


Fig. 5. Experiment Network.

310 nodes and a passive, real-time monitor are connected to a IEEE 802.3 local network. A global clock circuit is connected to each Intel 310 node to provide a synchronized measurement clock. The internal architecture of each node is shown in Fig. 6.

OSI communication services are provided by Intel's iNA-960 software [4] running on a 186/51 COMMputer™ board [5]. The 186/51 contains two processing elements: an 80186 (transport, network, and logical link control) and an 82586 (media access control). The TECS software runs on a separate host board based on an

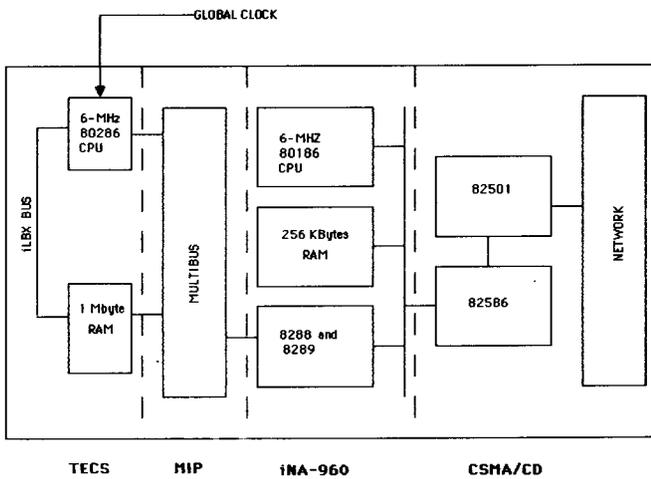


Fig. 6. Architecture of an Intel 310 Network Node.

80286 processor [3]. Communication between the host and COMMputer board is via message passing using the Multibus Interprocessor Protocol (MIP) [4].

Figure 7 shows how the global clock board provides a synchronized 100 us pulse to each Intel 80286 CPU board. The clock pulse is connected to a 16-bit programmable interval timer (PIT). The PIT overflows every 6.5 seconds causing a 16-bit software clock to be updated. The entire 32-bit clock is available to user software. Initial global clock synchronization among all stations on the network is accomplished in the following way. A toggle switch is turned off to stop the clock pulse

SYNCHRONOUS CLOCK SYSTEM

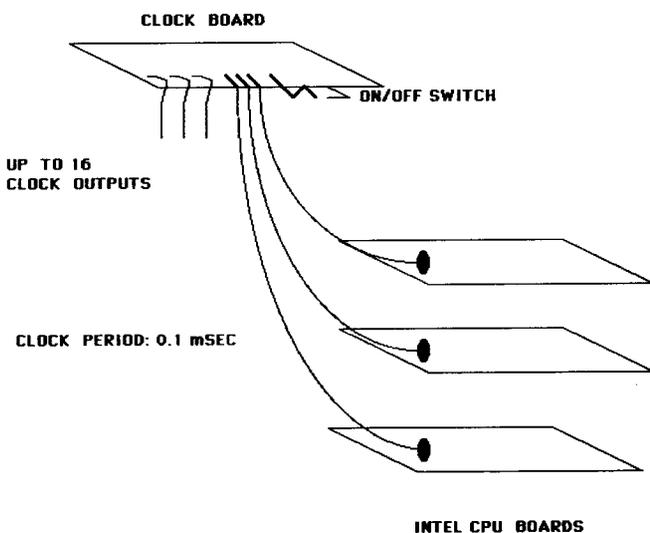
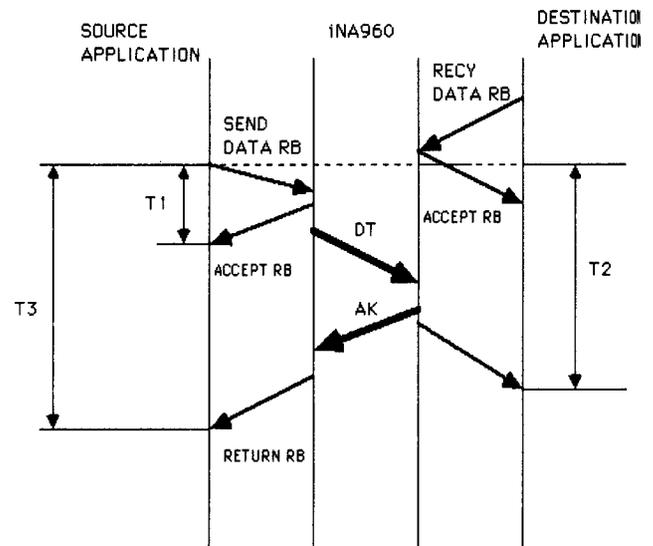


Fig. 7. The Global Synchronized Clock System.

generator. The software on each active station, sensing the clock being off, resets both the hardware (PIT) and software clock counters of that station to their initial states and notifies the user of its readiness to continue. When all stations have indicated their readiness, the toggle switch can be manually returned to the "on" position, thus starting the global clock.

Figure 8 illustrates the time delay measurements made within the application simulation tasks. A user task requests communications services by issuing a request block (RB) to iNA-960 via a system call. The time required to return from the system call is measured as T1. Once the iNA-960 has provided a requested service, the RB is returned to the user program. T3 measures the time elapsed between issuance of the RB and its return. An RB normally contains a user message within it. T2 measures one-way delay for user messages. T4 measures the duration of an experiment.



- T1: RB accept delay
- T2: User message delay
- T3: RB return delay
- T4: Experiment duration

Fig. 8. User Delays Measured

Limitations

TECS has several limitations as discussed below. One of these is the nonportability of the actual TECS implementation. Because TECS makes extensive use of operating system facilities to interface with the particular transport implementation, to make use of the network management facilities of the communications software, and to provide an interrupt handler for the global clock mechanism, the implementation of TECS is realized within the context of Intel's iRMX operating system and relies on properties of that operating system. Therefore, although the overall design of TECS is generalizable to other arbitrary LANs, this implementation of TECS is quite specific to the particular hardware

and software constituting the nodes on the described LAN testbed. Additional stations of the same type could easily be managed by TECS on this LAN by merely putting the TECS software on these nodes and adding the addresses of these nodes to the address database. Dissimilar stations, however, could not be accommodated by TECS.

In a slightly different vein, while TECS does enable good control of traffic patterns on the network by means of input parameters which appropriately configure the experimental environment, it does not permit changes to the actual algorithms of the underlying software. Thus, TECS does not provide a facility for manipulating such communication elements as scheduling strategies within the communications software, adaptive timer strategies, or acknowledgement strategies. At present, such investigations are made by actually modifying the communications software.

Another limiting feature is that TECS, as presently implemented, only makes use of a limited number of the network management facility services which enable the reading and/or setting of certain resources. Therefore, the TECS, itself, cannot collect some types of data, such as the number of retransmissions. This is currently done by other facilities such as separate network management software and a passive real-time transport protocol performance monitor. Future improvements to TECS will provide for a more thorough use of the network management facility so that TECS will perform many of the monitoring and reporting functions currently performed by these additional facilities.

Finally, the minimum granularity of 100 micro-seconds for the global clock might appear, at first glance, to be a problem. However, this granularity has proven quite adequate since most end-to-end delays occur in the millisecond range.

Transport Layer Performance Measurements

This section demonstrates the use of TECS to provide a performance evaluation of Intel's iNA-960 within the testbed discussed above.* The variables controlled by the TECS are listed below (Table I). Another set of variables, such as retransmission timer values and transport message sizes, are controlled by TECS on a connection-by-connection basis using iNA-960 network management services. The network management services are also used to monitor lower level measures such as collision counts, count of packets sent and received, and number of packets dropped due to buffer overrun. A passive, real-time monitor enables unobtrusive evaluation of experiment progress—indicating number of connections, number of retransmissions, protocol efficiency, and total data sent [7].

**An early release of iNA-960 was used for this performance evaluation. In the release, the communications software was implemented within a portable executive that was in turn implemented within Intel's iRMX operating system. Later releases of iNA-960 are implemented without iRMX. Preliminary tests on a later release of iNA-960 show that maximum achievable throughput increased to 120 Kcps and the lowest achievable one-way delays dropped to 18 ms.*

TABLE I
TRAFFIC GENERATION VARIABLES

Application Priority
Inter-message Delay
Duplex or Simplex Data Flow
User Message Size
Total Data Transferred
Number of Connections
Number of User Buffers

The experiments divide naturally into three sets: 1) throughput profile, 2) delay profile, and 3) multi-application profile. Measured results for each set are discussed in the following sections.

Throughput Profile

The throughput profile shows total measured throughput for increasing buffer allocations, as the number of connections increase from one to four. Only simplex data flows are considered. A flow control problem is discussed. User message sizes are always 10K octets. To achieve the best aggregate throughput the value for the retransmission timer parameters was found to depend on load at each node. (Acceptable values were determined hueristically. The results of these trials have not been included here.)

simplex transfer

Figure 9 shows the total throughput measured during simplex data transfer, between two Intel 310 systems, as the number of buffers per connection is varied. Measures are shown for one, two, three and four transport connections. The minimum values for the adaptable retransmission timers used for each experiment are given below (Table II). Throughput ranged between 60 Kcps and 108 Kcps. With only two buffers per connection endpoint available, throughput is increased (Fig. 9) by adding connections because unused capacity is available within the system. Once four buffers are available per connection, the unused capacity is reduced and the

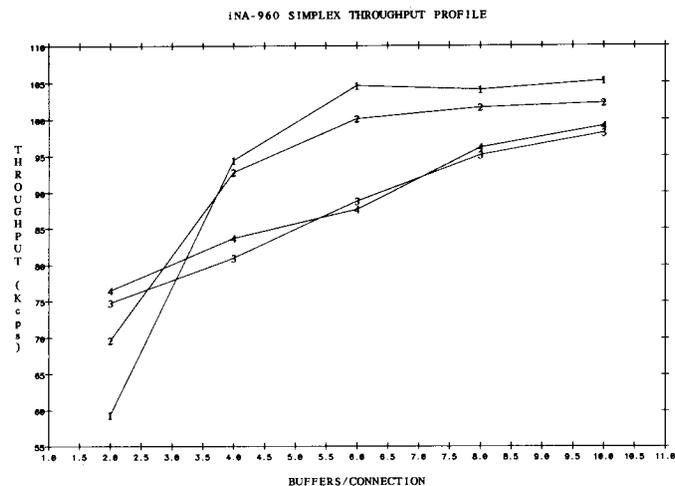


Fig. 9. Simplex Throughput Profile.

TABLE II
RETRANSMISSION TIMER VALUES FOR SIMPLEX THROUGHPUT

Connections	Minimum (Secs.)	Starting (Secs.)
1	.256	.512
2	.512	1.024
3	.819	1.638
4	1.024	2.048

overhead associated with connection scheduling becomes evident. Little throughput difference was observed between three and four connections.

flow control problem

During the throughput experiments a problem was found with the combination of the OSI transport protocol operating over a type 1 class 1 logical link control protocol. The problem is illustrated using the two throughput curves shown in Fig. 10. One curve (single sender) shows a pair of identical machines engaged in a two-connection simplex data transfer. As the number of transport buffers per connection increases, the throughput increases toward 104 Kcps. No matter how many transport receiver buffers are offered, the receiver's link buffers cannot be overrun because only two machines are involved and both machines have identical processing capabilities.

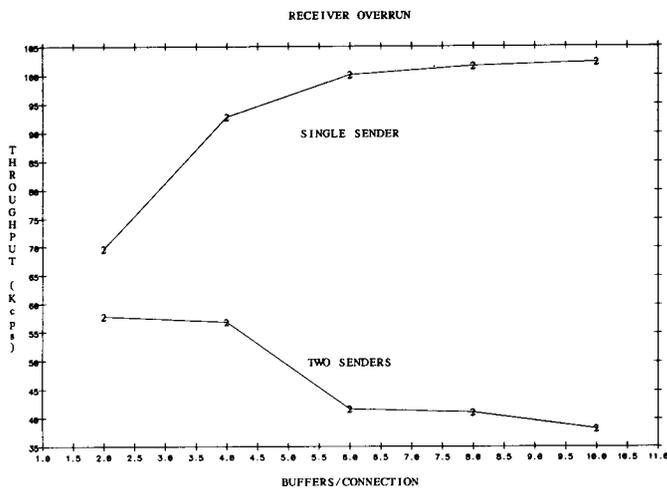


Fig. 10. Receiver Overrun.

The second curve on Fig. 10 (two senders), indicates what happens if the sending machine is faster than the receiver or if two machines are sending to one receiver. As the number of transport receive buffers per connection increases, the throughput decreases toward 35 Kcps. The lost throughput occurs because the transport flow control window, a direct reflection of the number of receive buffers, allows the link level buffers of the receiver to be overrun, invoking transport layer retransmission procedures. This conclusion was verified by using the passive real-time monitor and the iNA-960 network management services. The real-time monitor detected the transport layer retransmissions and the network

management services at the receiving station revealed an increase in the number of link layer packets dropped due to insufficient buffer space.

One-Way Delay Profile

This section presents a profile of one-way user message delays measured under a variety of conditions. In all of the delay experiments, the user message size is varied between 100 and 10,000 octets. However, when a user message is large, protocol segmenting is required because each link packet will hold only 1500 data octets. The sending user on each connection submits one message and waits for an acknowledgement indication before submitting the next message. This stop-and-wait operation limits the overall load on iNA-960 during the delay experiments.

single connection delays

Figure 11 presents measured one-way delays with and without checksum enabled. The message transfers occur over a single connection in a single direction. The receiver allocates three transport receive buffers so that no delay is incurred for closing and reopening the transport flow control window. The lowest delays obtained occur with 100-octet user messages and no checksum, 33.5 ms average and 70.5 ms maximum. The addition of the checksum raises the lowest delays to 38.4 ms average and 78.8 ms maximum*. As expected, the effect of the checksum on delay is more significant as the message size increases.

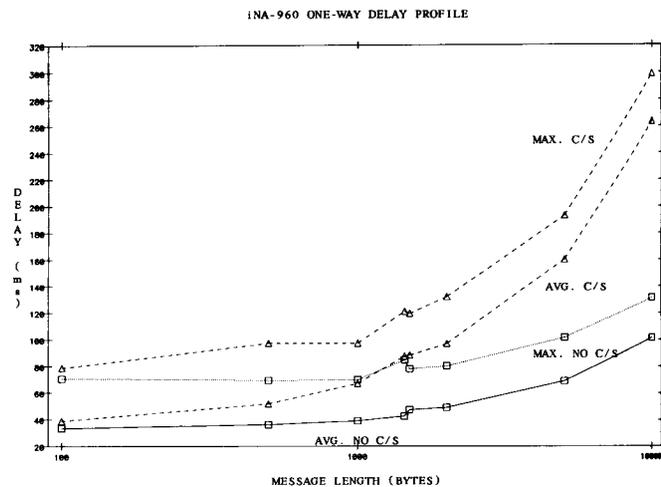


Fig. 11. One-way Delay Profile.

multi-connection delays

Figures 12 and 13 illustrate the effect of multi-connection traffic on one-way delays. For the results in Fig. 12,

*The OSI transport standard requires that the checksum be placed at a location within each packet header. The location may vary; therefore, most present day implementations must compute the value using software algorithms.

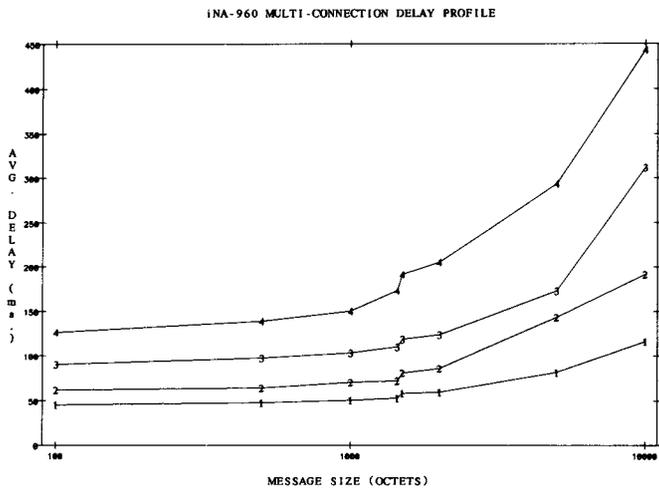


Fig. 12. Multi-connection Delay Profile with Taut Flow Control.

the receive buffers are limited to one per connection. Thus, the effect of closing and reopening the transport flow control window is evident. The lowest one-way delays are obtained with a single connection and 100-octet messages, 45.3 ms average and 88.6 ms maximum. This means that, on the average, 11.8 ms is required to handle reopening of the transport flow control window (comparing Fig. 12 with Fig. 13). In the maximum case, 18.1 ms is required. While this overhead is costly at low loads, it serves to control the one-way delay as the load increases.

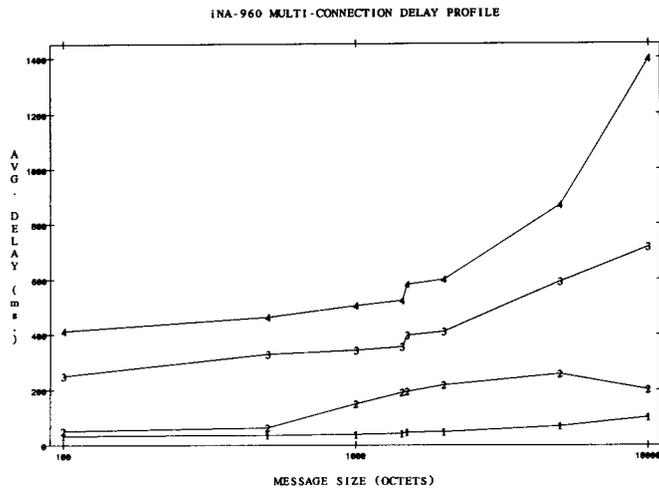


Fig. 13. Multi-connection Delay Profile Without Taut Flow Control.

Figure 13 illustrates the same experiment with three transport receive buffers allocated to each connection. As the load increases for two, three, and four connections, the average and maximum one-way delays increase significantly. The upper bounds on one-way delay in the previous case were 443.2 ms average and 904.5 ms maximum. The upper bounds in this experiment are 1395.3 ms average and 2076.0 ms maximum.

An increase of this magnitude is almost certainly due to a queuing delay incurred at the receiving user program.

The user program submits empty receive buffers to and accepts filled receive buffers from iNA-960. As configured, iNA-960 gives a higher priority to processing of transport operations, including passing filled receive buffers to the user, than to accepting empty receive buffers from the user. Therefore, a user's receive queue grows during periods when the user program is blocked waiting for iNA-960 to accept an empty receive buffer. This effect is demonstrated by an increasing request block accept delay (T1 in Fig. 8) as iNA-960 traffic intensity increases. This effect might be reduced if the MIP task on the 186/51 is run at a higher priority than the iNA-960 task.

Multi-Application Profile

The next set of experiments involves a pair of application simulation tasks on each of two Intel 310 systems. The first pair of tasks is generating bulk data traffic. The second pair of tasks simulates a status report application, submitting messages at a rate sustainable by the system so that no queuing delay is included. The load caused by the bulk data transfer is controlled by varying, between 400 octets and 40K octets, allocation of transmit and receive buffers. Status report messages are fixed at 100 octets. The operating system priority of the status reporting task is higher than that of the bulk data task.

Figure 14 shows the experiment results when the data flow for both applications is in the same direction. The abscissa plots throughput of the bulk data transfer. The ordinate plots the average one-way delay for status report messages. Ideally the status report message delays (average and maximum) will be kept near the lowest delays available from the system. These target delays are superimposed on the graph with dashed lines.

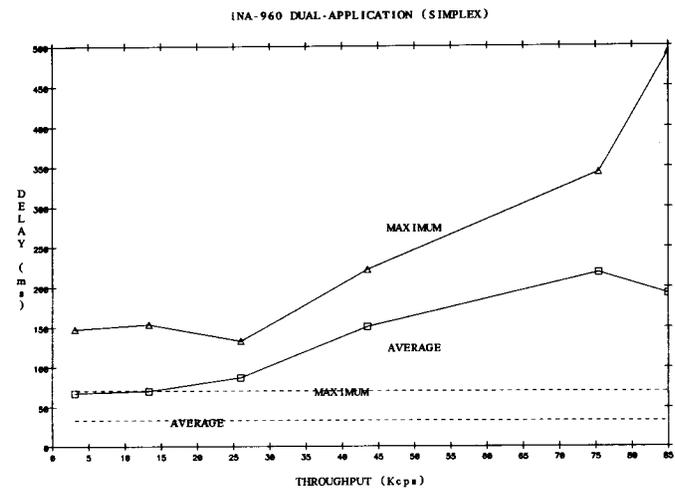


Fig. 14. Dual-application (Simplex).

The results show that the status report delays increase in a pattern similar to that seen when message sizes increase (Fig. 13, two connections) though the status report messages do not increase in size. Also note that the lowest average and maximum delays increase over the ideal by 100%. These results represent unacceptable behavior for applications requiring real-time response. The only control mechanism available in the OSI trans-

port standard is the allocation of buffer space. Therefore, iNA-960 does not provide priority scheduling for transport layer connections and the MIP implementation contains no multi-queue mechanism. Thus, the host operating system task priority mechanism is not complemented by necessary control mechanisms in the communication system software. These issues are the subject of future work.

Figure 15 gives the results of the same multi-application experiment except that status reports and bulk data flow in opposite directions. These results show the lowest average delay is 500 percent above the ideal, while the lowest maximum delay is 300% above the ideal. Although these results are much worse than for the simplex case, the delays do not rise much as the bulk data throughput increases.

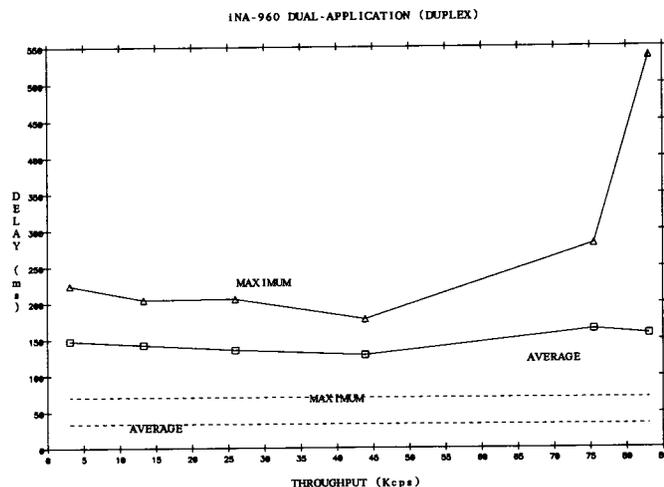


Fig. 15. Dual-application (Duplex).

Conclusions

Performance experiments within a distributed computing environment are difficult to configure, coordinate, and control. This paper described a set of tools, designed and implemented at the NBS, to carry out such experiments. The collection of tools, known as the Transport Experiment Control System (TECS), includes software for experiment management, time control, application simulation, measurement, data collection, and statistical analysis. The TECS is augmented by a global, synchronized clock, network management software, and an NBS developed passive, real-time transport protocol performance monitor. The scope of these tools demonstrates the effort required to produce performance measurements from easily configurable experiments in a distributed systems environment.

The application of these tools was demonstrated by a performance evaluation of Intel's iNA-960, an early OSI transport layer product. The throughput profile shows that the maximum measured user throughput is about 108 Kcps. To achieve the best aggregate user throughput, the values for the retransmission timer parameters on each connection had to be increased as the number of active connections increased. Flow control problems may occur

when an OSI transport protocol is used over a type 1 class 1 logical link control protocol.

The delay profile shows that the minimum measured one-way delay for iNA-960 is about 33.5 ms without checksum and 38.4 ms with checksum. The time for reopening a closed transport flow control window is about 11.8 ms. Granting a single message credit per connection increases the delay at low loads, but serves to control the delay at high loads. As the flow control window for each connection is allowed to exceed one message and the load on iNA-960 is increased, one-way delays increase significantly. This increase is probably due to a queuing delay at the receiving application and can be reduced by reconfiguring iNA-960.

OSI protocol standards, as now specified, do not provide adequate mechanisms for guaranteeing real-time performance for selected connections. This weakness in the standards is demonstrated by the iNA-960 multi-application profile where high throughput transport connections dominate the available resources forcing up the delay on all connections.

Acknowledgments

Cornelis Franx of Philips designed and implemented the global synchronized clock hardware and software. Sara Collins and Tuong Nguyen implemented application simulation tasks for the TECS. Intel donated much of the hardware and the iNA-960 software required to implement the testbed.

References

- [1] D. M. Chitre, et al., "A joint COMSAT/NBS experiment of transport protocol," *Proc.: VII Digital Satellite Conference*, May 1986.
- [2] R. Colella, R. Aronoff, and K. Mills, "Performance improvements for ISO Transport," *Proc.: Ninth Data Communications Symposium*, IEEE, Sept. 1985.
- [3] *Guide to Using the iSBC 286/10 Single Board Computer*, Intel Corporation, 14627-001, Santa Clara, CA, 1983.
- [4] *iNA 960 Programmer's Reference Manual*, Intel Corporation, 122193-001, Santa Clara, CA, 1984.
- [5] *iSBC 186/51 COMMputer Board Hardware Reference Manual*, Intel Corporation, 122136-002, Santa Clara, CA, 1984.
- [6] W. McCoy, K. Mills, and R. Colella, *Implementation Guide for ISO Transport Protocol*, National Bureau of Standards, ICST/SNA 85-18, Dec. 1983.
- [7] K. Mills, J. Gura, and C. M. Chernick, *Performance Measurement of OSI Class 4 Transport Implementations*, NBSIR 85-3104, Jan. 1985.
- [8] K. Mills, M. Wheatley, and S. Heatley, "Prediction of transport protocol performance through simulation," *Proc.: SIGCOMM 86*, ACM, Aug. 1986.
- [9] J. Vinyes, E. Vazquez, and K. Mills "Throughput analysis of a transport protocol over An X.25 network," *Proc.: Computer Networking Symposium*, IEEE, Nov. 1986.

Robert Aronoff is a computer scientist and project leader in the Protocol Design and Measurement Group of the Institute for Computer Sciences and Technology (ICST) at the National Bureau of Standards (NBS) Gaithersburg, Maryland. Mr. Aronoff joined the NBS in 1984 and has worked on the design and development of protocol measurement tools used in performance evaluation and analysis of OSI protocols operat-

ing over a wide range of network systems. He received his B.A. in Philosophy from Haverford College and a Master's degree in Computer and Information Science from Temple University. He is currently involved in investigating and defining management issues of OSI networks, as well as developing an OSI network management prototype implementation.

Kevin Mills, Acting Division Chief of the Systems and Network Architecture Division of the ICST, joined NBS in 1982 and established the OSI protocol performance research program. This research program resulted in international collaboration between government, industry, and academic institutions to evaluate and enhance the performance of OSI protocols. Prior to joining the NBS, Mr. Mills developed communications performance measurement products at Tesdata Systems Corporation. He performed data communication research and develop-

ment for the System Development Corporation and the United States Marine Corps. He received an M.S. from the American University and a B.S. from Frostburg State College.

Marnie Wheatley is a computer scientist in the Protocol Design & Measurement Group of the Institute for Computer Sciences and Technology (ICST) at the National Bureau of Standards (NBS), Gaithersburg, MD. Ms. Wheatley joined the NBS in 1982 as system manager for the ICST Experimental Computer Facility. Since 1983 she has worked on the design and development of protocol measurement tools used in the performance evaluation and analysis of OSI protocols operating over a wide range of network systems. She received a B.S. in Mathematics from Mary Washington College and a Master's degree in Computer Science from the George Washington University.