PERFORMANCE ANALYSIS OF DATA TRANSFER IN GM MAP

W. Timothy Strayer
Sharon K. Heatley
Alfred C. Weaver

Computer Science Report No. TR-87-18
October 22, 1987

This paper has been submitted to *IEEE Networks Magazine*, October 1987.

Performance Analysis of Data Transfer in GM MAP

W. Timothy Strayer*
Sharon K. Heatley**
Alfred C. Weaver*

*Computer Networks Laboratory, Department of Computer Science, University of Virginia, Charlottesville, VA 22903.

**Institute of Computer Sciences and Technology, Systems and Network Architecture Division, National Bureau of Standards, Gaithersburg, MD 20899.

The General Motors Manufacturing Automation Protocol (GM MAP) is being developed as an attempt to standardize communications between intelligent manufacturing and controlling devices. Intel Corporation has developed a communications board as a front-end processor to be used in Intel Multibus computers, which with MAPNET2.1 software provides an implementation of MAP version 2.1. The Data Link and the Transport Layers were studied through performance analyses. For the Data Link Layer we measured the one way delay and the throughput for various packet sizes. We also measured the number of messages that could be sent and received per second for a range of Data Link packet sizes. For the Transport Layer we found a buffer configuration which optimized throughput and used it for experiments that measured throughput as the dependent variable. Throughput was measured with respect to TSDU message size. The effects of decreasing the retransmission timer, varying the maximum Transport Protocol Data Unit size, using multiple virtual circuits, and varying the maximum window size are described. One way delay was measured with respect to the TSDU message size. Comparisons between the two layers indicated that there are both benefits and drawbacks to using a front-end processor for communications, largely due to message segmentation.

1. MANUFACTURING AUTOMATION PROTOCOL

The development of the General Motors Manufacturing Automation Protocol (GM MAP) is an event which will change the course of history with regard to factory automation. Before 1980, every plant or assembly line which GM built involved a control system which was supplied by one of the major vendors of industrial automation equipment. However, these products could not communicate across vendor boundaries unless someone (usually GM) financed custom hardware and software to overcome the problem. To solve some of the problems of interoperability, General Motors, working with Boeing Computer Systems, developed the Manufacturing Automation Protocol (MAP) [GENE86] in an unusual attempt for the end-user to dictate a standard to its vendors.

One of the goals of MAP is to provide a method by which any system from any vendor can be inserted into the MAP environment without hardware or software customization, similar to the way that stereo components can be purchased independently, plugged together, and still be expected to interoperate. MAP is also designed to provide technical services to control various intelligent devices in a consistent and cost-effective manner, contributing to the automation of the factory floor.

Rather than being a set of newly invented protocols, MAP is primarily a collection of existing national and international standards based on the International Organization for Standardization's Open Systems Interconnection (ISO OSI) reference model as defined by [ISO7498]. The reference model defines a seven layer hierarchy for providing functionality in a modular fashion. The following is a list of the OSI layers and the MAP standards adopted:

OSI LAYER	MAP VERSION 2.1 FUNCTIONS
7 - Application	ISO 8649 Common Application Service Element (CASE) ISO 8571 File Transfer, Access and Management (FTAM)
6 - Presentation	Null
5 - Session	ISO 8327 Session Protocol Specification
4 - Transport	ISO 8073 Transport Protocol Specification Class 4 (TP4)
3 - Network	ISO 8473 Connectionless-Mode Network Service (CLNS)
2 - Data Link	IEEE 802.2 Link Level Control Class 1 IEEE 802.4 Token Passing Bus Access
1 - Physical	IEEE 802.4 Token Bus with Broadband (10 megabits/second) or Carrierband (5 megabits/second) Modulation on Coaxial Media

The MAP network can be a collection of segments, some with MAP compatibility and some without. Bridges, routers, and gateways provide access to stations on different segments or different networks. Also, a provision is made for expedited communications using an Enhanced Performance Architecture (EPA). EPA bypasses the upper layers in the OSI model and provides application processes with direct access to the Data Link services.

GM MAP has been through several versions thus far and will undoubtedly continue to change. Most commercial MAP hardware and software, including the system we tested, conforms to MAP version 2.1. Some extensions were made in version 2.2; version 3.0 has been circulated in draft form but is not expected to be established as the new specification until 1988.

2. ENVIRONMENT

The hardware consisted of two Intel 286/310 computer systems serving as nodes on a MAP network. Each system contained one 6 MHz Intel 80286 microprocessor with 80287 math co-processor and one megabyte of RAM, operating over a Multibus. The MAP network connection was provided by the Intel iSXM554 COMMengine, which consisted of an Intel

80186 microprocessor, 256 kilobytes of RAM for use as a packet buffer, and the hardware for the token bus interface. The COMMengines were connected to the bus (coaxial cable) by multitaps. The other end of the bus was connected to an Industrial Networking Incorporated (INI) head-end remodulator [INDU85a,b]. A custom-designed global synchronous clock [FRAN86] provided a time reference accurate to 100 μs.

The Intel computers used the iRMX86 [INTE85] real-time multi-tasking operating system. MAP communications services were provided by MAPNET2.1 [INTE87a,b], Intel's implementation of MAP version 2.1. MAPNET2.1 software provided the OSI Application, Session, Transport, Network, and Logical Link Control Layers. The iSXM554 hardware provided the Medium Access Control (MAC) sublayer and the interface to the 10 megabit/second IEEE 802.4 token bus broadband network. The Intel product iNA960 [INTE86a,b,c] provided a subset of the MAPNET2.1 services, specifically the Transport, Network, and Logical Link Control Layers. iNA960 was the product actually used in the experiments because it provided Transport and Data Link services in exactly the same manner as MAPNET2.1 without the complication of the upper layers. All the MAPNET2.1 or iNA960 software resided on the front-end processor.

Our application processes, written in 'C' and 'PL/M' and resident on the host processor, communicated with MAPNET2.1 and with iNA960 using another Intel convention called Multibus Interprocessor Protocol (MIP) [INTE86c]. MIP allows the host processor (the 80286) and the front-end processor (the 80186 on the iSXM554) to communicate request blocks by simply passing pointers to shared memory via a Message Delivery Mechanism (MDM). If the block to be transferred is not in shared memory, MDM will copy the block from host memory into the front-end processor's packet buffer. MIP isolates user tasks from the complexities of communicating across the Multibus by handling the interaction between the host processor, the front-end processor, and other intelligent Multibus devices. MIP supports functions which can locate a port, attach/detach a task to/from a port, and transfer/receive a buffer to/from a port.

3. EXPERIMENTS

Since MAP is designed to operate on the factory floor, we developed a set of tests which was intended to measure its performance characteristics. In MAPNET2.1, CASE and Session Layer data transfer primatives map directly onto Transport Layer data transfer primatives. The Transport virtual circuit service is built upon the datagram service of the Data Link Layer. For these reasons our measurements were targeted toward the Transport and Data Link Layers.

The configuration of MAPNET2.1 (and the iNA960 subset) in common use is not totally compliant with MAP 2.1; it permits Data Link packet sizes up to 1024 bytes, whereas MAP 2.1 specifies Data Link packet sizes up to 8000 bytes. A specially configured version of MAPNET2.1 has passed conformance testing.

Performance Measures

The External Data Link (EDL) is an interface used to access the services of the Data Link Layer directly, allowing users to circumvent the Transport and Network Layers. A user of the EDL accesses iNA960 through the general request block programmatic interface common to all layers in MAPNET2.1 and iNA960. Through request blocks, commands can be issued to establish and terminate Data Link connections, transmit data, post receive buffers for incoming data, configure the Data Link Controller, reassign an individual address for a node, and add and remove multicast addresses to and from a list maintained by the data link node. We measured several aspects of data transfer using this datagram service.

The transport user also accesses iNA960 through this programmatic interface. We used the Transport Class 4 Virtual Circuit Service and measured several aspects of its performance and error recovery mechanisms. We then made comparisons between the datagram service of EDL and the virtual circuit service of Transport, identifying some benefits and drawbacks to using a front-end processor communications board.

One Way Message Delay

The total delay incurred for the delivery of one EDL message required timestamping that message as it was transmitted and noting the time as it was received. To eliminate any queueing delays, one buffer was posted and one packet was sent for each run of the experiment. This ensured that all resources were ready and waiting to service that one packet as it traversed the network. Packet sizes ranged from 16 to 1024 bytes by powers of 2. Four runs of the experiment were performed for each packet size, these delays were averaged and the results are shown in Figure 1. There was a linear relationship between the packet size and the delay suffered during transfer. For packets of length L, the end-to-end delay was approximately (8.5 + L/125) milliseconds.

The delay which a Transport Service Data Unit (TSDU) message suffered as it traversed the network was also measured. The TSDU message sizes ranged from 16 to 1024 bytes by powers of 2 as in the EDL experiments and then from 928 bytes to 5569 bytes by multiples of 928 and those multiples plus one. The TSDU message size of 928 represents the largest amount of data that could be transferred using a single 1024-byte Transport Protocol Data Unit (TPDU) (the remaining bytes are reserved for upper layer headers). One transmit buffer and one receive buffer were used. A delay of 100 milliseconds between the transmission of each TSDU message ensured that all resources were ready and waiting to process that message, and that there would be no queueing delay.

As Figure 2 shows, delays increased linearly with TSDU message size until the TSDU size exceeded 928 bytes. At 929 bytes, a penalty was assessed due to the overhead required to segment the TSDU into two TPDUs. This penalty was repeated each time the TSDU size exceeded a multiple of 928 bytes. The first division of a TSDU into two TPDUs cost about 20 milliseconds; further TSDU segmentation cost 5 to 15 milliseconds per additional TPDU required.

Throughput

The EDL throughput was measured for packet sizes ranging from 16 to 1024 bytes. One hundred buffers were posted on the receiving node and 100 packets were sent sequentially from the transmitting node. This throughput, therefore, was a measure of network efficiency with respect to packet size, and did not necessarily represent the best possible throughput at the Data Link Layer.

The timestamp of the first packet received was subtracted from the time the last packet was received to obtain a total time to process the 100 packets. The total amount of data sent (100 times the packet size) was divided by this total time to arrive at the throughput, shown in Figure 3. There was a very strong dependency between the packet size and the throughput. For packets of length L, throughput was approximately 65L bytes/second.

A transport user buffer is a dedicated segment of user memory which held a message as it was prepared for transmission. When the measurement program submitted a transmit request, the buffer holding the packet to be transmitted was not used again until iNA960 released it. The performance program used the multiple transmit user buffers for multiple transmit requests. Once that message had been acknowledged by the receiver, the user buffer was returned to the circulating queue of free buffers and reused.

By varying the number of transport user buffers allocated by the transmitter and the receiver, the number of transmit and receive buffers needed to optimize throughput was determined. The number of receive buffers was set to a high number (10) and the number of transmit buffers was then increased from 1 to 10. The throughput was measured at each buffer configuration. Seven transmit buffers resulted in the best throughput. Then the number of receive buffers was decreased to one. Again, the throughput was measured at each buffer configuration. The number of receive buffers which resulted in the best throughput was 5. This configuration of 7 transmit and 5 receive buffers was used in all experiments measuring throughput against another variable, and it was used to measure the impact of TSDU message size on throughput. The range of the TSDU message size was the same as the range used in the One Way Delay experiments.

As seen in Figure 4, throughput increased linearly for TSDU message sizes that were equal to or smaller than 928 bytes. At a TSDU size of 929 bytes, throughput suffered due to the overhead associated with segmenting one TSDU into two TPDUs. A decrease in throughput was likewise observed whenever the TSDU size exceeded a multiple of 928 bytes. The curves of the throughputs for each multiple of 928 (the peaks in the graph) and the multiples plus one (the dips in the graph) were quadratic, with the two curves converging asymptotically to approximately 72 and 55 kilobytes/second, respectively. The difference between these two values reflected the penalty for segmentation.

Number of Messages Per Second

The number of EDL messages per second was measured using 100 transmit and 100 receive EDL user buffers for maximum enqueueing. These EDL user buffers were represented similarly to the transport user buffers. One hundred EDL user buffers were used to enqueue 100 messages so that all of the messages could be transmitted at the peak rate. Figure 5 shows the number of messages transmitted per second for various packet sizes. The peak EDL transmission rate observed was 156 128-byte messages per second.

Because the optimal configuration for the transport user buffers was found to be 7 transmit and 5 receive user buffers, the number of messages per second for Transport Layer was a function of the throughput. When throughput was optimized the number of messages per second was optimized as well. Figure 6 shows the number of messages transmitted per second for various TSDU sizes. The peak transmission rate observed for Transport was 56 16-byte messages per second.

Retransmission Timer

The Network Management Facility (NMF) interface provided access to intra-layer configuration parameters such as retransmission timeout interval. The actual retransmission timer uses an adaptive algorithm, adjusting itself to the characteristics of the network. The user of iNA960 cannot set the actual retransmission timer value. Instead, one supplies the Minimum Retransmission Timer setting which is used as the minimum for the actual retransmission timer setting at any point in time. Intel limits the Minimum Retransmission Timer value to no lower than 100 milliseconds during configuration of iNA960. However, we used the NMF functionality to reset this value lower than 100 milliseconds.

The values of these related timer settings ranged from 5 milliseconds to 500 milliseconds. The values at the high end of the range were chosen to show what would happen if the floor (minimum) value of the retransmission timer was reasonably large. The values at the low end of the range likewise show what would happen if the value was unreasonably small. Ten user buffers were used by both the transmitter and the receiver to ensure that the hardware and software tasks remained busy.

As the Minimum Retransmission Timer value was decreased, as shown in Figure 7, the throughput remained mostly unaffected until the value was set at 50 milliseconds. For values above 50 milliseconds there were no retransmissions due to network error, so reducing the minimum value of the retransmission timer did not affect throughput. Throughput peaked at 50 milliseconds and then dropped dramatically for lower values. At these points duplicate TPDUs were being sent and rejected because the retransmission timer was expiring before the TPDUs could be acknowledged. Thus we observed that 50 milliseconds was the lowest usable value that could be supplied to the Minimum Retransmission Timer for an unloaded network. We concur with Intel documentation [INTE86a] that this value should never be set below 100 milliseconds for a general purpose network. However, the implication of setting the Minimum Retransmission Timer to a value no less than 100 milliseconds is that lost TPDUs are not detected for at least one-tenth of a second. The default configured Minimum Retransmission Timer value is 500 milliseconds.

Segmentation

The TPDU is the packet frame and consists of both data and headers. The Network Management Facility allowed the user to specify a maximum TPDU size. iNA960, however, uses the minimum of 1024 bytes (the maximum Data Link packet size allowed in our configuration) and this maximum TPDU size as its actual TPDU size.

Figure 8 shows the relationship between throughput and TPDU size. Throughput is defined as the rate of data delivery in bytes per second and excludes the bytes required for framing. As TPDU size decreases, throughput decreases because the ratio of header bytes to data bytes increases.

Virtual Circuits

We varied the number of virtual circuits, or connections, from 1 to 16 to observe its effect on throughput. As seen in Figure 9, one virtual circuit provided the best throughput because it required the least internal overhead. Even though multiple virtual circuits between two stations provided additional avenues for transfer of data, all virtual circuits used the same physical connection and thus overall throughput was not enhanced. In fact, there was the penalty of maintenance overhead levied on the user of multiple virtual circuits (connections), reducing the overall throughput.

The dramatic reduction in throughput for 3 virtual circuits was an anomaly. The experimental evidence strongly suggested that this was a worst case occurrence. The asynchronous characteristics of the communications between the host and the COMMengine, the number and use of the internal transmit and receive buffers, and the token bus access all contribute to delays imposed by the system. We hypothesize that the poorer performance observed for 3 virtual circuits was due to a worst-case alignment of these factors, although we were unable to verify the hypothesis due to proprietary restrictions on the software.

Window Size

In Transport Class 4, a *window* is an ordering of *sequence numbers* that are termed active. The sequence number identifies and orders a particular TPDU so that the receiver may reassemble multiple TPDUs into one TSDU. The window *slides* to incorporate new sequence numbers as the TPDUs are acknowledged and their sequence numbers become inactive. Thus, the size of the window dictates how many unacknowledged TPDUs a receiver is willing to buffer. The receiver communicates this information via a *credit* field. The receiver can control the flow of data by varying its window size and thus throttle the transmitter by reducing its credit. A window size of 0 is called a *closed window*, and effectively shuts off the transmitter until the window is reopened by the receiver.

The maximum window size was set using the Network Management Facility and throughput was measured to show the impact of lowering the maximum window size. The range of maximum window sizes was 1 to 15, where 15 was the default setting from the iNA960 configuration. This configuration also prevented the window from closing, even when this was appropriate. The decision not to allow the window to close was based on Intel's empirical data which showed better performance at the risk of losing messages due to lack of resources [INTE86a].

As the receiver's buffer space was filled with incoming TPDUs, the number of TPDUs that it had room to receive decreased. To keep from being overrun, the receiver sent a "credit" with the acknowledgements. This credit told the transmitter how many more TPDUs the receiver was prepared to handle; it was in the range of 0 to the maximum window size. As the maximum window size was decreased, the credit was likewise decreased. This caused fewer TPDUs to be in the pipeline to the receiver and thus decreased the throughput. As Figure 10 shows, there appeared to be no effect when decreasing the maximum window size from 15 to 4, but decreasing it from 4 to 1 caused the throughput to suffer dramatically. This indicated that the receiver could not handle TPDUs sufficiently fast for the additional credit greater than 4 to matter. However, credit less than four caused the transmitter to throttle itself to having only one or two TPDUs in the pipeline.

Layer Comparisons

There were benefits and drawbacks of using a front-end processor like the iSXM554 COMMengine. Having the communications services provided by a front-end processor allowed concurrency. The COMMengine and the host processor ran in parallel and interacted via the Multibus. The host processor was not concerned with servicing the messages as they arrived asynchronously from the physical network. More computing time could be dedicated to the user application since the use of the network did not require CPU cycles.

However, the only means of access to the COMMengine and its software was through the MIP interface across the Multibus, which was a bottleneck. By accessing iNA960 through the External Data Link interface, restrictions were placed on the size of the messages that could be transmitted by the user application. Large messages had to be buffered in the host's memory and delivered to the EDL in smaller (1024 byte) segments. In this instance many small messages were sent through the MIP interface, across the Multibus and to the COMMengine, subsequently causing the throughput to suffer.

By accessing iNA960 through the Transport Layer, large messages were delivered to the front-end processor and stored there until they could be processed. iNA960 and the underlying data link hardware worked most efficiently while there was data in buffers on the COMMengine. When iNA960 was finished processing one large message, it could be sent another to be stored in the on-board buffers. This increased throughput by decreasing the number of interactions across the MIP interface.

Figure 11 shows that time-dependent messages should employ the EDL rather than the Transport Layer. There was a constant cost of approximately 7.5 milliseconds associated with using the additional functionality provided with the Transport layer, which may be too expensive for short command/response or status/request messages. The bound on the message size was 1024 bytes for the Data Link Layer. At Transport, the bound was 928 bytes before the TSDU message was segmented into two TPDUs. For messages of length L and smaller than 928 bytes, the one way delay at transport was approximately (16.5+L/500) milliseconds; using the EDL interface it was approximately (9+L/500) milliseconds.

Data Link provided better throughput for packets of the same size, as seen in Figure 12. This was because the TSDU messages were all small enough to fit into one TPDU, so segmentation was not an issue. For messages of length 928 bytes or shorter, the difference between the throughputs reflected the difference between the overhead associated with a relatively simple Data Link data transfer service and the overhead associated with a complicated but reliable Transport data transfer service. Data Link, however, is restricted to packets of size 1024 bytes or smaller. When messages are larger than 1024 bytes, the message must be segmented.

When an application process uses Transport, it moves entire TSDUs onto the front-end processor, then Transport segments the TSDUs into TPDUs as required. If the application process uses EDL, the host must perform the segmentation and deliver packets no larger than EDL is configured to accept. By forcing the maximum TPDU to be the same size as the Data Link packet, we observed the difference between segmentation on the front-end processor by Transport and segmentation on the host by the application process. In Figure 13, the throughput for Transport is slightly greater than throughput for Data Link. This shows that an application process cannot segment messages on the host as well as Transport can segment the messages on the front-end processor. Thus, in terms of performance, segmentation on the front-end processor was more efficient.

4. CONCLUSION

The iSXM554 COMMengine provided a front-end processor which successfully off-loaded the task of communications from the host. Its primary advantage was that it transparently provided segmentation (the reduction of an arbitrary size TSDUs into multiple

TPDUs). Even so, segmentation was expensive — the first segmentation increased end-to-end delay by approximately 20 milliseconds and each additional segmentation added another 5 to 15 milliseconds. When using the largest TSDU not requiring segmentation (928 bytes), throughput averaged 47 kilobytes/second; when using the smallest TSDU requiring segmentation (929 bytes), throughput dropped to 30 kilobytes/second.

Data Link supported a connectionless-mode (datagram) service for packets up to 1024 bytes. Transport used that underlying connectionless-mode service and added segmentation, sequencing, acknowledgment, and reassembly to provide a connection-oriented (virtual circuit) service. As expected, direct access to the Data Link Layer provided shorter one-way delays and higher throughput than did direct access to the Transport Layer. For large messages, however, sending one large TSDU to Transport and allowing it to perform segmentation was more efficient than having the host perform segmentation and pass multiple smaller packets directly to Data Link.

In terms of absolute performance, a two-station system with no other computational tasks using the Data Link interface supported continuous transmission rates ranging from 156 128-byte messages per second to 84 1024-byte messages per second. At Transport, it supported continuous transmission rates ranging from 56 16-byte messages per second to 13 5000-byte messages per second. When Transport was supplied continuously with large messages, segmentation overhead limited the maximum throughput to approximately 72 kilobytes/second. We confirm that Transport's retransmission timer should not be set to less than 100 milliseconds; the implication is that lost packets will not be detected for at least 0.1 second which could be very significant in real-time control systems. We observed that a window size of 4 was sufficient to achieve maximum throughput, and that using multiple virtual circuits decreased data throughput due to circuit maintenance overhead.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the technical assistance and financial assistance of Steve Alten and Matt Katzer at Intel Corporation; Kevin Mills, Dan Stokesberry and Tuan Tu at the National Bureau of Standards; and John DeLaat at the NASA-Lewis Research Center.

LIST OF FIGURES

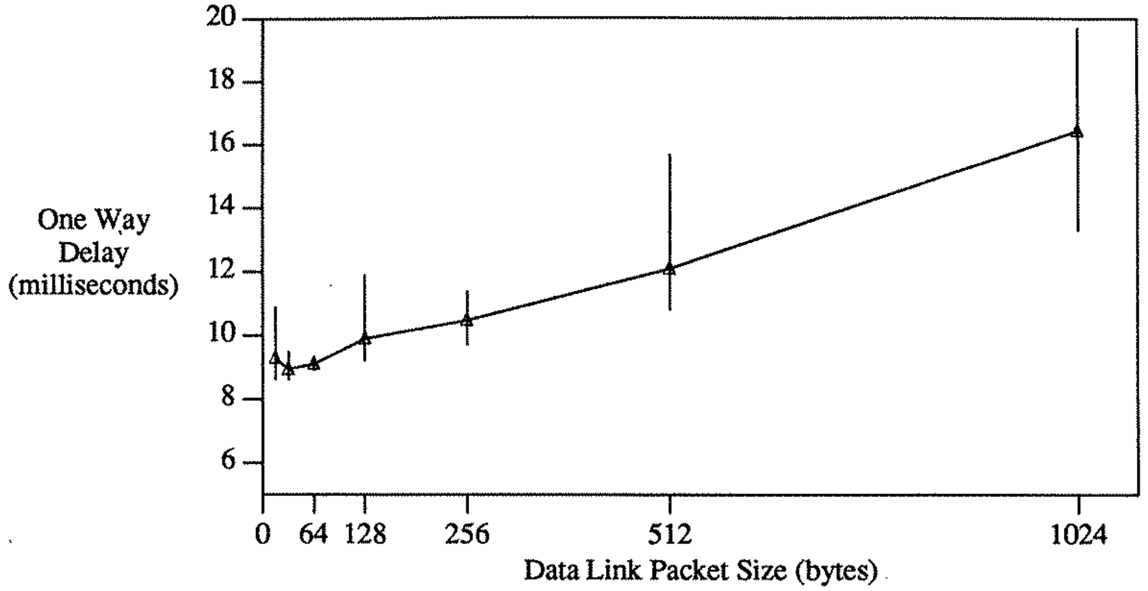

Figure 1 - One Way Data Link Delay vs. Data Link Packet Size

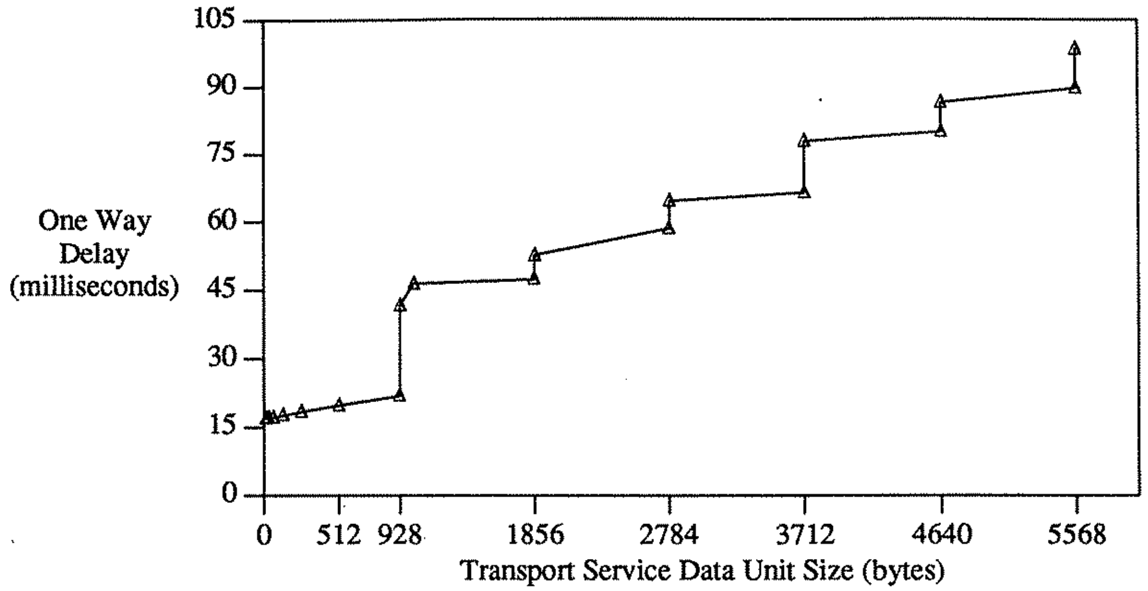

Figure 2 - One Way Transport Delay vs. Transport Service Data Unit Sizes

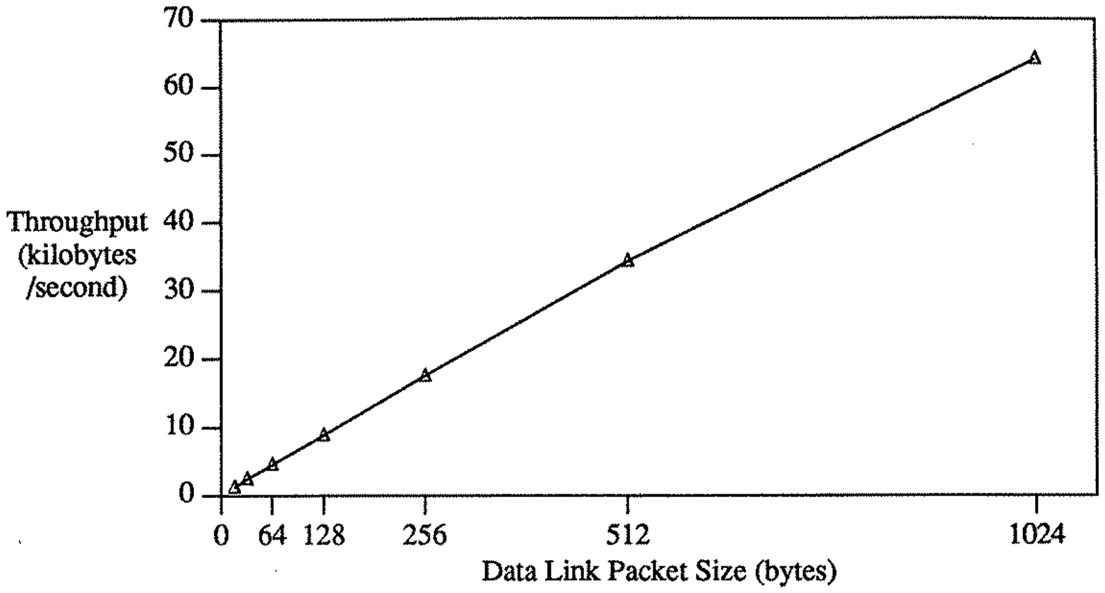

Figure 3 - Data Link Throughput vs. Data Link Packet Size

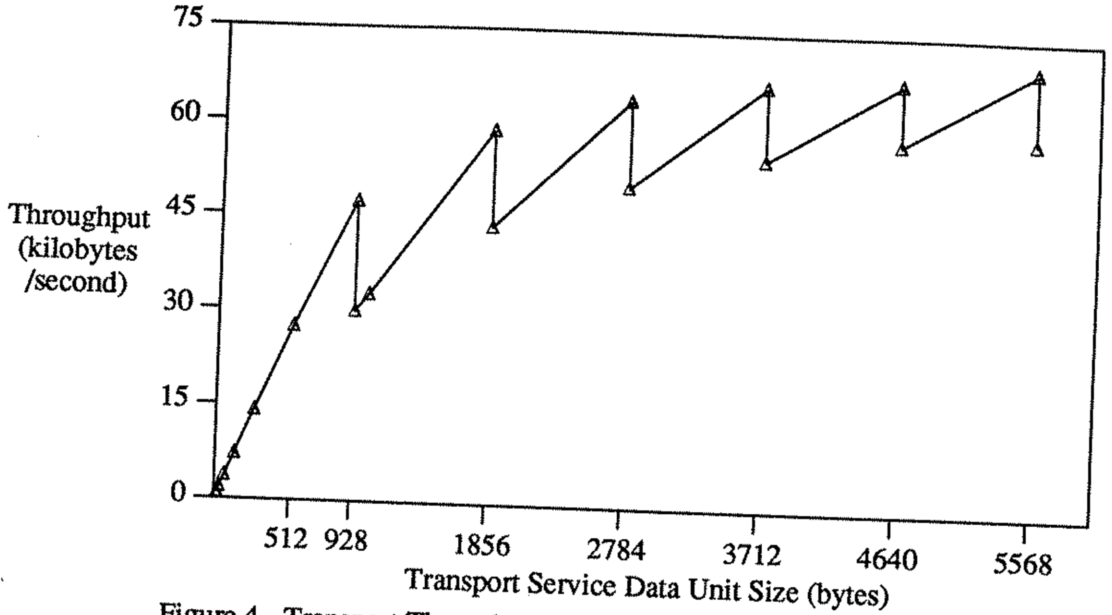

Figure 4 - Transport Throughput vs. Transport Service Data Unit Size

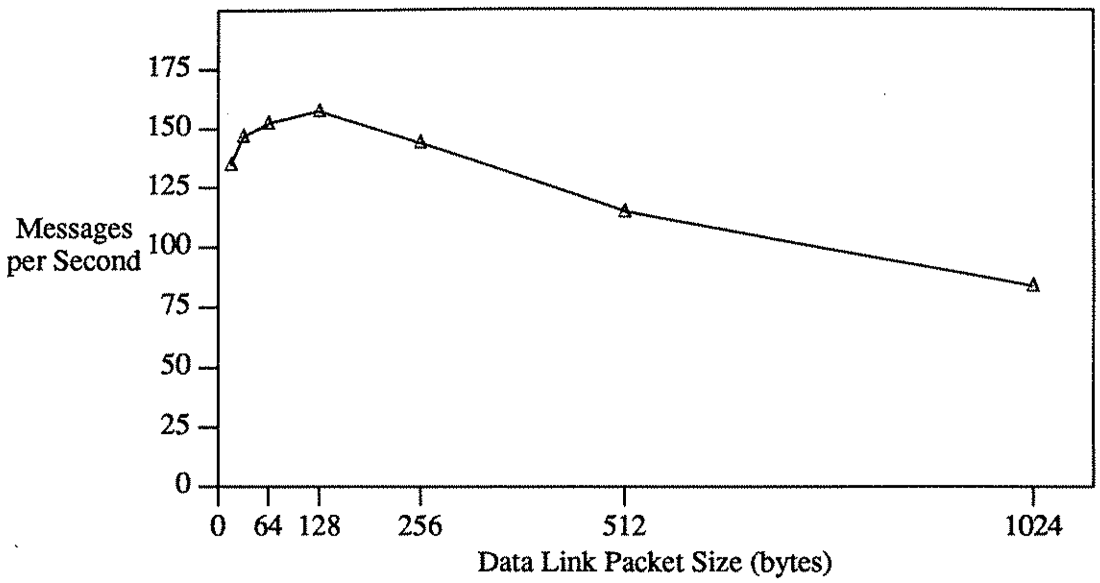

Figure 5 - Number of Messages per Second at Data Link Layer

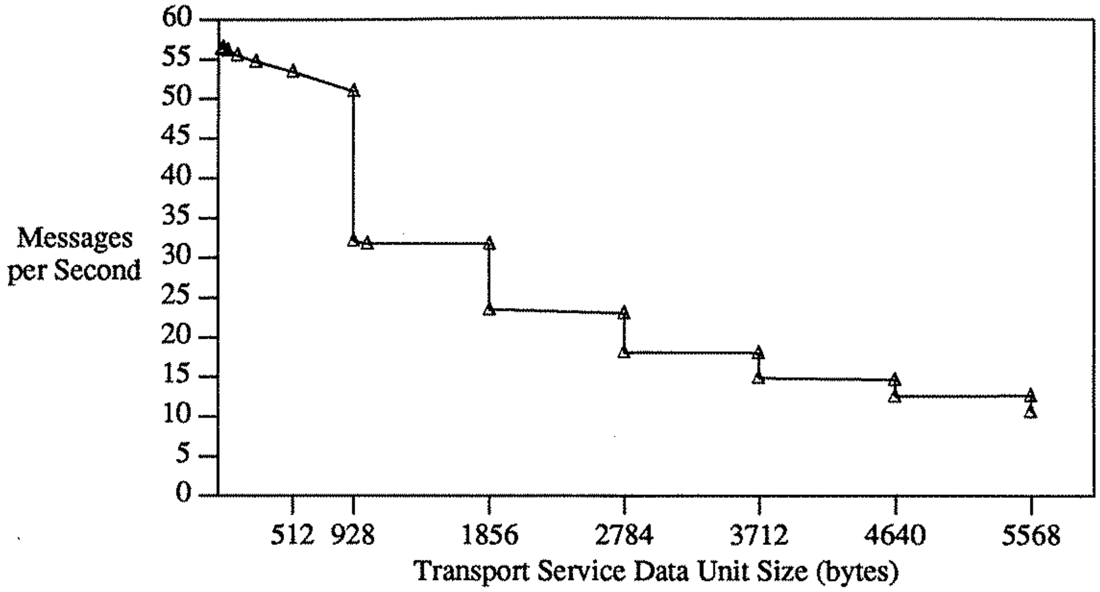

Figure 6 - Number of Messages per Second at Transport Layer

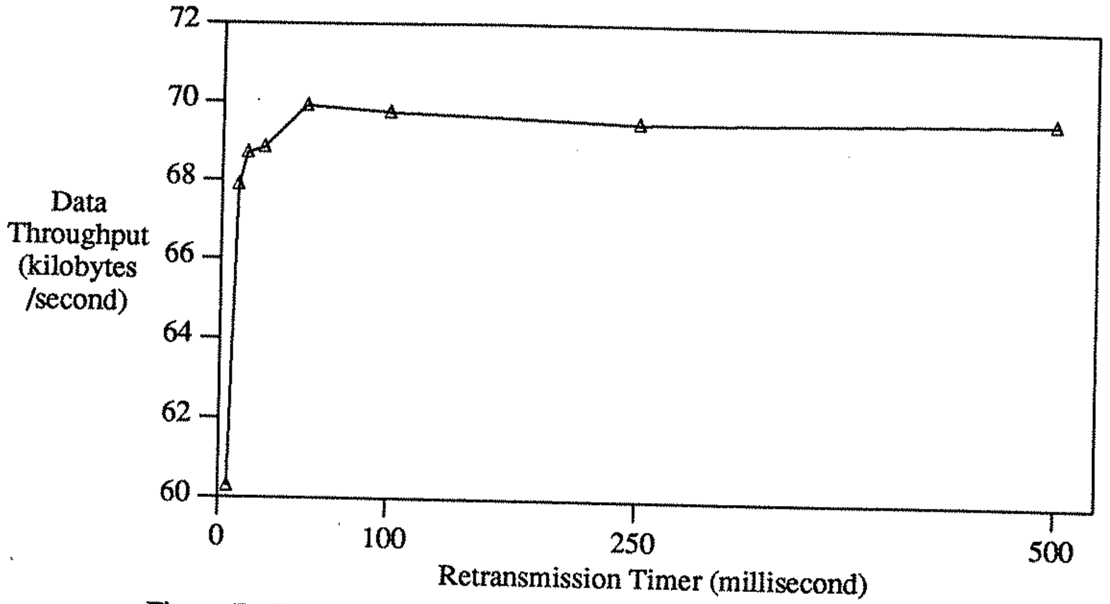

Figure 7 - Transport Throughput vs. Setting of the Retransmission Timer

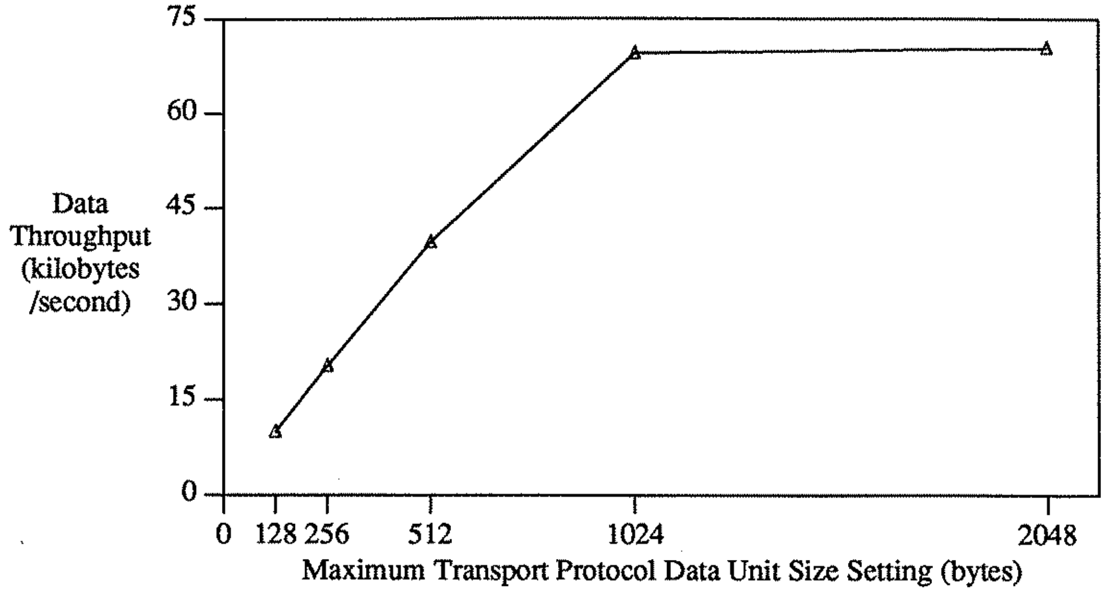

Figure 8 - Transport Throughput vs. Maximum Transport Protocol Data Unit Size

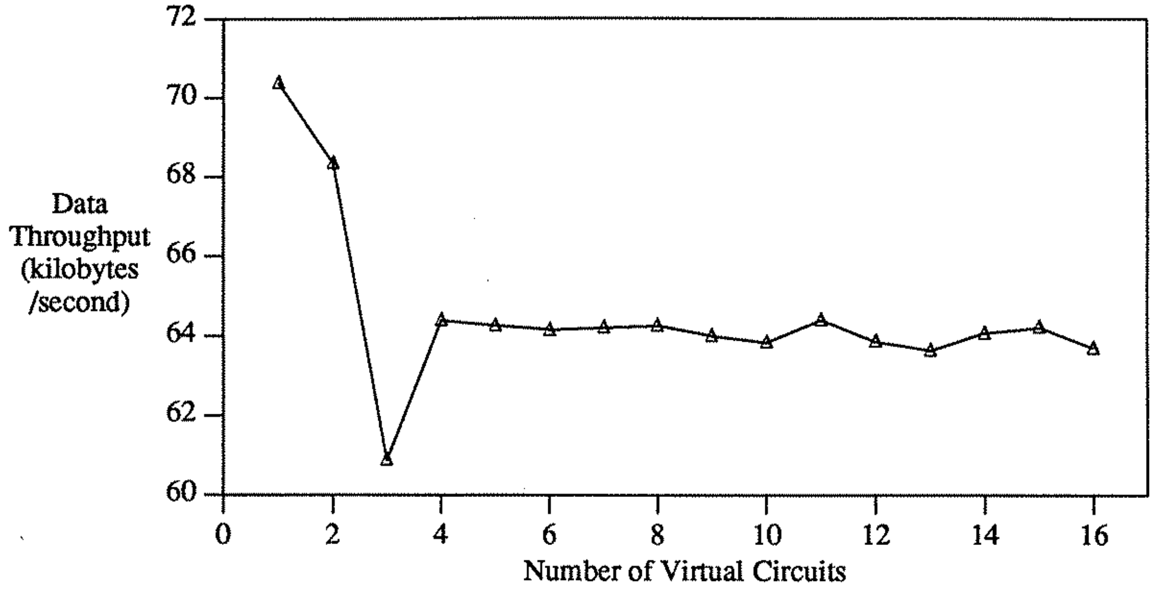

Figure 9 - Transport Throughput vs. Number of Virtual Circuits

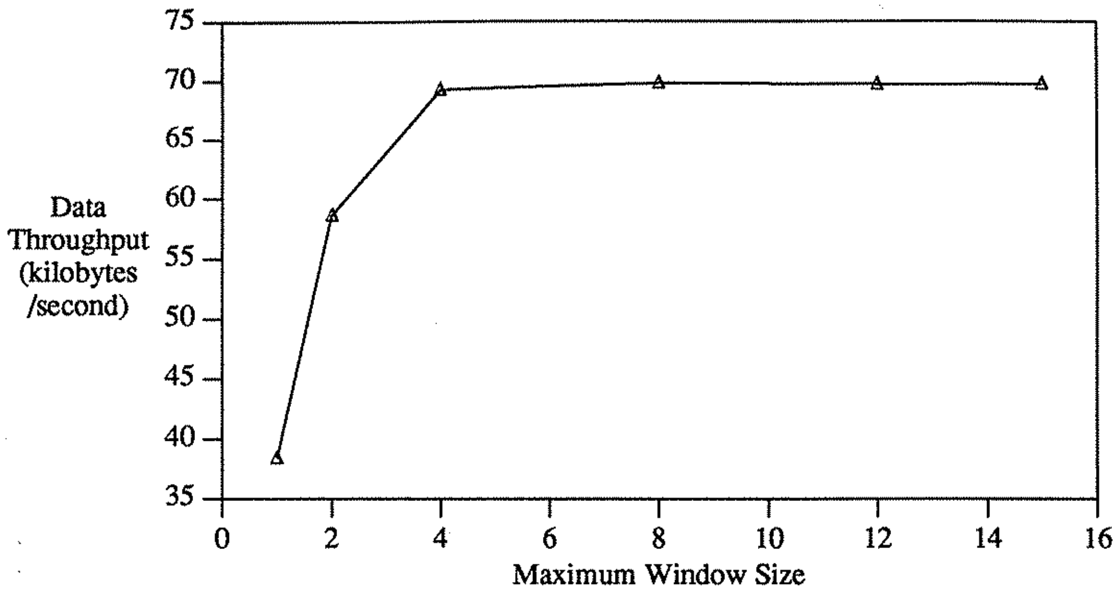

Figure 10 - Transport Throughput vs. Maximum Window Size

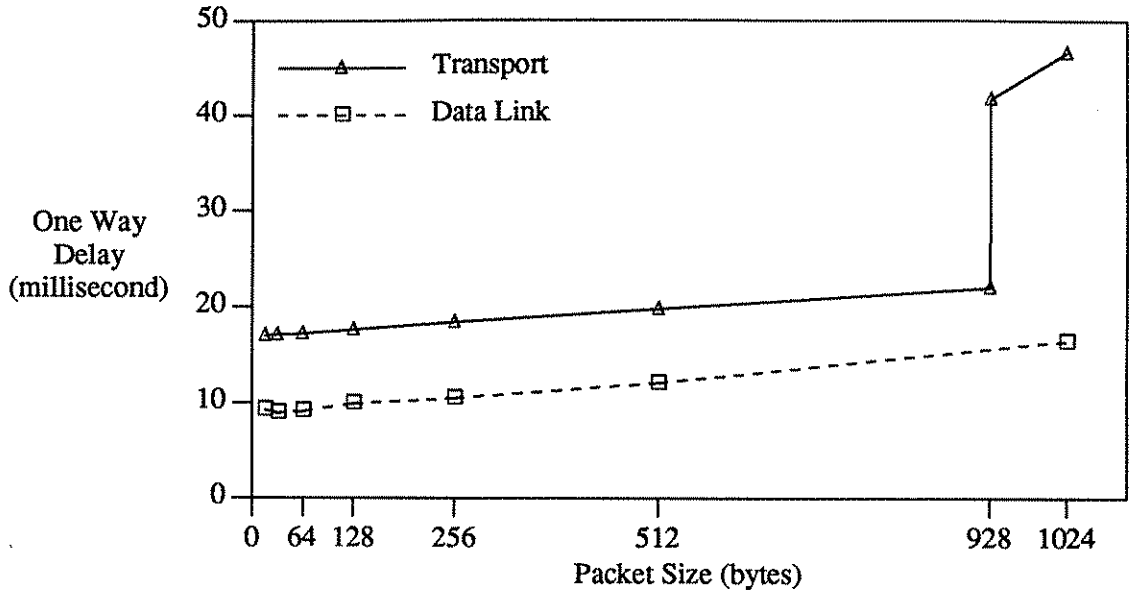

Figure 11 - One Way Transport and Data Link Delay vs. TSDU and Data Link Packet Sizes

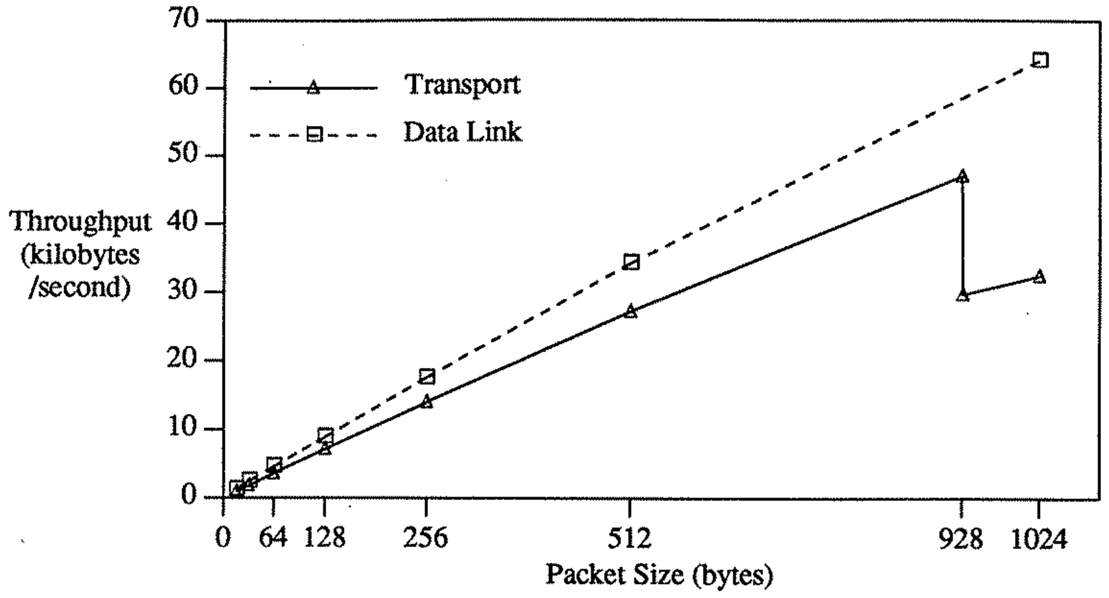

Figure 12 - Transport and Data Link Throughput vs. TSDU and Data Link Packet Sizes

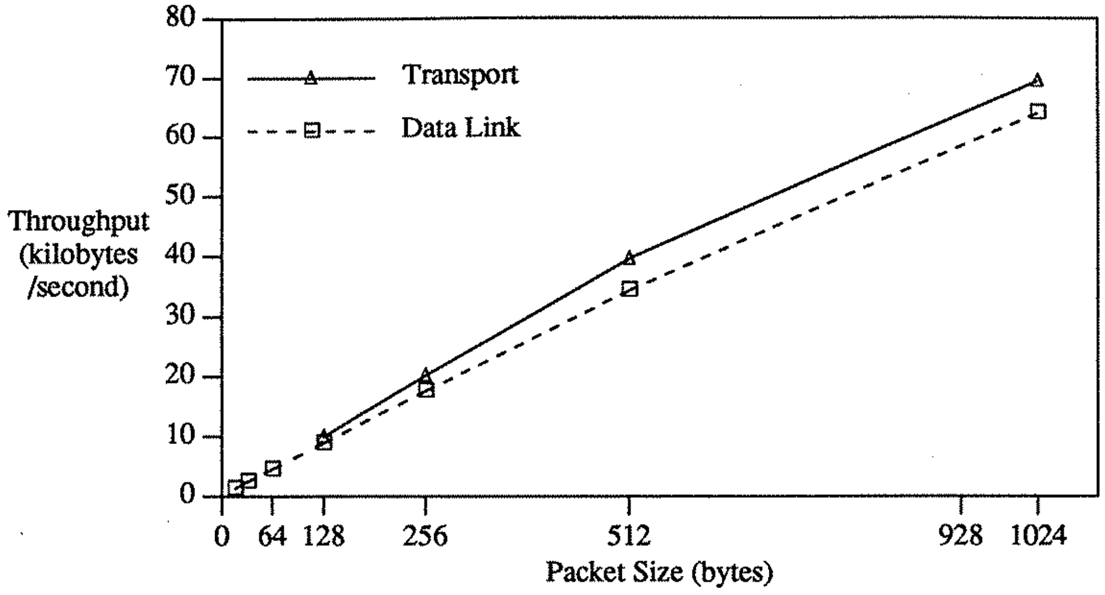

Figure 13 - Transport and Data Link Throughput vs. TPDU and Data Link Packet Sizes

BIBLIOGRAPHY

ELEC86	Electronic Industries Association IE-31 SP 1393A Working Group, *RS-511 Manufacturing Message Service for Bidirectional Transfer of Digitally Encoded Information*, Draft 5, June, 1986.
FRAN86	Franx, C., "The Synchronous Clock System", *Unpublished*, July, 1986.
GENE86	General Motors Manufacturing Automation Protocol Committee, *GM MAP Specification*, version 2.2, 1986.
IEEE84a	The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard 802.1B Station Management*, 1984.
IEEE84b	The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard 802.2 Logical Link Control*, 1984.
IEEE85	The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard 802.4 Token-Passing Bus Access Method and Physical Layer Specifications*, 1985.
INDU85a	Industrial Networking, Inc., "MAP/One Cable Starter Kit *Installation Guide*", 1985.
INDU85b	Industrial Networking, Inc., "MHR-40 Head End Remodulator *Installation Guide*", 1985.
INTE85	Intel, "iRMX86 Reference Manuals", 1985.
INTE86a	Intel, "iNA960 R2.0 Configuration Guide", Final Draft, December 1986.
INTE86b	Intel, "iNA960 R2.0 Installation Guide", Pre-Release, August 1986.
INTE86c	Intel, "iNA960 R2.0 Programmer's Reference Manual", Final Draft, November 1986.
INTE87a	Intel, "MAPNET2.1 User's Guide", July 1987.
INTE87b	Intel, "The Big MAP Attack", Unpublished, March 1987.
ISO7498	International Organization for Standardization, *Draft International Standard 7498*, "Information Processing Systems - Open Systems Interconnection - Basic Reference Model", October 1984.
ISO8072	International Organization for Standardization, *Draft International*

Standard 8072, "Information Processing Systems - Open Systems Interconnection - Transport Service Definition", June 1986.

ISO8073 International Organization for Standardization, *Draft International Standard 8073*, "Information Processing Systems - Open Systems Interconnection - Transport Protocol Specification", July 1986.

ISO8326 International Organization for Standardization, *Draft International Standard 8326*, "Information Processing Systems - Open Systems Interconnection - Basic Connection Oriented Session Service Definition", September 1986.

ISO8327 International Organization for Standardization, *Draft International Standard 8327*, "Information Processing Systems - Open Systems Interconnection - Basic Connection Oriented Session Protocol Specification", September 1986.

ISO8473 International Organization for Standardization, *Draft International Standard 8473*, "Information Processing Systems - Open Systems Interconnection - Data Communications Protocol for Providing the Connectionless-Mode Network Service", March 1986.

ISO8571 International Organization for Standardization, *Draft International Standard 8571*, "Information Processing Systems - Open Systems Interconnection - File Transfer, Access, and Management", August 1986.

ISO8602 International Organization for Standardization, *Draft International Standard 8602*, "Information Processing Systems - Open Systems Interconnection - Protocol to Provide the Connectionless-Mode Transport Service Utilizing the Connectionless-Mode or Connection Oriented Network Service", February 1986.

ISO8649 International Organization for Standardization, *Draft International Standard 8649*, "Information Processing Systems - Open Systems Interconnection - Service Definition for Common Application Service Elements", June 1986.

ISO8822 International Organization for Standardization, *Draft International Standard 8822*, "Information Processing Systems - Open Systems Interconnection - Connection Oriented Presentation Service Definition", July 1986.