PERFORMANCE MEASUREMENT OF DATA TRANSFER SERVICES IN MAP

W. Timothy Strayer
Alfred C. Weaver

Computer Science Report No. TR-88-04
February 9, 1988

This work was supported by Intel Corporation and by the Flight Data Systems Branch of NASA-Johnson Space Center. This paper has been submitted to *IEEE Networks*.

Performance Measurement of Data Transfer Services in MAP

W. Timothy Strayer
Alfred C. Weaver

Computer Networks Laboratory
Department of Computer Science
University of Virginia
Charlottesville, Virginia 22903

ABSTRACT

The manufacturing community has adopted the Manufacturing Automation Protocol (MAP) in an attempt to foster standardization and interoperability, but is now justifiably concerned about the responsiveness of MAP-specified protocols when used in time-critical applications. One result of this concern is the emergence of the Enhanced Performance Architecture which provides the application process with direct access to a fast datalink datagram service rather than the more robust virtual circuit service provided by full MAP. We explain the functional differences between the two service types and why both are useful in the manufacturing environment. We show that message segmentation makes a substantial contribution to overall message latency. We illustrate how the retransmission timer setting and the maximum credit window size affect performance. We report throughput and end-to-end delay for one implementation of MAP and draw conclusions about the expected performance cost of messages sent as datagrams versus messages sent over virtual circuits.

1. MAP and the Manufacturing Environment

Modern manufacturing environments represent a special challenge for in-plant communication and control. Historically, the various vendors of process controllers, programmable controllers, and numerical controllers developed their own proprietary hardware and software for interconnecting their controllers with their controlled devices. The predictable result was that there was very little communication across vendor boundaries unless the purchaser financed custom interfaces. In an attempt to address the problem of interoperability, the manufacturing community has recently embraced the Manufacturing Automation Protocol, or MAP [GENE86].

Rather than inventing yet another set of communications protocols, MAP is a collection of existing international standards based on the International Organization for Standardization's Open System Interconnection (ISO OSI) reference model (ISO 7498). The reference model defines a seven layer hierarchy of functions and protocols as shown in Table 1. Each layer in the hierarchy builds upon the services supplied from the layer below, enhancing them to provide even more robust services to the layers above. Each layer in the reference model is specified by a protocol whose services are expected to be of value to the manufacturing industry.

Whether or not implementations of the MAP-specified protocols can operate with the responsiveness required for real-time control is a very real concern among potential MAP users. While the functions shown in Table 1 are all generally useful, not all types of applications require the full functionality of all seven layers. Some types of applications are best served by highly reliable communications, whereas some others may need high speed at the sacrifice of reliability. At Philips in Eindhoven, the Netherlands, discrete assembly line factories are

OSI Layer Number and Name	Function Performed	Protocol Specified by MAP version 2.1
7 — Application	Provides application-specific services to the application process	ISO 8649, 8650 ISO 8571
6 — Presentation	Negotiates an acceptable transfer syntax between heterogeneous hosts	(not used)
5 — Session	Open, closes, and aborts connections	ISO 8326, 8327
4 — Transport	Establishes connections; detects and retransmits lost or corrupted messages	ISO 8072, 8073
3 — Network	Resolves addressing across multiple networks	ISO 8348, 8473
2 — Datalink	Frames data; computes and checks CRC; controls access to the physical medium	IEEE 802.2/ISO 8802.2 IEEE 802.4/ISO 8802.4
1 — Physical	Provides physical signaling on the medium	IEEE 802.4/ISO 8802.4

Table 1.
OSI and MAP 2.1 Functions and Protocols

automated based on a hierarchical model of control [FRAN86]. Lower layers in the hierarchy demand more stringent time restraints on communications; for example, device controllers communicate with very small (less than 20 byte) messages in control loops with a repetitive frequency of up to once every 10 ms. In recognition of this, the MAP community has sanctioned the idea of using less than a full protocol stack by eliminating some of the layers (and their associated functionality) in return for improved, although less reliable, performance. The recent addition of the Enhanced Performance Architecture (EPA) provides only the physical and datalink layer services to the application process in an attempt to improve the speed of communications while accepting a reduction in service provided.

EPA is a subset of the MAP architecture that is designed to provide faster message response times at the sacrifice of upper layer functionality. Whereas a full MAP node would include all seven layers in the protocol stack, EPA bypasses the upper layers, connecting the application processes directly to the datalink layer in an effort to streamline critical communications. It is expected that certain communicating devices within a factory would have both architectures; that is, be able to provide full MAP functionality with the option to provide EPA in special circumstances. A node that provides only MAP is called Full MAP, a node with both is called MAP/EPA, and a node with only EPA is called Mini-MAP. Figure 1 compares these three architectures.

Therefore, given a specific application with specific performance requirements (e.g., throughput, latency, reliability), the user must decide whether his application can "afford" the standardized services of the full protocol stack; if not, then the user must implement an EPA-style interface and provide any missing services in his own application process. As we shall see, the performance characteristics of two layers, the datalink layer and the transport layer, figure prominently in this decision.

2. MAP Functions and Layers

Each layer is unique in its contribution to the robustness of the communications services provided. We describe in more detail the services generally expected of each layer, and more specifically the services provided by MAP and how they contribute to communications in a manufacturing environment.

The physical layer provides the modulation/demodulation necessary for signal transmission and reception. For full MAP, the physical media is a 10 megabit/second broadband bus; EPA specifies a 5 megabit/second carrierband bus. The datalink layer formats data frames and operates the protocol state machine for medium access control. On its transmit side it frames the user's data, adds source and destination addresses, and appends a cyclic redundancy code (CRC) for each transmitted message; on its receiver side it identifies messages addressed to its device, copies them to receive buffers, and checks the message's framing and CRC, simply discarding any messages received with errors. In MAP 2.1, the combined physical and datalink layers provide a basic *datagram* service. Like a postal service, datagram systems use their best efforts to deliver a message from source to destination, but delivery is neither guaranteed nor acknowledged.

Some manufacturing subsystems need only a basic datagram service. An example is a polling-based system which periodically reads a sensor value. If the data is non-critical, then the loss of an occasional message is not important because the sensor value will be updated automatically on the next read cycle. In return for this less-than-perfect service, the user should expect to see some performance benefits, such as lower end-to-end latency and reduced network loading. EPA is expected to be just such a basic datagram service.

The network layer provides addressing services across multiple networks or network segments. Since real-time control systems tend to be implemented on single segment LANs, the network layer is often inactive. In MAP, single segment networks can operate using the Inactive Network Layer Protocol (INLP) which merely places a special INLP code in the message header field reserved for the network layer. When using INLP, the performance impact of the network layer is negligible.

The transport layer provides a more robust connection-oriented service. By means of error detection and recovery the "best effort" datagram service of the lower layers is converted into a highly reliable virtual circuit service, giving the transport user the illusion that a direct, wire-like path connects transport entities. Unlike datagrams, messages sent over a virtual circuit are guaranteed to arrive, and they arrive in the order sent and without duplication.

In converting from datagrams to virtual circuits, however, transport must perform a great deal of protocol processing. Before messages can be sent, the transport layer must first set up a successful transport connection with its peer transport layer in the receiver. After connection establishment, messages arrive for transmission as arbitrarily large units of information called Transport Service Data Units (TSDUs) as shown in Figure 2. Since the transport layer actually works with *packets* called Transport Protocol Data Units (TPDUs), which have an implementation-dependent maximum size, the transport layer divides one TSDU into as many TPDUs as necessary, labeling each with a sequence number. Then the transport layer sends each TPDU to the network layer below, starts a retransmission timer, and awaits a special acknowledgement message from the transport layer in the intended destination. When that acknowledgement is received the next TPDU is released for transmission. However, both messages and acknowledgements can be lost for many reasons, e.g., addressing errors, bit errors during transmission, or buffer congestion at the receiver. If the retransmission timer expires before the acknowledgement is received, then either the TPDU or its acknowledgement was lost en route, so a duplicate packet is sent with the same sequence number. If the destination now receives this message for the first time, it is acknowledged in the normal way; if the destination identifies it as a duplicate from its sequence number, it discards it but acknowledges it again. In either case the protocol is robust, retransmitting lost or corrupted messages and acknowledgements until they are finally delivered.

If the processing and/or propagation delays are long relative to message transmission time (as is normally the case), then sending one message and waiting for its acknowledgement (a *stop-and-wait protocol*) is inefficient. Instead, a *sliding-window protocol* is used in which the receiver indicates by means of a *credit window* how many messages it is willing to accept. A receiver can thus *throttle* a sender by reducing its window size on a given connection, consequently reducing the number of outstanding (i.e., unacknowledged) messages currently in the pipeline.

In manufacturing, this more robust connection-oriented service should be used for critical applications requiring acknowledged, in-order delivery, such as file transfer, program download, or emergency messages. Of course these additional services are "purchased" at a cost: increased protocol processing, increased end-to-end latency, and increased network traffic due to acknowledgements and retransmissions.

The session layer builds upon this reliable virtual circuit service to provide session users with a facility for dialogue control. No major additional data transfer primitives are provided here, so its impact on performance is due solely to maintaining a dialogue state machine and not on providing additional reliability. Furthermore, the presentation layer is null in MAP 2.1, so its performance impact is also negligible.

The application layer primitives provide the only normal access to the communications services prescribed by MAP. There is a group of frequently used primitives, called the Common Application Service Elements (CASE), which use the robustness of the full protocol stack to relieve the application process of all concerns of how the data is addressed and delivered. There are also sets of primitives which are tailored for specific applications. These collections are called Specific Application Service Elements (SASE) and include Manufacturing Message Service (MMS) and File Transfer, Access and Management (FTAM). These are complex protocols which provide extensive user services and whose performance is dependent on intangibles such as the type of host file system; we do not treat them here.

Thus the major data transfer enhancements are provided at two layers: the datalink layer, where media access control primatives are used to allow groups of bits (frames) to be transferred as single units called datagrams, and the transport layer, where this "best effort" datagram service is converted into a highly reliable virtual circuit service. We therefore examine these two layers to expose fundamental performance characteristics inherent in the two basic service types.

3. Our MAP Network

Our experimental MAP network [INTE87] consisted of two MAP stations, each with front-end communications processors which implemented the protocol processing, a very short (3 meter) IEEE 802.4 token bus, and the associated cables, taps, and headend remodulator as shown in Figure 3. In all of our experiments, one station was the transmitter and the other was the receiver; thus there were never significant network access delays (waiting for the token) or propagation delays. With only one transmitter, network offered load was always low, never exceeding 10% of the physical network's 10 megabit/second capacity. Our goal was not to measure the network's utilization or efficiency, but rather to characterize the performance characteristics which a user program would see when it utilized the services of the transport and datalink layers.

In the context of manufacturing, we think the two most important performance indications are the throughput and end-to-end latency which can be achieved between two stations; these measurements are reported for both the virtual circuit and datagram service types. We also illustrate the role of two network parameters which the user can control (retransmission timer setting and connection window size) and show their impact on the performance of a real-time system.

4. Performance Measurements

Unlike analytic models, performance measurement requires that there be a physical implementation to instrument. Thus measurements are unavoidably tied to a particular hardware and software implementation. Since advances in technology, increases in CPU speed, and additional experience in writing protocols and building communications systems will all cause performance to improve over time, we do not focus on the absolute performance reported, but rather on the larger issues such as the relative difference between datagrams and virtual circuits.

4.1 One Way Delay

One way delay is the measure of the elapsed time (latency) between message transmission and receipt. We measured this delay by timestamping each packet as it was transmitted, and comparing that timestamp to the time of receipt. We paused between transmitting packets so that all queues were empty and all system resources were available for processing that packet. These delays are representative of those encountered in polling or command/response communications, not bulk data transfer.

Figure 4 shows the one way delay for packets at the datalink layer. For a packet of size *L* bytes, end-to-end delay for a datagram was about $(7.8 + \frac{L}{215})$ ms. The initial cost-of-use delay of 7.8 ms included submission of the packet to the operating system for transmission (observed to require no less than 4.0 ms) and protocol header processing.

Since TSDUs can be arbitrarily large, transport must segment them into one or more TPDUs. As shown in Figure 5, the minimum transport delay was about 18.0 ms and increased linearly until one TSDU could no longer fit into one TPDU. Maximum TPDU size is implementation-dependent; in our system it was 1,024 bytes. After reserving space within the

TPDU for protocol headers, the space available for the information field was 962 bytes. When TSDU size first exceeded 962 bytes, the TSDU was segmented into two TPDUs; this initial segmentation required approximately 23 ms. As TSDU size continued to increase, further segmentations were required each time TSDU size exceeded a multiple of 962 bytes; each additional segmentation required 5 to 10 ms.

We observe that short packets sent over a transport connection suffer about twice the delay of the equivalent size datagram. This increase is exclusively due to the additional protocol processing involved. As packet size increases beyond the segmentation limit, the segmentation process accounts for the majority of the total delay. We note that datalink does not have an equivalent segmentation process. Datalink packets are restricted to some maximum length (1,024 bytes for our system; 8,000 bytes for MAP in general) and the layers above must honor this limit.

4.2 Throughput

Throughput is defined as the rate of data delivery in bytes per second. This metric is a good indication of the efficiency of bulk data transfer operations (e.g., file copy). For both datalink and transport layers, we transmitted packets continuously until 10 megabytes had been transferred and then recorded the average transfer rate. At the datalink layer, throughput for packets of length L bytes was $77L$ bytes/second (see Figure 6). Note that datalink packet size L was bounded (in our system at 1,024 bytes), which resulted in an absolute limit on the throughput which datalink can achieve.

Figure 7 shows that segmentation is again an issue for the transport layer. As TSDU size increased, throughput increased steadily until the first segmentation. The drop in throughput around 962 bytes is directly attributable to the processing time required to divide one TSDU into two TPDUs. Further segmentations, with a resulting loss of throughput, occur as TSDU size exceeds multiples of 962 bytes. The throughput curve eventually converges to an asymptote of about 91 kilobytes/second. Convergence is explained by observing that any efficiency gains realized from sending larger messages were eventually countered by the penalty of segmenting the larger message.

4.3 Retransmission Timer

The transport layer retransmission timer can be set by an interface providing access to the intra-layer configuration parameters (ISO calls this "station management"); this value controls the amount of time the transmitter will wait for an acknowledgement before sending a duplicate. The manufacturing user must be able to set an optimum value for the retransmission timer. If the value is too small, needless retransmissions occur, resulting in decreased throughput and wasted protocol processing. If the value is too large, the time to detect a lost message increases, thereby decreasing the responsiveness of any real-time control loops.

As our retransmission timer value was decreased from an initial setting of 100 ms, as shown in Figure 8, throughput was unaffected by values of 60 ms and larger, indicating that in this range there were no retransmissions. Throughput dropped dramatically, however, for values below 60 ms because duplicate TPDUs were being sent.

To understand why this particular setting of the retransmission timer is the smallest value without diminished service, refer to Figure 5 which showed one way delay for the transport layer. In our system, the delay for the largest message not requiring segmentation was about 23 ms; acknowledgements, which are short messages, require about 18 ms; message processing, internal latencies, and acknowledgement turnaround require another 15-20 ms. Thus a retransmission timer setting of less than 60 ms resulted in increased retransmissions and decreased throughput, as confirmed by Figure 8.

Each MAP installation must measure these delays before an appropriate retransmission timer setting can be chosen.

4.4 Virtual Circuits and Throughput

We varied the number of virtual circuits, or connections, from 1 to 16 to observe its effect on throughput. One virtual circuit provided the best throughput (about 91 kilobytes/second) because it required the least overhead. Even though multiple virtual circuits between two stations provided additional avenues for the transfer of data, all virtual circuits used the same physical connection and thus overall throughput was not enhanced. In fact, each additional virtual circuit levied a maintenance overhead penalty, such that the total throughput for four or more connections reduced throughput to about 64 kilobytes/second.

4.5 Window Size

Recall that in the transport layer, a window is an ordering of active sequence numbers. 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 can thus throttle the transmitter by reducing its credit. In our system the maximum window size could be set from 1 to 15, with 15 being the default.

As the receiver's buffer space was filled with incoming TPDUs, it decreased the size of the credit which it returned with each acknowledgement. The credit told the transmitter how many more TPDUs the receiver was prepared to handle and varied from 0 to the maximum window size. As the maximum window size decreased, the maximum allowable credit was likewise decreased. As the credit decreased, fewer TPDUs were in the pipeline and throughput decreased. Figure 9 shows that, in our system, there was an immediate effect of decreased throughput due to the decrease in the window size. This indicated that the window size was in fact throttling the transmitter by limiting the number of outstanding TPDUs allowed in the pipeline. Dropping the window size from 8 to 1 cut throughput in half on our network.

Each MAP system must individually measure the tradeoff between buffer space (i.e., maximum window size) and throughput to achieve optimal performance.

5. Conclusions

While we recognize that the experimental results we present here are based on one vendor's interpretation and implementation of the MAP standards, we believe that the conclusions we shall draw are valid by virtue of the inherent characteristics of the protocols studied and not just the actual numbers this implementation produces. For instance, the delays suffered by a message at the Transport Layer will necessarily be affected by the number of segmentations that have to be performed on that message. The setting of the retransmission timer is dependent on the expected round-trip delays (one-way for the TPDU, internal latencies, and return of the acknowledgement), and once those numbers are known, their sum provides the lower bound on the setting of the retransmission timer. We also note that the setting of the maximum window size is an effective throttle of data flow.

Furthermore, we are able to draw some general conclusions about communications products. The vendors who will be producing MAP products are given a wide degree of freedom with respect to interpretation and implementation of the protocols. Some vendors are convinced that all services should be handled by the same processor. Some lean toward the front-end processor architecture. Others are trying to put the whole protocol stack on a chip-set. The setup we used was a front-end processor which off-loaded the task of communications from the host processor. We make the following observations.

There are both benefits and drawbacks to using a front-end processor to provide communications services. Having these services provided by a front-end processor allowed concurrency; the front-end and host processors ran in parallel and interacted via the Multibus backplane. Thus the host processor was relieved of the concerns of servicing messages as they arrived asynchronously from the physical network, and could devote more computing time to the user application.

However, the only means of accessing the front-end processor and its communications software was through a standard interface across the Multibus, which presented a bottleneck. Accessing the communications software through the datalink interface placed restrictions on the message size that could be transmitted by the user application. Large messages had to be buffered in the host's memory and delivered to the datalink layer in smaller (1,024 byte) segments. Therefore many small messages were sent through the Multibus to the front-end processor, which subsequently limited throughput.

By accessing the transport layer, large messages were delivered to the front-end processor and stored there until they could be processed. The communications software and the underlying data link hardware worked most efficiently while there was data in buffers on the front-end processor. When transport 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 Multibus interface.

Manufacturing makes at least two demands on communications services. A highly reliable service for data-critical communications is essential for transferring bulk data, like data files or control programs. In such a service, data rate is traded for data integrity. The robustness of the transport layer data transfer primitives, namely the virtual circuit service, is required. However, this service may not be adequate for time-critical communications, such as command/response or polling. Given that data errors are rare on a properly tuned MAP network, emphasis on speed supercedes that of error recovery for this type of service. Datalink layer datagram service provides basic data transfer primitives without the overhead of perfect reliability.

As seen in Figure 10, datalink provided a much quicker end-to-end service than did transport. The difference in the two curves is due to the additional overhead associated with the increased functionality of the transport layer. Included in this additional overhead are such functions as sequencing TPDUs, buffering unacknowledged TPDUs, running timers and retransmitting TPDUs when necessary. Thus the difference between the two curves is the price paid for guaranteed delivery. However, for packets within data link's size restriction (1,024 bytes and less), datalink provides a service more suited to the command/response or polling type communications than does the more complex transport service.

Because of the simpler service datalink provided, it is still not surprising for datalink to provided better throughput than transport for packet sizes of 1,024 bytes and less, as seen in Figure 11. However, datalink cannot provide the function of segmentation, and is therefore restricted to a fixed maximum packet size. Figure 11 plainly shows the performance penalty with transport's extra services.

Full MAP provides communication primitives to application programs through the application layer. CASE primitives provide the application programs with common transfer services. Other SASEs like FTAM and MMS use the full stack to ensure reliability. These services are all based on the virtual circuit service first introduced in the transport layer. But since these services rely on such a complex service type, they share its the drawbacks.

For now the manufacturing user must choose the service types best fitted for his applications. MAP is a good idea, and as long as it satisfies the needs of the manufacturing community, it will greatly influence the direction of automated manufacturing for many years. But EPA was an afterthought for MAP, and at best it can only be considered a patch. We hope there will come a time when EPA can be eliminated, and the real-time needs of the manufacturing community will be serviced by a real-time transport protocol, combining the reliability of the conventional transport layer with the time-critical performance of the datalink layer.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the technical support and financial assistance of Steve Alten and Matt Katzer at Intel Corporation and William Reed and Virginia Whitelaw at the NASA-Johnson Space Center.

LIST OF FIGURES

BIBLIOGRAPHY

FRAN86	Franx, Cornelius, and Kevin Mills, Proceedings of the *Workshop on Factory Communications*, National Bureau of Standards, March 17-18, 1987, pp 51-72.
GENE86	General Motors Manufacturing Automation Protocol Committee, *GM MAP Specification*, version 2.2, 1986.
INTE87	Intel, "MAPNET2.1 User's Guide", July 1987.

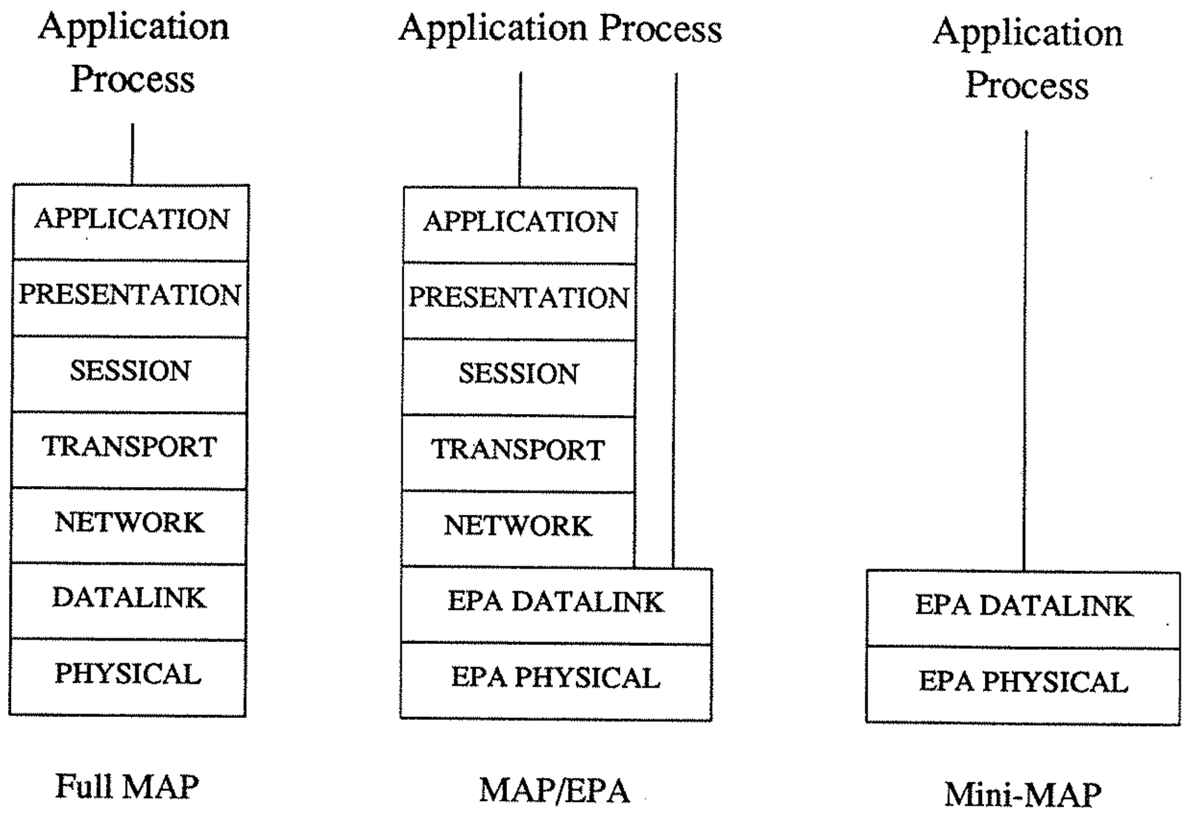
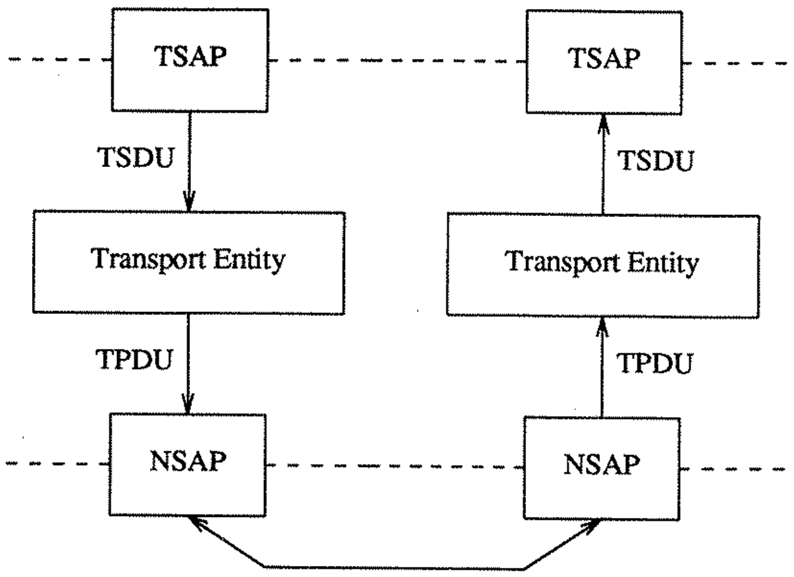
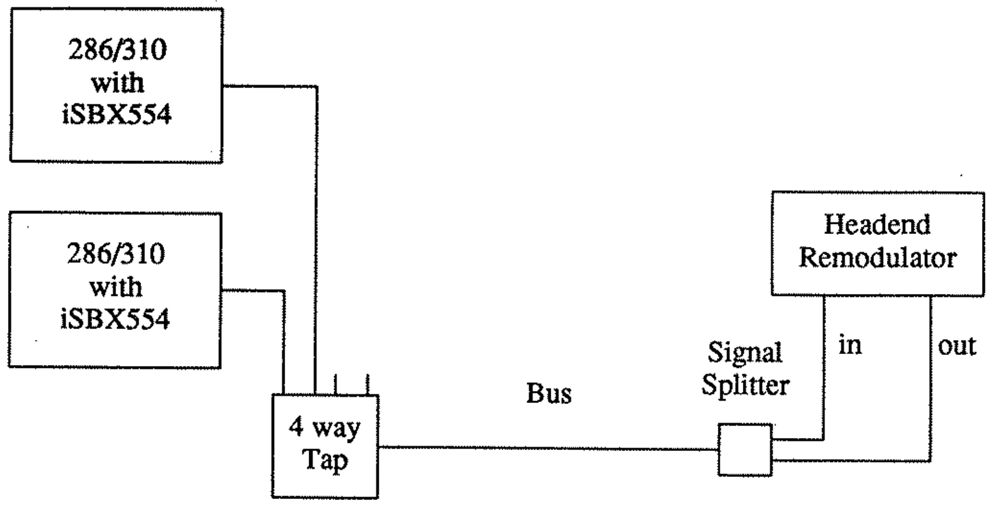


Figure 1 - Comparison of Full MAP, MAP/EPA and Mini-MAP



TSAP — Transport Service Access Point
NSAP — Network Service Access Point
TSDU — Transport Service Data Unit
TPDU — Transport Protocol Data Unit

Figure 2 - Model of the Transport Layer



Host: Intel 286/310s
- 80286 CPU 6.0 MHz
- 1 Megabyte RAM

Front-End Processor: iSBX554 Token Bus COMMengine
- INI Token Bus Modem
- 256 Kilobytes onboard RAM
- 80186 CPU 6.0 MHz

INI Headend Remodulator with Cable Kit

Figure 3 - Our MAP Token Bus Network

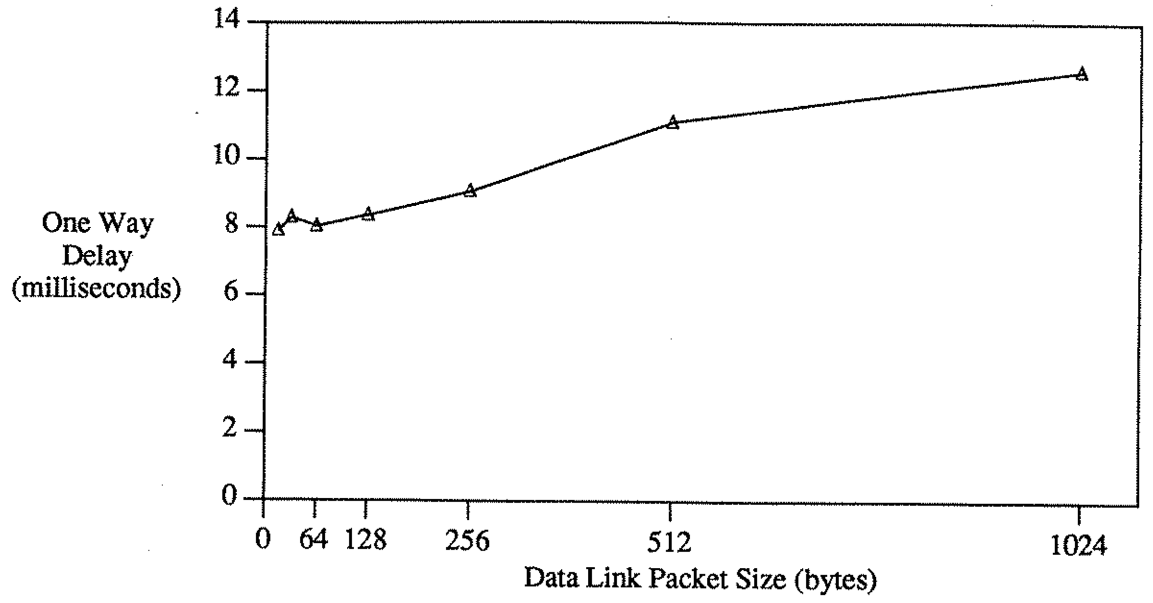


Figure 4 - One Way Data Link Delay vs. Data Link Packet Size

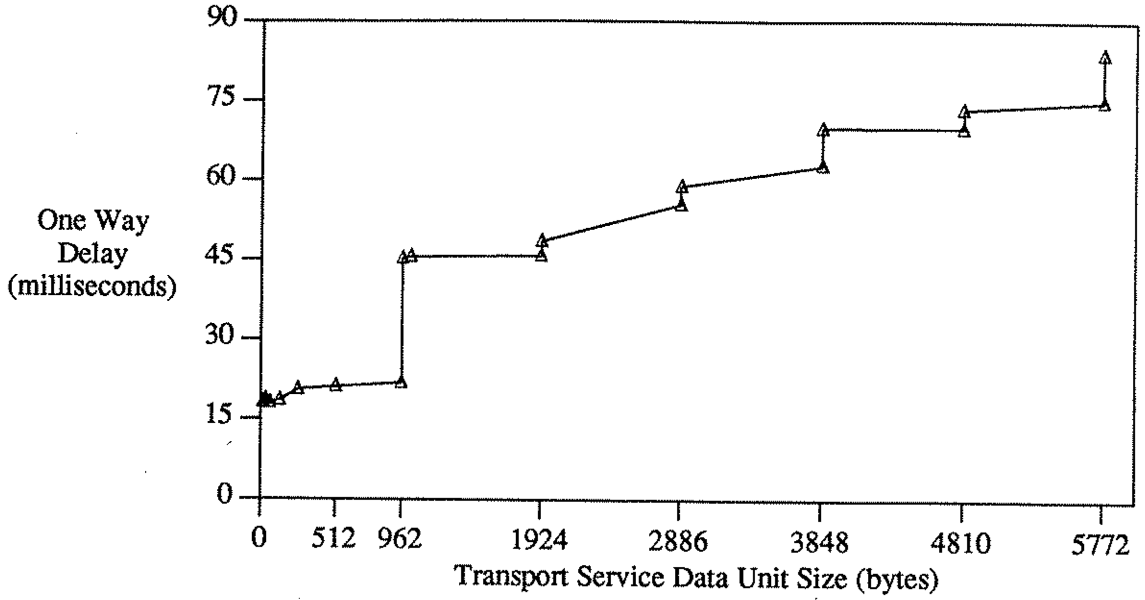


Figure 5 - One Way Transport Delay vs. Transport Service Data Unit Sizes

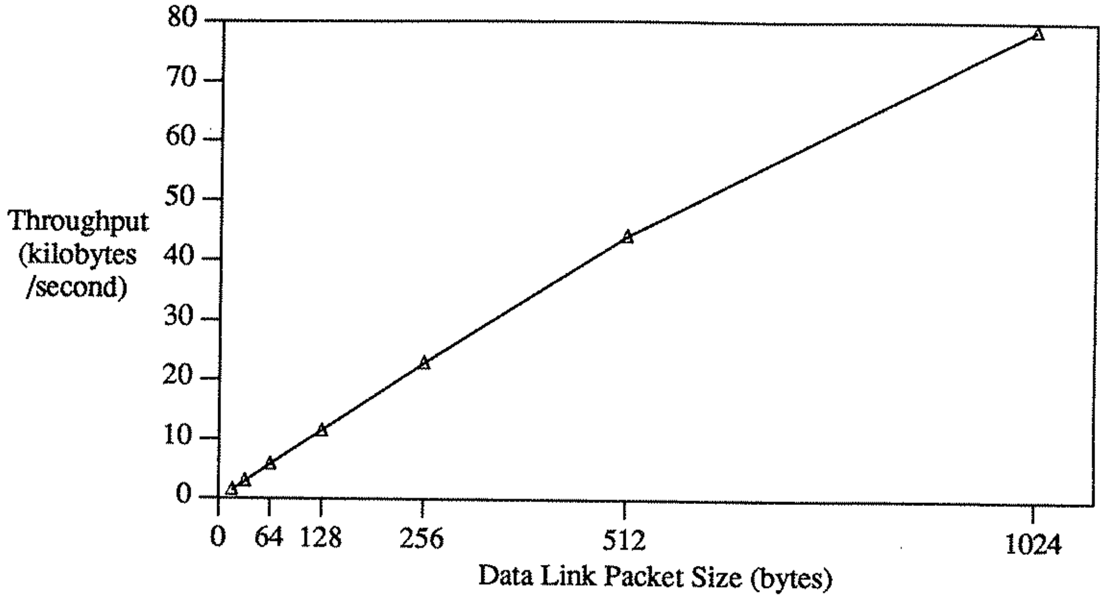


Figure 6 - Data Link Throughput vs. Data Link Packet Size

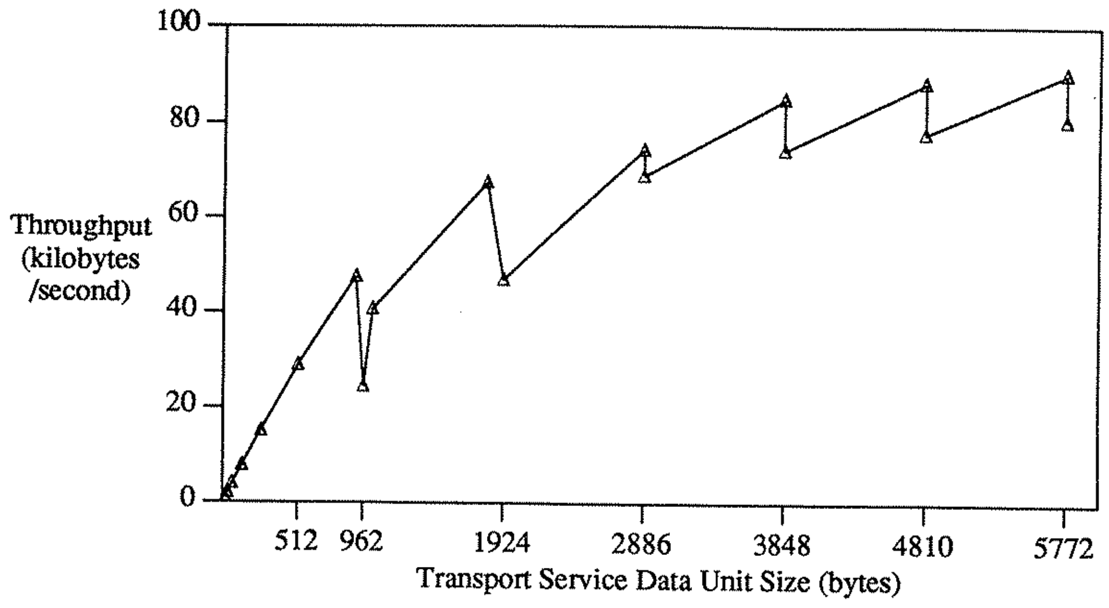


Figure 7 - Transport Throughput vs. Transport Service Data Unit Size

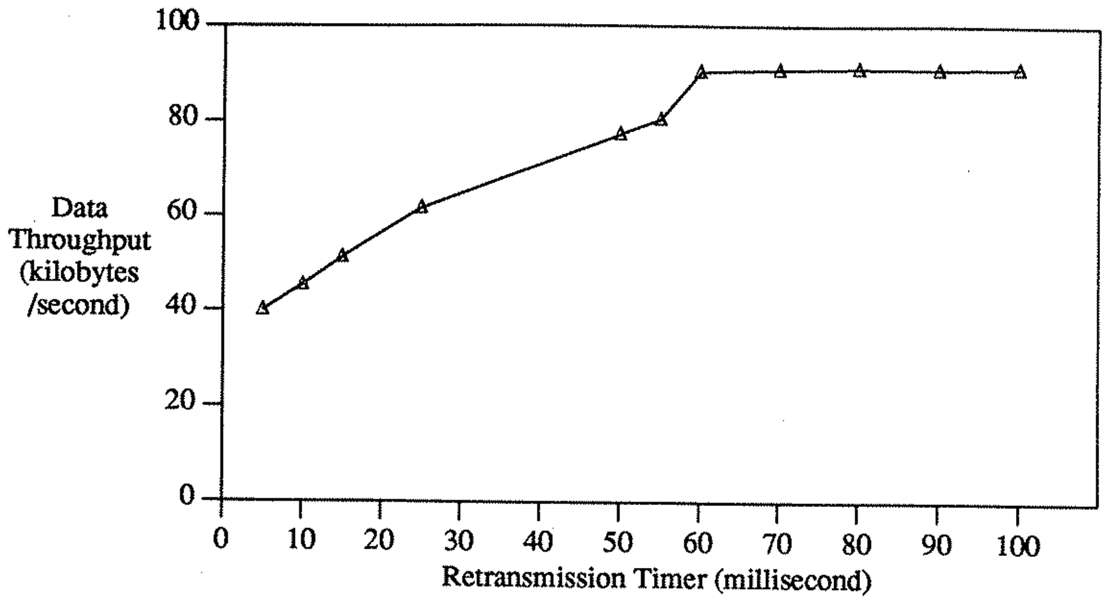


Figure 8 - Transport Throughput vs. Setting of the Retransmission Timer

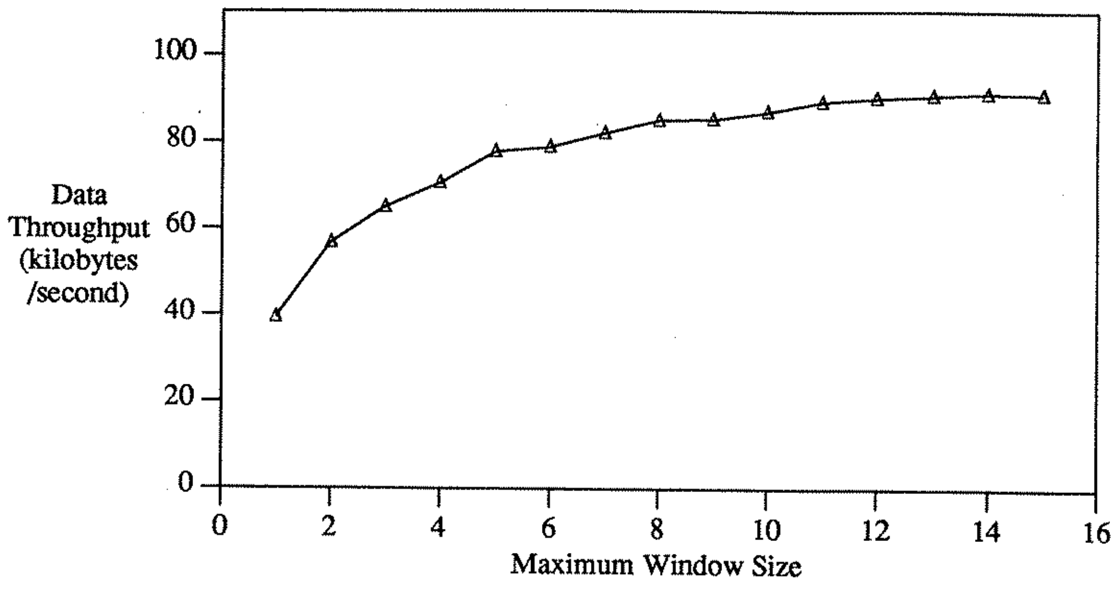


Figure 9 - Transport Throughput vs. Maximum Window Size

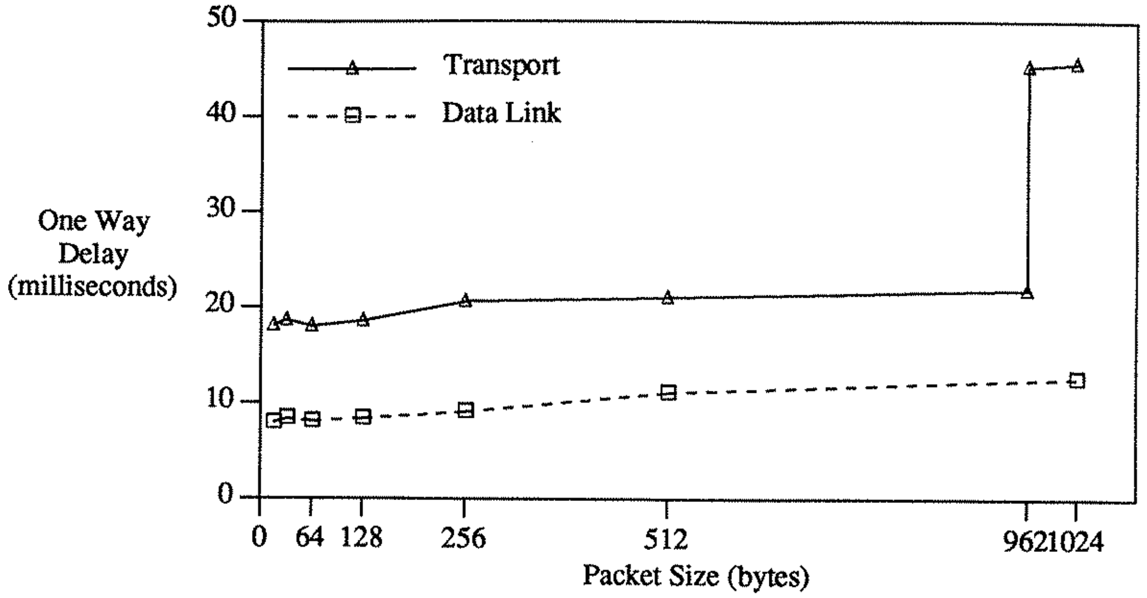


Figure 10 - One Way Transport and Data Link Delay vs. TSDU and Data Link Packet Sizes

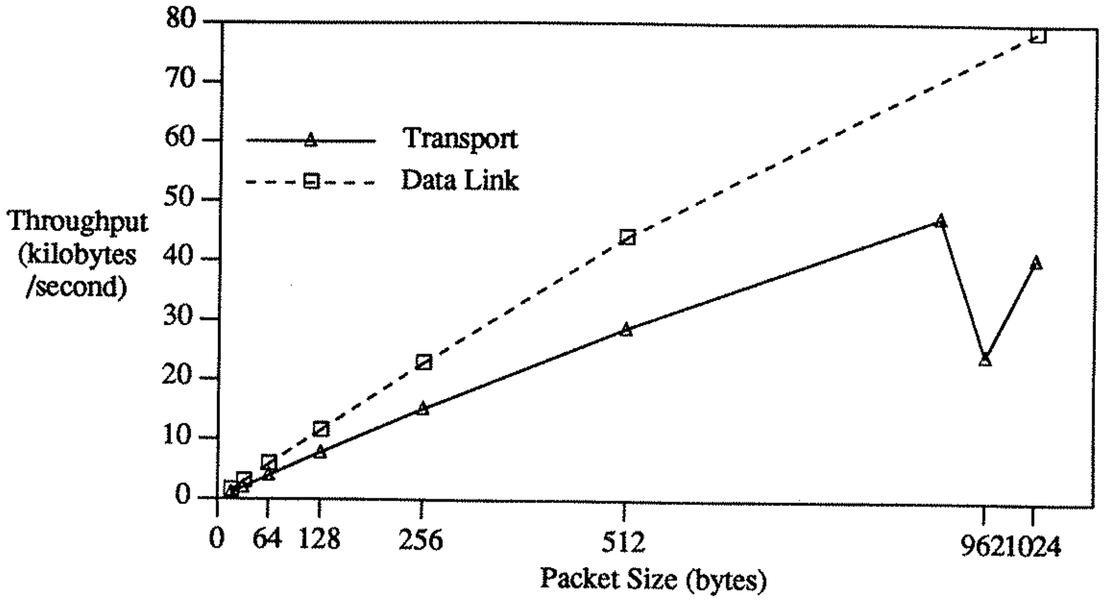


Figure 11 - Transport and Data Link Throughput vs. TSDU and Data Link Packet Sizes