EVALUATION OF REAL-TIME TRANSPORT PROTOCOLS

W. Timothy Strayer
Alfred C. Weaver

Computer Science Report No. TR-88-21
October 10, 1988

Evaluation of Real-Time Transport Protocols

W. Timothy Strayer
Alfred C. Weaver

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

June 1988

ABSTRACT

We first make some observations about the environment in which a real-time communication system is to serve — namely, real-time distributed systems. We identify the principles of real-time communications as we understand them from the archival literature. To create a framework for the comparison of protocols, we briefly discuss the ISO OSI Reference Model and its suitability as a paradigm for real-time systems. Next we identify those functions which we believe are most useful for achieving real-time communications. We evaluate eight specific network protocols and characterize each by identifying the presence or absence of our identified functionalities. Finally, we examine each of the eight protocols in turn and comment on how well each protocol would support a real-time distributed system.

CONTENTS

LIST OF TABLES AND FIGURES

Tables

Figures

1. Purpose

The purpose of this report is three-fold: (1) to identify communication functions which are considered useful within a communication system; (2) to document the extent to which these functions are incorporated in extant protocols; and (3) to identify those communication functions which are potentially useful for a real-time communication system.

To accomplish these goals, section two makes some observations about the environment in which the real-time communication system is to serve — namely, real-time distributed systems. Section three identifies the principles of real-time communication as reported in the archival literature. To provide a framework for the eventual comparison of protocols, section four briefly discusses the ISO OSI Reference Model and its suitability as a paradigm for real-time communications. Section five identifies those functions which are commonly found in transport protocols. In section six we discuss eight specific network protocols and characterize each by identifying the presense or absence of our identified transport functionalities. Section seven evaluates the protocols and comments on how well each would support real-time communication. Our conclusions are in section eight.

2. Real-Time Distributed Systems

A real-time distributed system consists of some number of application programs (processes) which are physically distributed over a multiplicity of processing elements (processors), interconnected by a communications server (e.g., a local area network and its protocols). What makes the system *real-time* is that processes must complete their computations within a known and bounded time. The processes may be periodic or aperiodic (interrupt-driven), and the resources they require may be either local (on the same processor) or remote (on another processor). The purpose of the communications subsystem is to provide

fast, transparent, and reliable access to remote resources without jeopardizing the time constraints of the application programs.

To appreciate the importance of the underlying communications server, consider the range of implementation possibilities. At one end of the spectrum, all processes could be assigned to a single processor. This would minimize the communications delay because all message passing would occur in the memory of the local processor, but would maximize computational delay because parallelism could not exist. At the other end of the spectrum, every process could be assigned to a different processor. Now computational delay is minimized because all modules execute in parallel, but communication delays are maximized because every process needs the communications server for every transaction. Since remote (processor-to-processor) communication can be orders of magnitude more "expensive" (in terms of delay) than process-to-process communication on the same local processor, the design of the underlying communications subsystem is of paramount importance.

In the sections which follow, we identify the functionalities of extant protocols which we think are important for real-time communications. Then we evaluate the protocols themselves to determine their suitability for supporting real-time distributed systems.

3. Principles of Real-Time Communication

In order to make an educated evaluation of existing protocols with respect to supporting real-time communications, we have surveyed the literature for considerations and requirements that are felt to be most important in achieving the desired timing and robustness properties.

A real-time distributed system consists of several applications that are distributed among different processing nodes connected via a common network. The applications may be executing in parallel subject to both precedence and timing constraints. The underlying

network has responsibility to provide the services necessary for a timely response to changes in the application's environment. A network is not real-time; the applications that use the network are. Thus a real-time communication system is a network that can provide sufficient functionality and performance to be useful to applications which require time-constrained communication, allowing the application to effectively handle error conditions without the communication system inducing errors of its own.

There are three major requirements of distributed real-time processing on a local area network environment [ZNAT87]. *Robustness* is the combination of reliability, availability, and dependability requirements, and reflects the degree of system insensitivity to errors and misinformation. *Flexibility* relates to the ease of designing and structuring a network that can support real-time processing. Finally, *timing requirements* are the timing guarantees demanded of the network by any given station attempting to access the channel.

The real-time communication system must be efficient and reliable. It must be performance oriented in order to provide a very short response time while ensuring that network errors are detected and are recovered from without involving the network users. There is a trade off: as network error detection and recovery protocols provide more robust and hence more reliable service, the cost of executing these protocols on each protocol data unit increases, degrading performance. Thus it is desirable for a real-time communication system to implement only the necessary communication services. This is a minimum according to the required quality of service.

A major requirement of real-time applications is fault tolerance [HWAN87]. Generally, real-time applications produce specific network load conditions and traffic patterns; the load is relatively light and the traffic pattern varies slightly from predictable behavior [STOI88]. Errors within the application, however, must be detected, confined, and corrected using fault tolerance

techniques built into the application. These fault tolerance techniques may cause unpredictable sudden bursts of high priority communication, are called *alarms*. An alarm message must not be stopped by flow control. This sudden peak in load due to alarms is called an *alarm avalanche*. Unless the communication system can meet the deadlines of each of these alarms during an alarm avalanche the error condition will worsen, and more faults will be introduced. It is not sufficient for a real-time system to provide adequate performance during normal activity; the system must actively influence the ability of the fault tolerance techniques to confine and correct the faults before others compound. Real-time system designers must anticipate these message bursts, providing sufficiently robust transfer capacity for all network nodes under all load conditions. This requires high throughput and predictable message delays. Simply increasing the transmission bit rate cannot relieve congestion caused by burst traffic [MIRA83].

End-to-end resource allocation is also necessary to guarantee response times or throughput. Typically, this is accomplished with end-to-end flow control using windows and credits. However, this method only guarantees that buffer space will be available when messages arrive. Le Lann, in [LELA85], identifies at least two other types of resources that must be allocated, CPU cycles and I/O capacity. In fact, buffers have become a cheap and readily available resource, whereas most LANs are bound by the CPU and system busses.

Connections allocate resources upon creation. However, this allocation may be too costly to maintain if the connection is scarcely used or if there are many connections being supported. The robustness of a communication system must not adversely effect the cost of maintaining the system, whether that cost is measured in resources or message delay.

In order to be flexible, a real-time communication system must provide *enhanced communication* [MINE87]. An enhanced communication architecture is designed to handle

real-time communication according to different scenarios. Several important scenarios are *point-to-point* communication, such as sampling and command/status; *broadcast* communication, such as redundant control; and *concentration* communication, such as response from subordinates. The communication system must also handle combinations and hybrids of these scenarios.

The most notable and fundamental principle of a real-time communication system is that it must ensure that a message is delivered to its destination before the deadline. Its inability to do so is considered a failure. It is both useless and dangerous to deliver a message past its deadline, as its significance is no longer a factor and its presence may confuse or overwrite data that did meet its deadline. The mechanisms used for meeting deadlines are priorities and multilevel message scheduling.

According to Znati in [ZNAT87] there are four criteria for priority schemes: (1) increasing loads from lower priority classes should not degrade the performance of higher priority classes; (2) a priority scheme should treat each message within a priority class fairly in terms of access time; (3) proper operation and performance of the priority scheme should be insensitive to errors in status information; and (4) priority schemes should keep to a minimum the overhead required to implement them.

Messages handled at all layers of protocols should be scheduled according to specific algorithms [LELA85]. Establishing finite upper bounds for access delay only solves part of the general problem. Finite upper bounds must be guaranteed for all service times at all layers for given conditions of utilization. Also, a cost function must be minimized when these bounds are exceeded. Such cost functions for a real-time system are the number of messages which miss their deadlines or the average or maximum message lateness over a given time interval. Deterministic message scheduling algorithms are needed to enforce the guarantee of finite upper

bounds as well as ensure that the least costly messages are discarded on overload or in abnormal situations.

Priorities are used to associate costs with messages present in waiting queues. Static priorities are generally not appropriate for real-time applications; they are mere approximations of time-dependent priorities. Typically, these approximations are only valid for periodic traffic. For aperiodic traffic, however, time-dependent priorities are needed. These priorities can be the deadline of the message itself. Deadline driven algorithms can break ties between messages with identical or nearly identical deadlines by a hybrid scheduling algorithm which also assigns static priorities and weights to the messages.

In a loosely coupled distributed environment, local clocks must be synchronized. They are needed for enforcing bounds on communication delay, including time spent on waiting for access to the network. A method for ensuring accuracy without degrading the system must be employed. It may also be a requirement for inter-application communication to be synchronized [HWAN87]. This feature is present in some languages (e.g. Ada), or the application may find it necessary to rely directly on the network communication services for support. Acknowledged communication provides synchronization, but it is only as accurate as the end-to-end message delay.

Overall system performance improvement results from an extensive cooperation and synchronization between processes. This depends on an accurate synchronization of clocks within individual hosts in the system. The precise clock synchronization is also important for performance measurement and evaluation, correlation of events, testing communication protocols and diagnosis of failures within the distributed system.

Algorithms for clock synchronization can adopt centralized or decentralized schemes which are classified as follows [GORA87]. Central-dictatoric is a scheme with central controller

which directs the participating processes to synchronize their clocks. Central-democratic is a scheme with a dedicated processor periodically computing the general network time by a voting mechanism with individual controllers. Finally, Decentral-democratic is a scheme whereby all of the time-processors execute an identical procedure to achieve clock synchronization. The central-dictatoric is typically used for special hardware, and is not a general solution for an heterogeneous environment.

Several criteria are used to compare clock synchronization. The necessary accuracy depends strongly on the application and generally dictates the complexity of the algorithm. If the synchronization algorithm depends on communication, what are the costs of such communication? There is a cost of computation incurred by running the algorithm, especially if it is complex. The algorithm must be managed in the event of failures. Also, an algorithm should be flexible in adapting to different configurations.

4. The ISO Communication Paradigm

The ISO OSI Reference Model [ISO7498] defines a seven layer hierarchy for providing functionality in a modular fashion. The layers of the model may be grouped into three categories: (1) the layers providing the basic data transfer service, (2) the layer providing reliable and transparent data transfer service, and (3) the layers which provide additional services based on the reliable transfer service. The following paragraphs reflect these groupings.

The *physical layer* provides a means by which the physical connection can be activated and deactivated. The *datalink layer* provides for and manages the transmissions of individual frames of data. A data frame is a collection of data and control bits delimited at the start and end by special markers. Instead of ensuring that each bit individually arrives at its destination

intact, data frames are handled as units. The *network layer* provides services to convert global address information into routing information so that a message can be delivered from one end-node to another. To do this it must maintain routing tables and/or algorithms, establish and terminate network connections when appropriate, and provide switching services to incoming messages to route them onto the proper outgoing path. If there is only one segment (no routers) then a null network layer header may replace the routing information.

The *transport layer* exists solely to provide reliable and transparent transfer of data between session or transport user entities. It provides this service without regard to the underlying system of subnetworks or their network layer protocols. It relieves the transport user from any concern with how the data is transferred, where it must go to arrive where intended, or the ordering of the data as it arrives.

The purpose of the *session layer* is to enhance the services provided by the transport layer with mechanisms for managing and structuring reliable data transfer. The session layer manages a well-ordered series of events for establishing, maintaining, and closing a dialogue between session users. It provides major and minor checkpoints for rollback to keep the dialogue synchronized. Upon an error or desynchronization, the session layer manages the return to one of these checkpoints (the last place in the dialogue which was mutually agreed to be error-free). The *presentation layer* negotiates a transfer syntax for use by the application layer. This layer may actually cause the representation of the data unit to be changed. The *application layer* is the interface between user programs and the network. It defines the way in which communication functions are made available to a user program and manages the details required to support that communication. User programs access the network through service elements, using these elements in a prescribed manner. The interface hides the implementation details of the rest of the network from the user, providing only those elements necessary for user programs to establish connection, transfer data, and terminate the connection.

The ISO OSI Reference Model was developed in 1978 as a paradigm for network communications systems. Its design, and the ISO protocols which followed later, were primarily intended to solve the problems of interoperability in wide area networks. The Reference Model makes no claim of suitability for real-time communications networks, and in fact many researches (ourselves included) believe that it is an inadequate model for real-time communications. Nevertheless, we will use the Reference Model terminology and concepts since they are well established and widely understood, thereby forming a framework for the comparison of different protocols.

5. Transport Functions

In the ISO OSI Reference Model, each layer represents some encapsulation of functionality. Classically, the transport layer contains those functions necessary to provide a reliable and transparent data transfer. If the services of the transport layer are sufficiently robust, the service is considered a *virtual circuit* service, where a connection is established and maintained, and errors in data transfer are detected and recovery actions are taken. A quality of service, which indicates the robustness of the service required, must be negotiated at transport connection establishment.

These are functions which are normally provided by a transport layer:

Negotiation of Quality Of Service

Negotiation is a multi-way handshake to establish agreed upon set of transport connection parameters. These parameters are called quality of service (QOS) parameters. Among the considerations during negotiation are the service user's requirements, the quality of the available network services, and the acceptability of the ratio of user required service to cost.

Protocol Data Unit Numbering

A service provider allocates a sequence number to a protocol data unit (PDU) that starts at zero and increases by one for each PDU sent by that provider on the same transport connection. When a PDU is retransmitted, its sequence number must be the same as the sequence number of the original PDU. Modulo arithmetic is used to allow a fixed bit field to represent this number. These numbers may be used to impose ordering, flow control, error recovery, or any other function that requires a unique and sequenced identification number assigned to each PDU for a transport connection.

Segmentation and Reassembly

Segmentation is the mapping of a service data unit (SDU) onto an ordered sequence of one or more PDUs. This sequence cannot be interrupted by other PDUs on the same transport connection. All PDUs except the last one in a sequence must have a length of data greater than zero. An End Of Transmission flag may be set false to indicate that there are subsequent PDUs in the sequence, and set true to indicate that the sequence has come to an end. It is the service provider's responsibility to transparently segment a single SDU into multiple PDUs at the sending entity and to reassemble them into their original format at the receiving entity.

Acknowledgement

An acknowledgement is a special protocol data unit that is sent from the receiver to the sender indicating receipt of one or more PDUs. Acknowledgements typically carry the number of the next PDU expected, which is one greater than the sequence number currently being acknowledged. An acknowledgement confirms the receipt of all PDUs in sequence prior to and including the PDU being acknowledged. Another field typically carried by an acknowledgement is a credit field, which is used to adjust the sender's window size and thus control the flow of PDUs from the sender.

Retransmission

Retransmission is a method for recovering from lost or corrupted PDUs. A sender will timeout if an acknowledgement has not been received for outstanding PDUs within a time limit. For a set number of times, the PDU will be retransmitted in hopes that another try will succeed.

Service Data Unit Delimiting

Service data unit delimiting is a function used to determine the beginning and ending of an SDU. During transmission, an SDU may be segmented into one or more PDUs. The PDU which signals the end of the SDU will have some flag or marker, usually an End Of Transmission flag set to true.

Error Detection

Error detection is a function used to detect the loss, corruption, duplication, misordering or misdelivery of PDUs. Loss of a PDU is signaled by a timeout condition, and error recovery methods are taken. A PDU is considered corrupt if it has a checksum and the checksum algorithm indicates corruption. In some protocols this is the only form of error detection. If a PDU is received with a sequence number that matches a PDU that has already been received, an acknowledgement for that PDU is retransmitted. If PDUs arrive out of order, the error recovery functions are employed to properly order them. If a PDU arrives with a sequence number that falls outside the current window of the receiver, a protocol error has occurred, the PDU is discarded and recovery procedures take place.

Error Recovery

Error recovery is a function used to recover from detected and signaled errors. The underlying network may signal a disconnect, which will cause the service provider to reassign the transport connection to another network connection, then resynchronize the

communication. If this cannot be done, the transport connection is released and its reference frozen. If the network signaled a reset, then resynchronization occurs. A frozen reference prevents the re-use of a reference while PDUs associated with the old use of the reference may still exist.

In order to recover from detected errors, PDUs are retained until they are acknowledged. If no such acknowledgement is received before a timeout condition is reached, an error is assumed to have occurred and a duplicate PDU is retransmitted. If the PDUs arrive misordered, they are resequenced according to their sequence numbers. Prolonged inactivity on a connection causes the transport connection to be dropped assuming failure. Receiving a PDU that is invalid or constitutes a protocol error will cause an error PDU to be transmitted, the network connection will be reset or closed, or the transport connection will be released.

Flow Control

The object of using explicit flow control is to help avoid congestion at the transport connection end points and on the network connection. Its typical use is when traffic is heavy and continuous, or when there is intensive multiplexing. Proper use of flow control can optimize response times and resource utilization. One method of imposing flow control is to use a *sliding window* protocol, where the window size indicates the number of PDUs that the receiver is allowing the sender to transmit. By varying the size of the window, the sender's transmissions can be throttled according to the resources available in the receiver. Transmission may be cut off altogether by closing the window (setting the window size to 0). The window size generally reflects the amount of buffer space available to the receiver. Proper use of the sliding window protocol can prevent PDUs from being sent and rejected due to resource errors, thus cutting down on the retransmission traffic and delays waiting for timeouts.

Congestion Control

Congestion occurs when too many packets are present in a subnetwork; this condition generally degrades service. Congestion is a point-to-point phenomenon, as opposed to flow control, which is end-to-end. In routing between segments, stations that perform routing must have adequate resources in order to move the packet from one segment to another. When these resources, such as buffer space, are not available due to overcrowding at the router, some packets are lost. Reserving buffer space at each intermediate station upon connection establishment is one method to control congestion.

Address Mapping

Service access points are the point at which the service user may access the service provider. These access points have at least one and sometimes many addresses associated with them. It is the service providers responsibility to map these service access point addresses onto the addressing scheme of the underlying network service.

Expedited Service

Expedited service is a function used to bypass the flow control of normal data PDUs and use a separate flow control mechanism. The ISO transport expedited service employs a separate type of PDU, called *expedited PDU*. It requires a special expedited acknowledgement. No more than one expedited PDU may be unacknowledged at any time for each direction in the transport connection. The data length must be greater than zero. Other protocols specify the use of priorities to differentiate the order of processing by the service provider. The scope of the expedited or prioritized service, however, remains confined to the processing order and flow control mechanism of the service provider.

Connection oriented transport service may make use of multiple network connections to enhance service, or if several transport connections required the network infrequently, one network connection may be used to handle them.

Multiplexing and Demultiplexing

Multiplexing a function used to share a single underlying network connection between two or more service users. Demultiplexing directs the SDUs to their proper service user from the same network connection.

Splitting and Recombining

Splitting allows the simultaneous use of two or more underlying network connections to support the same transport connection to provide additional resilience against network failure, to increase throughput, or for other reasons. This implies that a resequencing function be used to ensure that the PDUs are processed in correct sequence.

Concatenation and Separation

A service provider may concatenate PDUs from the same or different transport connections while maintaining the order of the PDUs for a given transport connection compatible with the protocol operation. Concatenation is a function used to collect several PDUs into a single SDU of the underlying service provider at the sending entity and to separate the PDUs at the receiving entity

The ISO protocols are not the only set of defining documents. Other standards have been proposed that deviate slightly from the classic transport definition to provide additional or different services.

Security and Privacy

The service provider secures a transport connection against intrusion by providing

validation techniques. Without security, a service provider cannot guarantee the level of service in the presence of an intruder, and reliable delivery is jeopardized. Privacy is concerned with preventing all but the intended recipient of PDUs from access to the information contained within. Encryption services are used to provide privacy.

Accounting

Accounting is a function that records statistics concerning transport connections, such as the number of PDUs processes, the number of resource errors, the number of duplicate PDUs, the number of retransmissions, etc.

Monitoring Quality Of Service

The service provider transparently invokes a function that regularly exchanges status information to monitor the adherence to the negotiated quality of service.

Common Time Reference

The service provider maintains a guaranteed time reference for the rest of the LAN. This is necessary for the enforcement of deadlines.

Network Management

Network management configures, monitors, and maintains a network. Management configuration services provide for initializing a network and gracefully inserting and deleting network entities. Monitor services gather information about the usage of the network to aid in the reallocation of resources within the network. Maintenance services provide problem detection and diagnosis and preventive maintenance.

Within the ISO OSI Reference Model, the services that are provided by layers higher than transport are generally enhancements to the basic connection oriented service provided by transport. Dialogues are managed at the session layer. A transfer syntax is negotiated at the presentation layer. The application layer provides any specialized use of the network resource,

such as File Transfer, Access and Management (FTAM) functions.

Dialogue Maintenance

A dialogue is an orderly management of a communication session. During a dialogue the service provider is responsible for maintaining state tables and issuing synchronization points. A synchronization point is an established firewall during the dialogue before which all prior communication is guaranteed to have been successful. In the event of a network failure, these synchronization points represent the starting point of the dialogue once the network has been restored.

Syntax Negotiation

Heterogeneous environments may use different representations of data. During syntax negotiation, either environment's syntax may be chosen as the data representation for the subsequent communication session, or a third, abstract syntax notation may be agreed upon.

The services that are expected of the transport layer are provided by the underlying network. The transport layer is the first layer in the Reference Model that presents the user with the notion of end-to-end service. The network layer provides the routing capability to transparently handle multiple segment LANs.

Routing

When multiple segments are joined to form a LAN, generally one or more stations will be designated routers. A router must be able to recognize addresses external to the local segment and direct the packet to the segment where the address is valid. Routing tables and routing algorithms are the tools used to provide this functionality.

Framing

Data is transmitted over physical media as symbols. These symbols are collected into finite size units called data frames. Framing is the function of delimiting these data frames with header and trailer information for transmission over the physical media and the collection of the symbol back into the data frames.

Media Access Priority

Data frames are ordered for transmission due to some priority scheme imposed by the media access control layer.

6. Protocols

In this section we examine eight specific protocols:

(1) Manufacturing Automation Protocol (MAP)
(2) MAP Enhanced Performance Architecture (EPA)
(3) ISO transport protocols (virtual circuit and datagram)
(4) French GAM-T-103 Military Real-Time LAN protocol
(5) Versatile Message Transaction Protocol (VMTP)
(6) Express Transport Protocol (XTP) for the Protocol Engine
(7) ANSI Fiber Distributed Data Interface (FDDI)
(8) SAE High Speed Ring Bus

The functions defined in the previous section are provided as services in some protocols. In order to see what functions are included in which protocols we have constructed three quick reference tables. Table 1 shows the functions generally considered provided by the robust, connection-oriented transport services. By virtue of the fact that the transport layer is defined by the ISO OSI Reference Model, the services provided by Class 4 Transport are considered the classic transport protocol.

Function	MAP 3.0	ISO VC	ISO DG	GAM-T -103	VMTP	XTP	MAP EPA	FDDI	SAE HSRB
ISO Layers Included	1-7	4	4	3-4	4	3-4	1-2,7	1-mac	1-mac
Negotiation of QOS	X	X	X						
PDU Numbering	X	X		X	X	X			
Segmentation/ Reassembly	X	X		X	X	X			
Acknow-ledgement	X	X		X	X	X	X		
Retrans-mission	X	X		X	X	X			
SDU Delimiting	X	X	X	X	X	X			
Error Detection	X	X	X	X	X	X	X		
Error Recovery	X	X		X	X	X			
Flow Control	X	X		X	X	X			
Congestion Control						X			
Address Mapping	X	X	X	X	X	X	X		
Expedited Service	X	X		X					
Multiplex/ Demultiplex	X	X							
Splitting/ Recombining	X	X							
Concatenation/ Separation	X	X		X					

Table 1 — Classic Transport Functionality versus Existing Protocols

Table 2 shows the functions that are not normally associated with the classic transport. These functions are either additional services provided by other transport layer protocols or they are functions associated with layers above transport in the ISO OSI Reference Model.

Function	MAP 3.0	ISO VC	ISO DG	GAM-T -103	VMTP	XTP	MAP EPA	FDDI	SAE HSRB
ISO Layers Included	1-7	4	4	3-4	4	3-4	1-2,7	1-mac	1-mac
Security/ Privacy					X				
Accounting									
Monitoring OOS				X					
Suspension of Connection				X					
Common Time Reference				X					
Network Management	X			X					
Dialogue Maintenance	X								
Syntax Negotiation	X								

Table 2 — Functionality Above Transport versus Existing Protocols

Table 3 presents the functions associated with layers below the transport layer. These are also considered to in order to emphasize that it may be necessary to look beyond the transport layer for specifying the functionality of a real-time communication service.

Function	MAP 3.0	ISO VC	ISO DG	GAM-T -103	VMTP	XTP	MAP EPA	FDDI	SAE HSRB
ISO Layers Included	1-7	4	4	3-4	4	3-4	1-2,7	1-mac	1-mac
Routing	X					X			
Framing	X					X	X	X	X
Media Access Priority	X						X	X	X

Table 3 — Functionality Below Transport versus Existing Protocols

Now we examine the eight protocols individually. Each protocol is briefly presented with emphasis on its characteristics and functionality. Some discussion is given to what the authors of the protocols note as their motivating factors, which may include a criticism of other protocols. We defer our evaluation and criticisms of these protocols to the next section.

6.1. Manufacturing Automation Protocol

The Manufacturing Automation Protocol (MAP) [GENE87] is based on the ISO OSI Reference Model. The MAP network can be a collection of segments and networks with bridges, routers, and gateways providing 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 lower level network.

One of the goals of MAP is to adhere to the networking structure of the ISO OSI reference model. This model is supported by most active national and international standards organizations, and since it provides a common basis for the coordination of standards development, most existing and emerging protocols can be mapped onto one or more of the seven layers of OSI. Furthermore, the model's hierarchical design allows for easy assembling of these protocols on a layer by layer basis, reducing complexity while providing flexibility.

The following is a list of the OSI layers and the MAP standards adopted:

OSI LAYER	MAP 2.1 FUNCTIONS
7 - Application	ISO Common Application Service Element (CASE) File Transfer, Access and Management (FTAM) Manufacturing Message Service (MMS) (in MAP 3.0)
6 - Presentation	Null (ISO Presentation in MAP 3.0)
5 - Session	ISO Session Kernel
4 - Transport	ISO Transport Class 4
3 - Network	ISO Connectionless Network Services (CLNS)
2 - Data Link	IEEE 802.2 Link Level, Control Class 1 IEEE 802.4 Token Passing Bus
1 - Physical	IEEE 802.4 Token Bus with Broadband (10 megabits/sec) or Carrierband (5 megabits/sec) Modulation on Coaxial Media

Table 4 — MAP Functions

MAP has been through several versions thus far and will undoubtedly continue to change. Most commercial MAP hardware and software conforms to MAP version 2.1. Version 3.0 has been released in draft form and will soon be adopted.

Two alternatives have been specified for the network's physical media: broadband or carrierband on coaxial cable conforming to the specifications of IEEE 802.4. The Media Access Control (MAC) sublayer specified by MAP is the token passing bus access of IEEE 802.4. The MAC is responsible for several actions. It provides for the maintenance of the token, which is the "permit" passed between nodes granting transmission privileges. The MAC is also responsible for the maintenance of the logical ring of nodes. Although the physical topology is a bus with nodes attached to one cable, MAC ensures that each node has two logical (not physical) neighbors, a predecessor and a successor. The token passes from neighbor to neighbor, eventually returning to the first node to complete the logical ring. MAC also handles

error detection in addition to message and priority management. Messages can be assigned priorities, and the protocol guarantees that the highest priority messages will be delivered within a measurable and deterministic time limit.

The Logical Link Control (LLC) is specified by the IEEE 802.2 standard. There are three types of service: connectionless mode, connection-oriented, and connectionless mode with acknowledgements (called LLC Type 1, 2, and 3, respectively). Type 1 service is called datagram service, and it provides for the exchange of data between two logical link entities without first establishing a connection. These protocol data units are not acknowledged and there is not provision for flow control or error recovery. Type 2 operation establishes a connection between two LLCs prior to any exchange of information-bearing protocol data units. A protocol data unit is sent from the source to the destination and an acknowledgement is returned. A number is assigned to each protocol data unit and used in control of traffic and acknowledgements. Type 3 service is a proposed datagram service with acknowledgement of data frames only when the sender specifies it. MAP specifies using the Type 1 connectionless service. Type 3 service (datagrams with acknowledgements) is proposed for improving the reliability of the LLC for use with MAP/EPA.

The network layer may be active, providing services which convert global address information into routing information so that a message can be delivered from one segment to another. If there is only one network segment (no routers) then a null network layer header may replace the routing information; this is called the Inactive Network Layer Protocol (INLP).

The transport layer provides two types of services: connection management and data transfer. The facility for the creation and deletion of a data path to a peer transport user is provided in connection management. In data transfer, the data may be sent by the normal path or by expedited means when the message is urgent. To provide these services MAP has

selected the ISO Transport Protocol Specification Class 4. This protocol has provisions for flow control for data transfer efficiency. It can multiplex users, providing each user with access to the network for transmissions. It can detect errors and recover from lost, damaged or out-of-sequence packets of data. An addendum to the standard (ISO Connectionless-Mode Transport Service) provides a datagram oriented service.

MAP specifies the ISO Basic Connection Oriented Session Protocol Specification for the session layer, with full-duplex communications.

MAP 2.1 does not specify a presentation layer, and thus it is a null layer. However, future MAP versions (beginning with version 3.0) will specify the presentation layer as using ISO Connection Oriented Presentation Service Definition.

The MAP specification of the application layer is the ISO Common Application Service Elements (CASE). CASE provides the five service elements most often used by application processes. Also present at this layer are Specific Application Service Elements, or SASEs, which provide service elements used only in special applications. An example of a SASE specified by MAP is ISO File Transfer, Access, and Management (FTAM).

6.2. Enhanced Performance Architecture

The Enhanced Performance Architecture (EPA) [GENE87], is a subset of the MAP architecture that is designed to provide faster message response times at the sacrifice of upper level functionality. A full MAP node would include all seven layers according to the MAP specification. EPA bypasses the upper layers, connecting the application processes directly to the datalink layer in an effort to streamline critical communications. It is designed to exist in specific environments for obtaining faster response time for use on control and time-critical networks. This special case is usually a small, tightly coupled group of devices which require

rapid response, use small messages, and need little or no support from a directory server.

In order to achieve the quickened communications promised by EPA, several layers in the OSI reference model have been bypassed. These layers provide functionality necessary for more complex and robust communications, which increases the processing and response times. By eliminating the layers, the procedural interface between the layers and the control information used by the layers is also removed. This lessens the processing time by reducing the amount of software to actually handle the message and reducing the message overhead due to headers. There is a price to pay for such expedience, however.

In obtaining this streamlined architecture, the application processes interface directly with the datalink layer. This means functionality is lost and restrictions apply. Skipping the presentation layer means no negotiation or changing of the message encoding scheme or its transfer syntax. The presentation syntax must be known *a priori* for applications that use EPA services. Data streams cannot be monitored by the checkpoints and the resynchronizing services that the session layer normally provides. The transport layer is not present to guarantee message delivery. The only assurance given is that a "best effort" attempt will be made to deliver the message. High quality guaranteed message delivery is therefore sacrificed. The messages are restricted to transfer across a single network segment because the network layer internet services are missing, so there can be no global delivery of messages. Only one unacknowledged message no larger than the Data Link protocol data unit may be sent. This reduces throughput if errors occur frequently, and restricts the length of the message because no facility exists for disassembling and reassembling messages. The type of service can only be connectionless (datagram, or Type 1), with the option of immediate acknowledgement (Type 3). If the Type 3 acknowledged message is used, this becomes a "stop-and-wait protocol" (the sender stops and waits after each message transmission until an explicit acknowledgement of the last message sent is received).

The specifications for EPA include the IEEE 802.4 phase coherent carrierband for the physical medium. The datalink layer is IEEE 802.4 MAC and IEEE 802.2 LLC, the same as for a full MAP architecture. These layers will be managed by the same network management scheme as the full MAP, as specified in IEEE 802.1B Station Management.

6.3. ISO Transport

The transport entity provides transparent end-to-end transfer of data between transport users, relieving the user of any concern with the details of how the data is transferred. There are degrees of reliability offered and the techniques for ensuring this reliability are also transparent.

There are two types of transport layer services defined by ISO. The first is specified by ISO 8073 Connection-Oriented Transport Layer Service [ISO8073]. It provides the transport user with connection management and data transfer services through *virtual circuits*. The second is specified by ISO 8602 Connectionless Mode Transport Layer Service [ISO8602]. It provides a *datagram* service for data transfer without connection maintenance.

The connection oriented transport layer has the functionality to manage connections that form a data path between two transport entities. The data transfer service provides normal and optionally *expedited* transfer of data between peer transport processes. The transport entities also negotiate certain characteristics of a transport connection to be observed between the transport connection endpoints. These characteristics are quality of service parameters and are negotiated during connection establishment.

When a large TSDU is passed to the transport entity, it may have to be broken down into smaller, more manageable TPDUs. This is called *segmenting*, and it is the transport provider's responsibility to *reassemble* these pieces into the original TSDU upon delivery to the peer transport user. A transport connection may use two or more network connections to improve

throughput. This is called *splitting* and *recombining*. As in the case of segmenting and reassembling, the transport provider must recombine the pieces of the message before it can be delivered to the user. *Multiplexing* and *demultiplexing* allow two or more transport users to use the same network connection. Again, this is transparent to the users and it is the responsibility of the transport provider to make this service transparent. Sometimes the size of several TPDUs is small enough that they can be placed into the same network service data unit (NSDU). This is called *concatenation*. Once received by the corresponding transport provider, the TPDUs must be *separated* before final delivery as TSDUs.

There are five classes of service defined by the transport protocol. The most robust and thus must interesting of these is Class 4. Class 4 provides the highest degree of reliability for an error-prone network. Many mechanisms and timers are required to provide this level of service.

Sequence numbers tag the TPDUs with identifying and ordering information. Each TPDU is stamped with a unique number, one more than the last TPDU processed, and one less than the next TPDU in line. This number is then used to acknowledge receipt and order TPDUs before handing them up to the transport user.

A *window* is an ordering of sequence numbers that are termed active. This may be the list of sequence numbers on TPDUs that can be transmitted or received. The window extends from the next sequence number to be processed to one greater than the last sequence number for which processing is possible.

A TPDU with the special purpose of acknowledging the receipt of one or more data TPDUs is called an *acknowledgement* (ACK). ACKs carry the sequence number of the next active data TPDU (one greater than the sequence number that is being acknowledged) and a *credit* field. The credit field in an ACK is the means for specifying how many sequence numbers after this one are considered active by the receiving user. As a consequence ACKs

also represent permission to transmit. A *closed window* has a credit of 0, which effectively turns off the transmitting node. An ACK with updated credit information may be sent at any time to control, or "throttle", the transmitting node.

There are several timers that are useful in flow control and error recovery. These timers "timeout" when they reach certain set values. The values for the timers are determined from negotiations or supplied by the underlying network. These values are expected lengths of time for certain events, and if the event does not occur within that expected time an exception is signaled.

The retransmission timer signals when a TPDU has exceeded its time limit on waiting for an acknowledgement. Such a signal will cause the transport entity to retransmit the TPDU. To ensure that peer transport entities are still communicating, an inactivity timer is used. When the amount of time specified by this timer has passed without a response from the remote transport entity, the local transport entity initiates a release due to inactivity. To keep the window information current, and to ensure that flow control is affected, a window timer signals for periodic transmission of window and credit information.

Connectionless-mode transfer of data at the transport entity provides the transport user with a single-access data transfer without the overhead of transport connection establishment. This service is intended for the benefit of those applications that require a one-time, one-way transfer of data, towards one transport user, taking full advantage of mechanisms which are simpler than those used in Connection-Oriented Transport Services.

The functions provided are at least those necessary to bridge the gap between the service available from the network layer and the service to be offered to the transport user. These functions include selection of the network service that best fits the requirements of the transport service user, address mapping for determining the destination address from the function

parameters, TSDU delimiting, and error detection.

The model of the connectionless mode transport service is similar to the model for connection-oriented transport service. The transport entity communicates with the transport service user through one or more transport service access points (TSAPs) by means of transport service primitives. These primitives cause, or are the result of, the exchange of TPDUs between peer transport entities. Protocol data units are exchanged by making use of the network layer services.

Transfer of data may occur over a connectionless mode network service, or a connection-oriented network service. The purpose of using a connectionless network service is to provide the one-time, one-way transfer of a TSDU between transport service users without confirmation of receipt, without transport connection establishment and release, and without network connection establishment and release. Connection-oriented network service satisfies the purpose of connectionless mode network service with the added reliability of a network connection, and with the added cost of that connection overhead.

6.4. GAM-T-103 Military Real-Time Local Area Network Protocol

GAM-T-103 [GAMT87] is a military real-time local area network protocol which formalizes and organizes into an hierarchical framework all the functions that equipment must fulfill in order to ensure their proper interconnection within a system. From the user's standpoint, GAM-T-103 combines all facilities in order to ensure data communications between heterogeneous equipment in a real-time system. It allows multiplexing of various data flows on the connecting network, and integrates network management and network synchronization with data communication.

There are four layers in the GAM-T-103 Reference Model: the USER layer, the TRANSFER layer, the DATALINK layer, and the PHYSICAL layer.

ISO OSI	GAM-T-103
7. Application	USER
6. Presentation	
5. Session	
4. Transport	TRANSFER
3. Network	
2. Datalink	DATALINK
1. Physical	PHYSICAL

Table 5 — ISO OSI Reference Model and GAM-T-103 Layers

The USER-layer/TRANSFER-layer interface is standardized by GAM-T-103. The other interfaces are not standardized to allow equipment dependent optimization, nor does it specify or constrain any implementation of the physical interface.

There are several groups of functions presented to the USER layer. The *data communication functions* convey information between all USER entities. This information exchange can be either by data-units over a connection oriented service, or by datagrams over a connectionless-mode service. The *process synchronization functions* provide a common time reference and a tele-interrupt service. The *network management functions* provide configuration, monitor, control, and test and maintain services.

The TRANSFER layer transmits data-units in a transparent fashion between USER entities. This frees the user from having to supply transfer execution details as the services provided ensure end-to-end control of data-units. The TRANSFER layer is designed to handle the needs of real-time systems with predetermined and dynamic data flows.

The functions of the TRANSFER layer consist of the following. TRANSFER service data units are transferred by *transmission* functions. Service primitives provide for the interaction

between a service user and the service provider. These primitives are represented as events which are logically instantaneous and indivisible. A request contains local-scope management information which may be data flow control commands or interface control commands. To provide end-to-end *flow control* on individual connections, the interface may be set up as a queue which rejects data-units if it is full, providing a *flow control process* at the interface, or the interface may be set up as a circular queue which overwrites the stale data-units with newer data-units, called a *sampling process*. The *interface control functions* provide the user with the state of the service information.

All or some of the services available may be provided through a TRANSFER service access point (TSAP), and thus the TSAP characterizes the type of communication. A TSAP is mapped to a USER entity in a one-to-one manner. Any TSAP has an individual TSAP address which is unique and global in meaning. It may also be a group address which refers to all or part of the existing TSAPs. *Address mapping* is the mapping of a TSAP address onto a DATALINK address.

The TSAP multiplexes all or some of the services of the TRANSFER layer. The access point specific to a TRANSFER service is called a Connecting End Point (CEP). A CEP reference identifies the requested service for the USER entity. These types of services are connection oriented and connectionless-mode, further divided into service classes.

The quality of service (QOS) describes certain characteristics of a TRANSFER service. It is described in terms of the service class and the service quality parameters. The service class specifies the principle performance criteria, speed and reliability. The quality of service parameters allow a service class to be characterized in a refined fashion. These parameters may include latency delay, priority, active group integrity condition, maximum TSDU size, and minimum throughput. The *latency delay* is the maximum permissible delay between the

moment the service request primitive is accepted by the network and the moment the associated confirmation primitive is provided by the network. The *priority* is the relative importance of a service request with respect to the service processing order by the TRANSFER layer. A *scheduling function* manages the priorities and monitors the latency delays. *Active group integrity condition* sets the minimum number of members necessary to continue normal communication. Once a station's presence within an active group causes the quality of service to be diminished, that station is dropped from the active group. The remaining parameters specify the size of the largest TSDU the connection may handle, and what minimum throughput is expected.

Several types of communication are recognized. *Point-to-point* communication is the transmission of data between two users. *Multipoint* communication is transmission of data between more than two members of a group. *Broadcast* communication is transmission of data from a sender to members of an undefined group. *Multicast* communication is transmission of data from a sender to a totally or partially defined group. Finally, *concentration* communication is transmission of data from several members of a group to a single source; the resulting service data unit is a *concatenation* of all the service data units transmitted.

There is a connection-less mode service which has two classes. Class 0 provides a best effort with no guarantee of delivery, otherwise known as unacknowledged datagram. Class 1 provides best effort with a service report indicating delivery or rejection due to error or latency delay expiration. Connectionless-mode communication is useful for point-to-point transmission and multipoint transmission.

There is also a connection oriented service. Stations have a connection operating mode which dictates who may make and/or control a connection, which may be predefined or dynamic. *Connection* services include the creation, establishment, opening, closing, deletion,

recovery, and suspension of TRANSFER connections. End-to-end *sequence control*, *error detection*, and *error recovery* are provided for individual connections. Error detection includes *service quality monitoring*. *Fragmentation* of TSDUs for transmission and *reassembly* of the TSDUs for delivery are provided. *Concatenation* of concentrated TSDUs and their *separation* are provided. There are three service classes. Class 0 provides an unacknowledged connection service with TSDUs of limited size and no duplication or sequence control. Class 1 provides a service report on the delivery of TSDUs of limited size. If the transfer succeeded, the TSDUs were delivered in order without duplication. This is accomplished by end-to-end *sequence control*, *error detection*, and *error recovery* (including *service quality monitoring*) for individual connections. Class 2 provides Class 1 services on any size TSDU, which constitutes the addition of *fragmentation* of TSDUs for transmission and *reassembly* of the TSDUs for delivery. All quality of service parameters apply to all classes. There are many communication scenarios conceivable, based on the communication type, connection operating mode and connection topology.

The *network management functions* are used to configure, monitor and maintain the network integrity. Network management provides services to initialize connections with predefined network parameters; insert and remove transfer entities; monitor the states of all the various entities which constitutes the network; control network entities by changing their operating modes; and ensure network maintenance by diagnosing abnormalities.

Synchronization services are provided by a global time reference and a tele-interrupt service. The time quality must be guaranteed by the synchronization entity. The tele-interrupt service ensures the transfer of an interrupt vector to a destination TSAP or group of TSAPs with two quality of service parameters, latency delay and priority.

6.5. Versatile Message Transaction Protocol

The Versatile Message Transaction Protocol (VMTP) [CHER88] provides transport communication services via a message transaction model. This model is based on a request/response paradigm, and is designed to support remote procedure calls, multicast and real-time datagram services. There is no concept of a connection in VMTP. The message transactions, rather, facilitate conversations on a higher level. VMTP detects and handles duplicate packets within a transaction by techniques dictated by the situation, and thus are efficient. Overruns are recognized as the primary form of packet loss, thus a selective retransmission of the packets reduces the cost of error recovery. A simple form of flow control based on groups of packets follows naturally from selective retransmission, providing directly the behavior that sliding window protocols must mimic for efficiency. Information is provided with the selective retransmission scheme that allows for adjustment of inter-packet gaps as a tool for reducing overruns.

VMTP makes several assumptions on the environment. It is assumed that there are nodes connected by an underlying local area network or several tightly coupled networks. The underlying network is assumed to have some basic datagram and multicast facilities, as well as internetwork address mapping. Also, the inter-node communication is assumed to be dominated by page-level file access, application level remote procedure calls, real-time datagrams and multicasts.

A *message transaction* is initiated by a *client entity* sending a *request message* to a *server entity* and terminated by the server sending back a *response message*. A message transaction takes place between two network-visible entities. The response implicitly acknowledges the receipt of the request. Either the next request, an explicit acknowledgement, or a transaction timeout acknowledges the receipt of the response. A client may only have one outstanding

message transaction at a time. Request and response messages may consist of multiple packets, and conversations of more than one message consist of multiple transactions.

Each entity is addressed with an identifier. A group of entities may be identified by a single entity identifier, and those entities may be distributed across several machines.

A *conversation* is a sequence of related communication actions. A connection is a protocol mechanism for implementing conversations; that is, it is logical entity in a protocol that associates a set of communication actions. Most transport-level protocols use the virtual circuit notion of a connection to support conversations. VMTP is designed based on the claim that most communication in a distributed system either requires a conversation at a higher level than the transport level or does not require a conversation at all. Thus there are no significant performance benefits to a transport-level connection in the assumed communication environment. VMPT provides facilities for higher level modules to implement conversations, but does not implement them directly. To support conversations, VMTP provides stable addressing and message transactions.

Stable addressing ensures that an entity address either retains the same meaning as long as the meaning is valid, or that the address becomes invalid for a period of time sufficient to avoid confusion. This is essential is providing an adequate basis on which to build higher level conversations. A client is assured that a server identifier either maps to the same entity or is invalid. The server can compare the client identifiers with identifiers from previous transactions to be assured that the same client is requesting a transaction. The server can also test the validity of a client and remove any high-level connection resources associated with that client if its identifier is invalid.

Conversations are typically structured as a sequence of message transactions addressed to the same server. Stability ensures that each request is delivered to the same server or is not

delivered at all. The server associates each request message with the same conversation and authorization based on the common transaction identifier. Furthermore, VMTP provides reliable message delivery with duplicate suppression.

There are several reasons why facilities for conversations rather than connections are provided. There is a redundancy in connections when a connection is provided by the transport-level protocol which is in turn used to set up another connection at the application level. For example, a file server conversation starts with opening the file, there are reads and writes, then the file is closed. At the transport level, a connection to the remote file server must first be established before the file server conversation may take place. Once the file is closed, then the connection to the file server is closed. These two conversations are redundant. A connection must also be maintained by using connection descriptors. VMTP only maintains one record per client at the server entity, and one record per transaction at the client entity. These records are automatically allocated when a new transaction is started, and automatically deallocated when the conversation times out. Thus there is no formal open and close packet exchanges. Since a client may only have one outstanding transaction at a time, the client only needs to keep one transaction record. This is independent of the number of conversations the client currently has. Without mapping the higher level conversation onto a connection, VMTP can construct a variety of different types of conversations including multicast and real-time message transactions. Another problem with using a virtual circuit approach is that applications commonly have multiple files open at once, requiring the application to multiplex its conversation onto many connections.

There are three variants on the basic message transaction which widen its applicability and efficiency. These variations may be combined to provide even more flexibility. A *group message transaction* is a transaction in which the client sends multicasts to a group of server entities. In return, the client may receive multiple responses. The message transaction is

terminated when the client initiates a new message transaction.

A *datagram* transaction occurs when a client sends the request message as a datagram with an indication that no response is expected. Requests may be sent reliably (with explicit acknowledgement) or not. If acknowledged, the client may not issue another message transaction until the datagram is acknowledged.

A *forward message session* is a transaction in which a request message may be forwarded to another server, which responses directly to the client. This is an optimization of remote procedure calls where the last line of the procedure is another call. This resembles the elimination of tail recursion optimization. Clearly extra response messages are saved. This forwarding is transparent to the client, and thus isolates authorization to certain servers.

The basic message transaction and variants support higher level communication by providing an efficient and reliable base for implementing multiple types of communication, avoiding redundancy, overhead, and the restrictiveness of transport-level virtual circuits. There is a minimal two-packet exchange for most transactions. Furthermore, the variants are all simple modifications of the basic message transaction, and therefore use the some common protocol processing code. This means the protocol processing requires less space than if these variations were provided by other protocols (e.g., ISO Connection Oriented Transport and ISO Connectionless-mode Transport are two separate protocols).

A closed group of VMTP communicating entities is called an *entity domain.* Only entities in the same domain may communicate directly via VMTP. Thus entities in two different domains may have the same identifier, which increases the number of VMTP entities possible. Entity domain identifiers are contained within each packet. Domains are small, reflecting the use of VMPT as a basis for closely interacting entities. An entity may participate in multiple domains, which allows the domain to represent an administrative or authorization level. Thus

domains may serve as spheres of trust, authentication and security.

A VMTP implementation must be able to deal with duplicate request and duplicate response messages. Several techniques are provided; the most efficient is used in each situation. A request packet is identified by a source entity identifier and a transaction identifier. For any particular entity identifier, the transaction identifier is monotonically increasing. The transaction identifier is recorded in the client record. The server keeps a transaction record, but only for a maximum time, and then discards it. If a node receives a request packet and it has a transaction record for that source, then by examining the transaction record, it may be determined if the new packet is part of a new transaction, a current transaction, or a delayed duplicate. If the transaction record is not present, then the packet represents the start of a new transaction as the transaction timeout guarantees.

A response packet is discarded if it does not match the transaction identifier associated with the client entity identifier to which it is delivered, or the client is invalid. By enforcing timing relations, a response packet specifying a valid client identifier and the correct transaction identifier cannot be old enough to represent a previous use of these identifiers.

If a duplicate request is received after a response has been sent but before transaction timeout, the response must be retransmitted in case the original response was lost. Another message transaction from the same client implicitly acknowledges the receipt of the response.

In earlier communication systems, retransmissions were used primarily to recover from errors in transmission. Today, error rates are lower and signaling speeds are higher, so retransmissions are now largely due to overruns. Servers may overrun their clients because they are generally faster processors and deal with many clients at once. Interconnecting high speed networks through bridges and routers may cause these intermediate nodes to be overrun due to transmission bursts. If overrunning is the cause, systematic errors are more likely (i.e.,

dropping every k-th packet due to speed mismatch).

A VMTP message may consist of up to 32 512-byte data segments. These messages can be packetized into multiple *packet groups* by duplicating the header and suffixing the data segments. The packet group header includes a bitmask which identifies the packets contained within the packet group. In this way, messages may be segmented and reassembled, and only the packet group affected will need to be retransmitted. This is called *selective retransmission.* This is a flexible facility with no variable length header or acknowledgement information. The sender can efficiently recover from slightly overrunning the receiver or intermediate nodes, and is provided information that aids in adjusting the transmission rate to reduce loss due to overruns.

In this way, VMTP uses packet group-based flow control. The transmitter sends a group of packets of at most 16 kilobytes as one operation, which is one VMTP message. The receiver accepts and acknowledges this packet group only as a unit before further data is exchanged. Selective acknowledgement and retransmission provides a means of dynamically detecting when the transmission rate is too high. Larger amounts of data than 16 kilobytes may be sent by constructing a higher level conversation.

This approach is seen as better than the sliding window for several reasons. In the sliding window much overhead is needed for acknowledgements and credits. To make acknowledgements count for more, bursts of packets are sent and acknowledged as a group. Rather than sliding the window forward after each packet is received, the window is moved by a large amount. This is the same as the packet group approach used explicitly by VMTP. Also, a zero credit may tie up resources for an indefinite time. Packet group-based flow control ensures that the client must be prepared to accept the response of an entire packet group at once. Buffers will be tied up at worst for the maximum retransmission time for a packet group.

Clients are forced to accept an entire packet group at a time rather than incrementally.

VMTP provides three forms of support for real-time communication. First, a priority level is transmitted in each request and response which governs its handling. These levels correspond roughly to **urgent/emergency, important, normal,** and **background.** There are also gradations within each level whose interpretation and implementation is otherwise host-specific. Secondly, datagram requests allow the client to send a datagram to another entity without blocking, retransmission, or acknowledgement (unless requested). In non-streamed mode, the datagram request supersedes all previous requests from the same client. In stream mode, the datagram is queued on the stream of requests from the same client. Finally, VMTP provides several control bit flags to modify the handling of requests and responses for real-time requirements, avoiding unnecessary overhead when appropriate. Conditional message delivery is a flag which causes a request or response to be discarded if the recipient is not waiting for it to arrive. Another flag indicates if only the header's checksum has been calculated. Such a flag may be set by applications which are insensitive to bit errors in the data. The last flag indicates that retransmissions are not to be requested by the recipient of a message if part of the message is missing. Either the recipient uses what was received or discards it completely.

Thus VMTP is a departure from the current standard transport-level protocols. It provides message transaction service rather than a virtual circuit service, with an argument that this model of communication is sufficient to support higher-level conversations without imposing upon them a transport-level connection.

6.6. Express Transport Protocol

One of the legitimate criticisms of ISO protocols is that they were not originally designed for real-time applications; they were designed for interoperability, not performance. In a test of

protocol performance over an Ethernet at Silicon Graphics Inc., it was reported in [CHES87] that, of the 10 Mbps signaling speed on an Ethernet, 6.7 Mbps were available at the MAC layer, 4.5 Mbps after passing through the internet layer, 2.8 Mbps at the top of TCP, and 1.2 Mbps were available to the application program. Some measurements of ISO protocols, as in [STRA88a] and [STRA88b], reported even lower throughput. Two other problems often encountered are:

(1) In TCP and ISO Transport Class 4, flow control operates end-to-end, so gateways do not regulate traffic. This increases the probability of congestion in the gateway, which in turn increases the probability that messages will be lost and retransmitted, further reducing performance.

(2) Internet gateways add buffering to the connection, thus increasing end-to-end message latency.

These observations inspired Greg Chesson to begin the Protocol Engine Project at Silicon Graphics. The goal of the project was to design a high-speed transport protocol (called XTP for Express Transport Protocol [CHES88]) and then implement that protocol in VLSI hardware. XTP combines the classic transport and network layers into a single layer denoted as the "transfer" layer.

From the outset, the design of the protocol engine hardware is intended to scale from medium speed (10 Mbps for Ethernet) to high speed (100 Mbps for FDDI) to very high speed (1 Gbps for networks now operating in the laboratory). The initial target is to interface the protocol engine to the FDDI chip set, and afterwards to the Ethernet chip set.

By design, XTP's basic service is connection-oriented (i.e., virtual circuits). Datagrams are treated as short-lived connections. Also by design, a pair of protocol engines can be used back-to-back as a bridge across same-type address domains or as a router across dissimilar

address domains. This latter option has significant consequences with regard to overall protocol design, buffer management, and flow control.

One observation which drives the design of the protocol engine is that a receiver is inherently more complicated than a transmitter. Because the receiver must perform checks and tests, system performance is heavily dependent upon the receiver's performance. XTP attempts to simplify the receiver as much as possible so that it is a very high performance engine. An example is that receivers do not generate response messages unless specifically commanded to do so; this reduces backtalk and generally improves performance.

XTP's design strategy is to start with a clever receiver architecture. The design of the transmitter follows as a complement to that of the receiver. The transmitter's tasks are to build output packets and control the receiver with command codes within those packets. The non-data portion of the packet is built directly from the state vector. Data length is only used as an error check by the receiver; the transmitter can begin transmission of a packet without knowing its length. As a trailer to the data, a sequence number, length field, and user-defined type are then appended. The transmitter handles observing roundtrip delays, adjusting connection timers and performing retransmissions, and managing a buffer queue for each connection.

XTP provides two packet formats, a data packet and a control packet. The data packet contains header and trailer information, such as an address key, a sequence number, a packet command code and a type code. The command code tells the receiver to buffer the user data, generate a response message, or establish/remove a connection context. Control packets contain flow, error, window, message, and timestamp information important to the protocol but not needed with every packet.

Because the protocol engine is designed to provide the transport (transfer) layer user with data rates comparable to those found at the media access layer, data buffering is intentionally

quite limited — the protocol engine must accept, evaluate, and buffer one packet in the time it takes for a second packet to arrive. To achieve the high data transfer rates needed, accepting a packet consists of a series of steps.

(1) Translation. As the packet header arrives, a *key* in the header is used to look up state vector information in memory.

(2) Load. The result of the address translation is used to load the appropriate state vector into the protocol engine. If no connection is active (as would be the case with a datagram or with the first data packet of a connection), a new state vector is constructed.

(3) Evaluate. Sequence numbers are checked and the packet is determined to be in order (so processing continues) or out of order (so the packet is dropped).

(4) Buffer. Incoming data is buffered pending the next accept/reject decision.

(5) Commit. After the protocol engine computes the frame check sequence, the receiver either discards data if the CRC is incorrect or else accepts the data and updates the connection's state vector.

This sequence of events is shown in Figure 1.

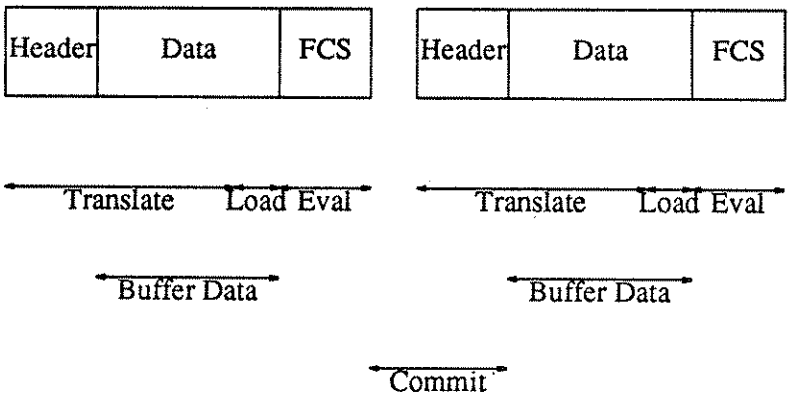

Figure 1 — Time Diagram of XTP Sequence of Events

A block diagram of the protocol engine is shown in Figure 2. The conceptual design is that the protocol engine, running XTP, would be reduced to three, four, or five VLSI chips, and that this collection would provide the transfer layer services. The protocol engine would in turn interface to FDDI or Ethernet chips for its datalink and physical layer services.

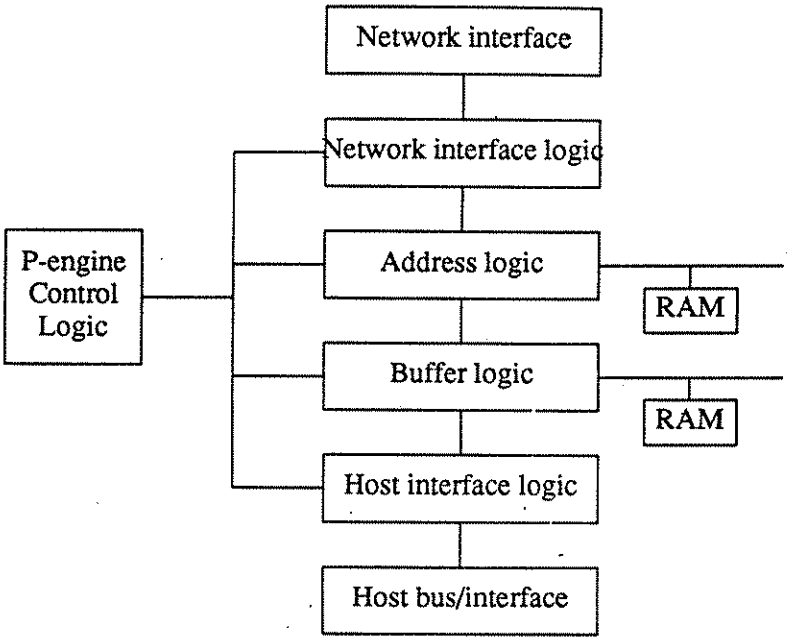

Figure 2 — Block Diagram of the Protocol Engine

Figure 3 shows how protocol engines can be used to interconnect networks. Information arriving from, say, an FDDI network is accepted and buffered by one protocol engine, then delivered to the system bus which interconnects protocol engines. From there a second protocol engine accepts the message and transmits it on another, possibly dissimilar, network.

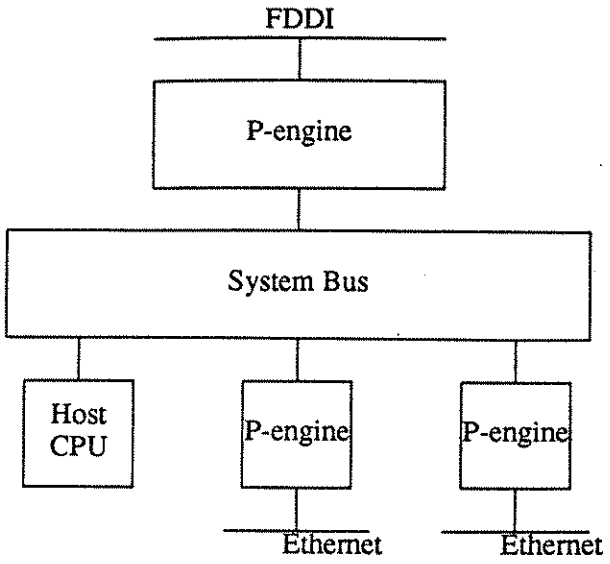

Figure 3 — Network Interconnection Using Protocol Engines

Note that interfacing an FDDI network to, say, an Ethernet could cause severe problems in a traditional transport protocol. If multiple packets were directed from the FDDI network to the Ethernet, the speed mismatch would undoubtedly cause buffer congestion in the intervening gateway, leading in turn to buffer exhaustion, a closed sliding window, and eventual retransmission of lost packets. While no protocol can make the 10 Mbps Ethernet accept data at the 100 Mbps rate of FDDI, XTP does regulate the flow of data by allowing the protocol engines, acting as routers, to participate in the flow control decisions. Unlike ISO Transport Class 4 or TCP, in which flow control operates end-to-end, XTP flow control operates between protocol engines. This allows XTP to do a much better job in regulating flow between networks of differing capacity.

6.6.1. Current Status

XTP and its underlying protocol engine are difficult to track because they continue to change. XTP's current version is 3.1 but a version 4.0 is already promised. An older version

numbered 3.0 has been coded as a 'C' program and demonstrated on a pair of Unix-based Silicon Graphics workstations. Version 3.1 has been coded and is near release, but still contains bugs as of this writing.

6.7. Fiber Distributed Data Interface

ANSI's Fiber Distributed Data Interface [FDDI86] is a completely fiber optic network with a dual redundant counter rotating ring. The dual ring encorporates enough redundancy into the architecture to allow recovery from a single break and some forms of double breaks in the network.

FDDI signals at 125 Mbps but, due to its 4-into-5 bit encoding scheme, it yields an effective data rate of 100 Mbps. FDDI specifies a maximum of 1000 stations cabled over a ring circumference not to exceed 200 km. FDDI supports three classes of transmission: *Synchronous* for time dependent communication, *Asynchronous* for interactive communication, and *Immediate* for network management communication.

The FDDI standard is divided into four parts. The Physical Layer Medium Dependent (PMD) portion specifies how stations should communicate with each other on the ring. It is responsible for translating encoded electrical data to and from optical signals. The Physical Layer Protocol (PHY) decodes the incoming signal and encodes the outgoing bit stream into grouped transmission codes using the 4-into-5 bit encoding scheme. The Medium Access Control (MAC) is primarily responsible for the transmission, repetition, verification and removal of protocol data units. Station Mangement Services (SMT) work logically beside PMD, PHY and MAC to coordinate the activities of these layers so that each station may work cooperatively with other stations on the ring. SMT is responsible for network initialization and reconfiguration; it also handles fault isolation and recovery.

In terms of the OSI Reference Model, the highest layer specified in FDDI is the MAC; thus FDDI does not implement any of the transport functionalities listed in section 5. It does implement the classic datalink functions of framing, error detection, and media access priority. Of these, only the later has any direct influence on FDDI's role in real-time communications.

PDU transmission is controlled by a "timed token" mechanism which utilizes a target token rotation time known to all stations. The target token rotation time or TTRT is negotiated by all stations through a bidding process in which each station makes known the shortest token rotation time it will need to service its most time dependent traffic. The least of these bids is adopted as the target token rotation time.

Within each station there is a token rotation timer. On each token access the token rotation timer or TRT is compared to TTRT. If TRT is less than TTRT the token is considered early and the station is allowed to transmit asynchronous traffic for this token access. If TRT is greater than TTRT the token is considered late and no asynchronous traffic may be transmitted for this token access. In this way the progress of the token around the ring is governed to insure that the token rotation time does not often exceed its target. If a station is allowed to transmit asynchronous traffic, the token holding timer or THT within each station controls the duration of this transmission for each token access.

FDDI supports both synchronous and asynchronous service classes. Synchronous service is for applications requiring guaranteed bandwidth and response time, such as real-time voice communication. At ring initialization synchronous bandwidth is allocated from the total ring bandwidth. Each station must be allocated a non-zero percentage of this bandwidth if the station is to support synchronous traffic.

Asynchronous service is for applications whose bandwidth requirements are less predictable. Asynchronous bandwidth is allocated from bandwidth unused and/or unallocated

by synchronous traffic. FDDI supports an infinite number of asynchronous priorities, each with its own priority threshold. When used, the priority thresholds impose restrictions on the asynchronous service class such that lower asynchronous priorities receive less service that higher asynchronous priorities. If asynchronous service is not defined, then the default for all priority thresholds is the target token rotation time. FDDI also supports a restricted token mode in which a small group of stations consumes basically all the bandwidth not used by synchronous traffic.

6.8. High Speed Ring Bus

The SAE HSRB [SAE87] is superficially similar to FDDI in that it is a high-speed token ring LAN based on a dual-redundant, counter-rotating ring topology. The HSRB protocol is written to be independent of medium, transmission rate, and station separation. The parameters which are dependent on these physical characteristics are defined in appendices to the protocol document. Both a 50 MHz electrical implementation and a 100 MHz fiber optic implementation are described. The standards committee is reviewing a proposal for a 500 MHz implementation.

The HSRB was specifically designed for real-time distributed systems, and is expected to replace MIL-STD-1553B for military avionics systems. Compared to FDDI, HSRB makes many changes to the framing structure to encourage high-speed decoding in hardware. Even so, the highest OSI layer defined is the MAC, and so HSRB, like FDDI, defines no transport services. In terms of the functionalities defined in section 5, only framing, error detection, and media access priority are supported.

The value of HSRB to this discussion is its media access priority scheme and how it differs from FDDI.

The HSRB protocol allows for up to eight distinct priorities of messages. These priorities are arbitrary classifications by the user which designate the importance of a particular message. The HSRB may neither set nor change these priorities. A queue is maintained by each station for each message priority, and within each queue messages are processed in order of arrival. The protocol does not attempt to set any bounds on message delivery time; instead, it attempts to guarantee a relative order of service.

The mechanism by which the priority operation is implemented is a reservation scheme. As a claimed token traverses the ring, stations with a message to send (the priority value of which exceeds the value of the current reservation) copy the priority value of their message into the reservation field. When the sending station receives the token it issued, it compares the value of the reservation field with its highest priority message (if any). If the reservation value is higher or if the station has no more messages to send, then a free token is issued with that priority. If, however, the priority value of the message is greater than that of the reservation, the free token issued has a priority equal to the message's priority. The token continues around the ring until claimed by a station with a message of equal or greater priority. This station need not have set the reservation in order to claim the token. The claiming station resets the reservation to the lowest priority and strips the Token Ending Delimiter from the token. Note that even though a station may have multiple messages of the highest priority to send, it must transmit a free token after each of them. Although this may cause a slight degradation in performance at high loads, it eliminates the problem of having long periods of time without the transmission of a free token. The placement of the priority bits, token status bits, and reservation bits in the token allow minimal delay in executing the reservation logic.

When a station has a message to send, it must first acquire a free token in the manner described above. Once a token is acquired, the station changes the token's status from free to claimed and appends to the token its highest priority message. Upon completion of message

transmission the station re-issues a free token. Each station may transmit only one message per token possession. If a station has multiple messages to send, it must nevertheless transmit a free token following a message, allowing any other station with a message of equal or greater priority to claim the token. Thus, under normal operation only one message will exist on the ring at a time.

One of the goals of the SAE HSRB designers was the optimization of throughput. If the reservation priority scheme is viewed with this in mind, two cases arise. If the expected length of a message exceeds the total ring latency, then the token will have returned to the issuing station before message transmission is completed. Therefore, when the station is ready to issue a free token, the reservation value from the previous token is already known, and the token can be issued immediately. This is done with no loss of throughput. If, however, the expected message length is less than the total ring latency, the time spent waiting for the claimed token to return to the issuing station (in order to obtain its reservation value) is effectively wasted. As the transmission speed of the network increases, more bits can be stored on the ring at a time, hence this throughput penalty becomes greater. The problem is that if a station wishes to transmit a free token immediately upon completing transmission of a short message, it has no way of knowing at what priority the token should be issued. Any transmission of the free token before receipt of the previous claimed token (and, hence, before receipt of the reservation value) must necessarily violate the priority scheme. As a compromise, the designers offer an optional Short Message Protocol.

If the Short Message Protocol is implemented, multiple short messages may co-exist on the ring. The value of the Short Message Count field indicates the number of short messages immediately predeeding the current message. If that value is less than fifteen, and if the current message is a short message, then the current station increments the Short Message Count and transmits a Free Token with its priority set to the lowest priority. However, if the Short

Message Count equals fifteen, the station must wait until it receives its claimed token (i.e. the message is "forced long") before it may transmit a free token. The token is transmitted with a correct priority as described previously, and the Short Message Count is reset to zero. Thus, no more than sixteen short messages will be sent before the priority system is reinstated.

In a network in which the Short Message Protocol is not implemented, a station issuing a free token always sets the Short Message Count to fifteen, thus "forcing long" each message.

7. Protocol Evaluation

The following is an evaluation of the above protocols and comments on how effective each would be in supporting real-time communication.

7.1. ISO-Based Protocols

There have been numerous criticisms about the ISO Reference Model and its ability to provide an effective framework for all types of communication, especially communication supporting real-time applications. The problems discussed are divided into two categories, media access control and data transfer. We examine media access because it is here, at the lowest level of communication, that the resource of the network must be shared. We also examine the data transfer and associated functionality of the higher layers because these functions affect the observable performance of the communication system. The discussion is primarily centered around the MAP specification as an example of the use of the ISO Reference Model for general and real-time communication.

7.1.1. Media Access Control

There are three families of LANs: bidirectional broadcasting networks, token-based logical ring networks, and unidirectional broadcasting networks [MIRA83]. In bidirectional broadcasting networks such as Ethernet, all systems are connected to the common media by a passive interface. There are several techniques used to control access to the media, namely CSMA/CD, but experience has shown that delays are increased when the load is heavy. In token-based logical ring networks a token is circulated around the ring, granting the station with the token the right to transmit. By using an appropriate protocol, such access techniques ensure a maximum message delivery time. Unidirectional broadcasting networks force transmission signals to be propagated on the cable in only one direction. There are two interconnected unidirectional channels so each station can receive messages on one channel and transmit on the other.

Token passing is a simple solution for guaranteed access delay, but the token is a single point of failure. The loss of the token or token duplicates impact access delay, and the recovery protocol is very complex and difficult to verify [SONG87].

Token passing LANs are labeled as deterministic. However, [LELA85] argues that this is a misnomer. Determinism in message transmission refers to systems such that there exists a finite number of system state transitions for switching from the message submission state to the successful message transmission state. This is a logical concept; the utilization of a deterministic algorithm is not sufficient to meet given time constraints.

In the IEEE 802.4 token bus, access units make use of four timers. It is an important question as to how to set those four timers to keep token rotation time to a reasonable value. Calculations for setting these timers require assumptions made on the nature of the station's offered load. Offered load assumptions are typically probabilistic in nature. So is the time

spent executing the leave/insert logical ring protocols. Thus the token bus upper bounds on access delay are probabilistic.

The four priorities for network access are static. They ignore the dynamic nature of deadlines, and in fact are not responsive to time constraints. The highest priority message may still have to wait while other stations transmit messages of lower priority. Any station not using the optional priority feature transmits every message at the highest priority. The priority scheme is a channel bandwidth allocation scheme, rather than a message priority mechanism.

In the IEEE 802.5 token ring, messages are first placed in the waiting queues of the various access units. Bounded transmission time is guaranteed only if static priorities are *not* used. When static priorities are used, queueing theory shows that only clients with the highest priorities enjoy guaranteed service. Some messages of lower priority might be denied access to a ring forever, which is not deterministic service.

It is more difficult to predict what impact faults might have on access delays with token passing LANs than with contention LANs, since contention LANs have no concept of a logical ring to maintain in the presence of station failure or noise. Again, such issues as how often a token is lost or how often a station fails has a probabilistic effect on the message delay. Furthermore, election of a new control unit for the logical ring cannot be resolved in some predetermined time in all circumstances.

The overhead incurred for every token passing operation by the token handling protocol for token transmission and for lost token recovery is fixed and largely independent of the bandwidth available. Also independent of the bandwidth is the time required during reconfiguration of the logical/physical ring. This overhead may be unbearably disproportionate to the propagation delay at higher bandwidths.

7.1.2. Data Transfer

The MAP network layer provides a connectionless-mode service for routing packets through several segments. A sender assigns a lifetime to a packet, which causes the packet to become stale after that lifetime is exceeded. This implies only that the message will be discarded when the lifetime has expired, but nothing is guaranteed about delivery before the deadline. There is no priority scheme at this layer, so packets employ the address mapping algorithms with equal urgency. Thus an expedited message from the transport layer is not handled exceptionally at the network layer.

Transport Class 4 connection establishment does not ensure that the connection can offer the required quality of service. The QOS only serves as an indication. No parameter is provided to take into account the scenario description or the real-time constraints. Thus no deadline information is exchanged when the connection is established. Furthermore, scenarios like broadcast and multicast transmissions are not available here, and must be artificially simulated by the transport user. When replicated equipment must receive the same input, there is no multicast ability to provide this naturally.

Connectionless-mode service provides unguaranteed delivery by best effort; there is no acknowledgement or confirmation. Duplicates may occur in this process, but no sequencing is provided to detect duplicates. If reliability is necessary, a connection-oriented service is required. However, deadlines may prevent connections from being established as the delay in establishment may exceed the deadline. If permanent connections are established at start up, valuable resources (buffer space) are dedicated to communications. This is not suitable for rare but important communication.

An important requirement of real-time communication is an alarm, which is a very small but extremely important message demanding immediate attention. At the transport layer, only

expedited service gives a message priority, but the priority is relative to transport layer service requests only; it does not ensure expedited service throughout the network. Consider such a message requiring routing. The network layer provides no discrimination of packets. Also, a stop-and-wait protocol is imposed on expedited service limiting a user to sending only one expedited message at a time. This implies that the process sending the expedited message must wait for that message to traverse multiple segments without priority, possibly to exceed its lifetime and die, all the while blocked without the ability to send other such important messages. This does not suitably support the concept of alarms.

Experiments using a MAP 2.1 implementation has produced results to support the claim that Class 4 transport may not be able to support real-time communication efficiently [STRA88a]. The basic datagram service of datalink could provide a one way message service approximately twice as fast as could transport for message sizes within datalink's packet size restriction, as well as higher throughput. For larger message sizes, segmentation at the transport layer was performed with significant cost in terms of end-to-end delay and throughput.

Thus, time critical applications and distributed real-time systems require communication schemes that are not defined within the ISO OSI model. MAP/EPA is designed to bypass some of the overhead associated with a robust full stack implementation, but it sets many communication functions in the application layer along with the application, and this is not considered satisfactory.

7.2. Non-ISO-Based Protocols

Several protocols have been suggested that are designed for real-time communication. These protocols fit generally into the transport layer of the ISO Reference Model, but are not ISO based or sanctioned.

7.2.1. GAM-T-103

Although the layering of GAM-T-103 suggests that the TRANSFER layer incorporates the network and the transport layers from the ISO OSI model, there are not any routing functions present in the TRANSFER layer. This implies that GAM-T-103 is designed to be implemented on a single segment LAN, in which case having no routing functions is roughly comparable to the Inactive Network Layer Protocol of the ISO network layer.

Heterogeneous equipment may be interconnected using GAM-T-103 as a common LAN protocol, but the protocol does not specify a syntax of the interaction. Unless a common syntax is specified, then one has to be agreed upon within each MRT-LAN installation. This does not facilitate interconnection. However, the general case of syntax negotiation, using ASN.1, may be too expensive for real-time communication. Thus the word "heterogeneous" must be interpreted here as meaning that various types of equipment share a common data representation, rather than its normal meaning of supporting different equipment regardless of data representation. Without specifying a syntax, however, the interpretation of data is the responsibility of the application, where special case techniques may be applied without the overhead of a fully abstract syntax.

The quality of service parameter *throughput* is only used to select a transmission channel in the case of a multi-channel network. There is no means provided to monitor the throughput and report diminished service if the throughput indication is not matched. Throughput depends on many factors, making effective monitoring difficult. The throughput parameter must be an estimate of what the channel can provide, thus the selection of a channel based on throughput is a rough discrimination at best.

The quality of service class and parameters are not negotiable, and cannot be modified by the service provider, unlike ISO transport, where negotiation takes place in the connection

handshake. This implies that if a channel cannot meet the needs of an application, the whole communication is rejected. This may be appropriate for some kinds of communication, but a real-time network is specialized enough to have been designed with quality of service in mind. Negotiation or not, determining whether to accept or reject a communication request consumes time without discrimination.

It is the responsibility of the synchronization functions to provide an accurate common time reference. There are two general ways to provide this: broadcasting a synchronizing message to adjust local clocks for skew, and hardwiring an external global clock into each station. The former provides additional traffic, and if the synchronization messages are sufficiently frequent, it could degrade normal service. These messages must be of very high priority to prevent clock error, yet the situation could arise where clock updates preempt other emergency communication. Furthermore, message processing time increases the inaccuracy of this method. Using a single, global, external clock creates a single point of failure, and requires a separate physical connection to each node other than the network connection.

The quality of service parameter *latency delay* provides the means of dictating a deadline to the communication system. If the message cannot be accepted by the network within the latency delay, no message is delivered and a service report is returned to the sender. In light of the distributed clock problem, the resolution of the clock may not be accurate enough to enforce this deadline. Also, this latency delay only dictates a deadline if an acknowledged service is being employed. No such deadline mechanism exists for unacknowledged services, connection oriented or connectionless-mode.

The quality of service parameter *priority* specifies the relative importance of a service with respect to the service processing of the TRANSFER layer. As with ISO transport, this priority is not propagated throughout the network processing. No mapping is specified between

priority and network access priority at the MAC layer. Once the TRANSFER layer entity has finished processing a service request, the priority is abandoned.

Since no other interface except the TRANSFER-layer/USER-layer is standardized, no application may use any other interface without machine specific information. This may be useful for securing the network as a resource, but it also forces the application into accepting all or none of the services provided. In defense of GAM-T-103, services may be included as needed by choosing the appropriate TSAP. By limiting access to one interface, the entire network resource may be collapsed into one set of services from which to choose, not a set at every layer.

7.2.2. VMTP

There is no common time reference support in VMTP. This implies that there can be no scheduling or preferential treatment of messages globally based on deadlines. Any such support would have to be from an external source, either a global clock or from the application's environment.

Messages are bounded by 16 kilobytes, and within a message there is a maximum of 32 512-byte packets. This requires the application to do some form of segmentation. This message size is handy for page level file transfers, but this limit may not be appropriate for other applications. Theoretically a message could be of arbitrary size since transport functionality typically includes segmentation and reassembly. The message transaction model does not naturally lend itself to arbitrary message sizes, even though some from of packetization is included.

Unlike sliding window, this protocol imposes synchronization on every message (maximum 32 packets) since a client may have only one outstanding transaction. The

assessment that VMTP directly implements the behavior of an efficient sliding window is not accurate. The sliding window is most efficient when the leading edge of the window is several sequence numbers ahead of the trailing edge, and the two never meet. VMTP behavior requires synchronization after every message, sliding window does not.

7.2.3. XTP

XTP and the protocol engine are poised to revolutionize communications for two basic reasons: (1) Compared to classic transport protocols such as ISO Transport Class 4 and TCP, XTP is a light-weight protocol. It is intentionally lean and is designed from the beginning with high performance as a goal. (2) XTP is intentionally designed to migrate gracefully from a 'C' code implementation into a firmware implementation using VLSI.

If the functionality provided by XTP is sufficient for a given real-time application, then it is safe to say that no other protocol examined here will come close to matching its absolute performance. Even when implemented as a program, XTP would have a speed advantage over ISO Tranport Class 4 or TCP; when implemented in VLSI, it will be one or two orders of magnitude faster than any comparable transport protocol implemented in software.

In spite of our enthusiasm for XTP, there are several unresolved questions about it.

(1) It is unclear whether XTP can support the ISO protocols above the transport layer. XTP is not itself an implementation of the ISO transport functionalities; it is by definition a "light-weight" (and hence non-ISO) protocol. Even though the session layer is supposed to be independent of the underlying transport protocol, there are certain assumptions. Specifically, the ISO Session Protocol Specification "assumes the use of the connection-oriented transport service defined in ISO 8072". Regardless of whether it is the session protocol's responsibility to be more independent of the transport protocol, or whether it is the transport protocol's

responsibility to ensure that its services are appropriate for the ISO upper layer protocols, the issue is not resolved in XTP.

(2) There is a field within an XTP packet that is reserved for the issue of security and privacy. XTP itself, however, does not offer a suggestion as to how this field is to be used. Its presence is for future use. In fairness, no protocols other than the ISO protocols specifically address privacy and security, and the relevant ISO specifications for the presentation layer have not yet been approved.

(3) There is no provision for network management, either globally or locally. There is no means for XTP to participate in some global network management scheme which controls network operations across all layers. Other transport protocols usually specify some local network management functions, but there are no such specifications in XTP.

(4) XTP is clearly not a standard. However, standardization is being aggressively pursued through ANSI, SAFENET, and ISO.

(5) XTP is not yet fully defined, and there is no public schedule for version 4.0. The suitability of XTP for any particular application can not be accurately judged until the protocol definition is stable.

(6) Like many transport protocols, XTP redefines "real-time communications" as being "very fast communications." More than any other protocol, XTP is likely to achieve the goal of being very fast. Still, that is not the same as ensuring that no application process will miss its real-time deadlines because of its interaction with the underlying communications system. A fast, light-weight protocol increases the probability of acceptable real-time performance, but does not guarantee it.

(7) While a software implementation of XTP would have a performance advantage over ISO Transport Class 4 or TCP, achieving very high performance depends upon a successful

migration to VLSI. The true benefits of XTP will not be seen until the protocol engine is available as a chip set.

7.3. Lower Layer Protocols

7.3.1. FDDI Performance

Sevcik and Johnson [SEVC85] have shown that, using FDDI's timed token mechanism, the maximum token rotation time is at most twice the negotiated Target Token Rotation Time (TTRT). In the error-free case, it is then possible to guarantee how often the token will arrive at a station (based on TTRT) and how much transmission time will be available to that station's synchronous service class (based on the station's negotiated portion of synchronous bandwidth). Service of the asynchronous classes is not guaranteed.

Peden has shown that, given a homogeneous traffic distribution which leaves some bandwidth unallocated and thus available for use by the asynchronous classes, one can predict the end-to-end delivery time of messages of each priority. In [PEDE87] and [PEDE88], Peden provides a closed-form analytic model for message delay and verifies his results by simulation. Under conditions of very heavy load (e.g., 99% offered load), the protocol operates exactly as expected, with service to the lower priority classes being reduced in favor of service to the higher priority classes. However, in the normal case of moderate-to-heavy loads (e.g., offered loads below 90%), there are two surprising results:

(1) The timed token mechanism does not meaningfully discriminate among priority classes. Low priority and high priority traffic are intermixed, with some high priority messages being delayed by low priority messages.

(2) The very use of the priority mechanism adds overhead to the protocol and thus increases average delay for all messages, when compared to the same message mix transmitted as a single priority.

In addition, he observes that imposing media access priority at the MAC comes far too late in a message's lifetime to meaningfully influence its end-to-end delay. If a message has not been treated expeditiously at every moment of its lifetime, then adding a media access priority is a futile gesture. For real-time performance, the concept of "importance" or "priority" or "deadline" must be attached to a message within the application process, and that message's importance must be inherited by every process which subsequently handles the message.

Simonson has created a complete computer simulation facility for FDDI, reported in [SIMO88], which predicts MAC-layer characteristics such as token rotation time, network access time, end-to-end delay, queue lengths, etc. He has conducted a very large set of simulations which basically confirm Peden's predictions regarding the functioning of the priority system.

In terms of real-time performance, then, FDDI's timed token mechanism provides a gross, high-level distinction among messages, but does not implement message priority on a message-by-message basis. FDDI's main utility for real-time communications is that its 100 Mbps signaling rate is likely to be much faster than the host processor to which it is attached, thus masking the fact that it does not enforce a strict message priority scheme.

7.3.2. HSRB Performance

Minnich has developed a simulation system for HSRB, as reported in [MINN88]. Peden has developed a closed-form, analytic model for end-to-end message delays for protocols which use this reservation priority scheme (but not the short message protocol); his results are published in [PEDE87] and [PEDE88].

When the short message protocol is not implemented, HSRB does carry the highest priority message next. As explained before and as verified in the Minnich simulations, this does

extract a performance cost. To keep the token circulating at the highest reserved priority, short messages must be "forced long" so that the token returns to the transmitter before the next free token is issued. To avoid the performance penalty, the short message protocol is used to permit multiple short messages to "piggyback" with each other. While this does improve performance, it violates the concept of strict priority since now up to sixteen short messages (of arbitrary priority) may separate two messages which are truly highest priority.

Both Peden and Minnich show that the use of the reservation priority exacts a performance cost. If two messages of differing priority are offered to the ring, the protocol will select the higher priority one through the reservation mechanism. The second message must then bid to lower the token priority, and then wait for the transmitter of the higher priority message to reissue the token at the lower priority. Given a uniformly distributed destination addresses, this means that the token makes two trips around the ring before the lower priority message can begin transmission. This, of course, increases overhead, increases delay, and decreases throughput.

In terms of efficiency, HSRB is better than FDDI for a configuration which is otherwise identical. In terms of absolute performance, however, that advantage is lost when one considers the 125 MHz signaling rate (100 Mbps data rate) of FDDI vs. the 100 MHz signaling rate (80 Mbps data rate) of HSRB implemented on fiber optics.

Like FDDI, HSRB can not be properly evaluated out of context. For a specific configuration and a known traffic pattern on an error-free network, we can accurately predict MAC-layer performance characteristics. It is impossible to predict what fraction of network capacity will be available to a user program without knowing the intimate details of the network-to-host interface. While the details of the link layer protocol are important because they represent an upper bound on performance, we believe that the true measure of the

network's utility for real-time communications will be determined by its upper-layer protocols.

8. Conclusions

There are many available approaches to communications, each with a protocol and standard, and each with its advantages and disadvantages. In general, the goal of a network protocol is to relieve the network user of (almost) all concerns with the mechanisms required to communicate information between distributed hosts. Real-time communication places additional criteria on this network: communication is time-critical. Much literature is devoted to expounding upon exactly what these demands are. Principles drive functionality; we attempt to identify the principles of communication within a real-time environment.

Within the ISO paradigm of communication, the transport layer seems to provide general communication functionality with the right degree of robustness. The goal of communication systems in general is the encapsulation of these functions in such a manner as to relieve the network user from concerns of communication. These functions vary among classes of applications. By examining several protocols we have tried to show what functions are included and with what motivation they are present. Protocol writers first subscribe to a model of communication, then functionality is added to make the protocol useful.

8.1. Principles

In section 2 we presented some principles generally regarded as necessary for real-time communication. These section is summarized with four principles and several observations.

1. *The communication system must not inhibit the application.*

If this principle is strictly observed, the communication system can be thought of as one of the abstract functional units within a real-time distributed system. Real-time applications have

special constraints that must be met. The communication system must provide sufficient functionality and performance to be useful in the application in meeting these constraints without introducing errors into the environment. This includes being able to perform under a wide variety of load and traffic conditions. There must be a sharp division between the real-time application and the communication system: the application sets the constraints; the communication system, as much as any other functional unit within the real-time environment, must abide by these constraints.

2. *The functionality of the communication system is used with known costs, or does not impose cost upon applications that do not use them.*

Functional requirements vary among real-time applications. An application must be able to choose characteristics of service such as the degree of reliability. Applications that do not require robust service should not pay for the mere existence of a robust service option.

3. *The functional interface to the communication system must be well-defined.*

The communication system is an abstraction of functionality. This functionality must be encapsulated within the communication system module with only a finite and well-defined method of access. This aids both the use and maintenance of the communication system: the application may rely on the stability of the services while the actual implementation may be replaced and improved without affecting the application.

4. *The communication system must be a complete abstraction of the functionality required.*

All of the functionality required of the communication system must be provided within the communication system, completely relieving the application of any communication concerns. This includes all the mechanisms for implementing the functionality, such as sequence numbers or sliding windows. For real-time applications, it may be determined that

this should include a common time reference or a real-time scheduler.

Within these principles are observations that deserve note. Real-time applications generally require some degree of *reliability* from the communication system. The application demands *promptness* and *correctness* where so constrained. Furthermore, it is desirable that the communication system be *flexible* to varying conditions within the environment, and *adapt* to them.

8.2. Models of Communication

Local area networks are based on the packet switching. Packets are groups of data symbols which traverse the network as a unit. The *datagram* model of communication is based on the "best effort" delivery of these packets. Within the ISO OSI Reference Model, a datagram service is provided by the datalink layer and is used by the transport layer for its value-added services. The transport layer may also provide communication services based on this model, such as the ISO Connectionless-mode Transport protocol.

A datagram has several properties. It is an independent unit of data, so it has no inherent ordering within the context of other datagrams, which implies that the application can make no assertions about its delivery outside of the scope of the data contained within the datagram. Datagrams may or may not be acknowledged. An acknowledged datagram loosely synchronizes the transmitter and the receiver with the completion of the delivery of the datagram. Acknowledgement also helps recover from lost or corrupted datagrams. Datagrams are useful when a real-time application wishes to have an infrequent or one-shot communication with another application, or when the data is not order-dependent. Generally, datagram-based communication has low overhead.

The *virtual circuit* model is based on a datagram service but provides addition properties. The data delivered by a virtual circuit is guaranteed to be in order, thus an application may make assertions about the relationship between data packets. This relationship is called a message, which may theoretically be of arbitrary size. Internal mechanisms decompose a message into packets, deliver these packets with sufficient additional information for their subsequent reassembly, and present the message in its entirity to the receiving application. Consequently, messages are ordered as well. This service is useful for order-dependent communications, such as file transfer and some types of data collection. Furthermore, the delivery of data is guaranteed. Since data must arrive in order, the transmitter can detect lost packets within a time limit by lack of acknowledgement, and can therefore retransmit the packet. This makes virtual circuits attractive for essential communication. However, depending on the mechanisms employed, time constraints are difficult to impose. The worst case latency for a message depends on the number of retransmissions allowed (persistence count) and the delay per packet, which implies that it depends on the size of the message. Virtual circuits are called connection-oriented because a connection is established before data is sent.

A departure from datagrams and virtual circuits is the *message transaction* model. A message transaction is a request followed by a response. No connections are established. Conversations, however, are like connections, but they are the abstract notion of what happens between applications using message transactions. Clients send requests; servers send responses. The first request starts a conversation. The response is an implicit acknowledgement of the request. The next request in the conversation implicitly acknowledges the last response. A client may have only one transaction active at a time. Thus completely implicitly, this model implies guaranteed delivery in order without imposing its own connection upon the conversation. This model is designed for client/server relationships, such as those embodied by

remote procedure calls, file servers, and command/response components within a control hierarchy.

The classic ISO Transport protocol Class 4 is based on the virtual circuit model, and thus protocols like MAP that specify this transport protocol are as well. GAM-T-103 provides both datagrams and virtual circuits. XTP is only connection oriented with the datagram service provided by short-lived virtual circuits. VMPT is the only protocol examined that is based on the message transaction model.

8.3. Quality of Service

The quality of service is a general term that implies some characterization of the degree of robustness while using a service. Generally the quality of service is negotiated during the connection establishment, and includes such parameters as throughput or error rate. Some of these characteristics are difficult or impossible to know locally, especially at connection establishment, and are typically added as rough indications, not hard requirements.

Other quality of service parameters indicate what degree of reliability is to be used during the connection, and thus they refine the meaning of the functions used to affect the communication. This violates the well-defined interface principle. A better approach is never to negotiate quality of service; rather, the functions themselves should reflect the quality of service required. The ISO based protocols, such as MAP and Transport Class 4, employ a negotiation during connection establishment to set the quality of service for that connection. GAM-T-103 and XTP have no such negotiation during connection establishment. VMTP is based on message transactions; it does not establish a connection.

8.4. Error Recovery

It has been recognized that the error recovery strategies currently employed assume that the majority of errors are due to transmission error. With the improved state of the art of physical transmission, error rates in fiber optic LANs can be very rare (1 in 10^{14}). Error recovery techniques are recovering from packets lost due to overruns as a result of mismatched transmitter/receiver speeds. If the application process producing data is slower than the one consuming it, no problem exists. The overrun occurs if the consumer cannot empty receive buffers fast enough. Since it is irresponsible to overwrite existing data buffers, the packets must be dropped for lack of space. The transmitter will not know of this until its retransmission timer has timed out, signaling the loss of the packet. A retransmission thus occurs.

An obvious improvement is to somehow learn from the lost packet. Flow control via credits and windows can be used to throttle the transmitter, but it is suited to the needs of connection oriented services. A datagram service, for example, would not typically have the overhead of a sliding window. A more general mechanism is rate control which can adjust the transmission speed of the transmitter to match the capabilities of the receiver. Unchecked, overruns tend to occur periodically, so the rate control mechanism can use the length of the period to help determine the amount by which the transmitters rate must be reduced.

Flow and rate control have been used for end-to-end coordination of transmission rates, but for multi-segment LANs overruns can occur within the routers. Because the network layer is independent of the transport layer, packets going through routers cannot take advantage of the flow and rate control techniques one layer above. Thus ISO based protocols have no flow control between peer network layers and thus no mechanism for implementing congestion control in a router. One solution is the combination of the transport and network layer functionalities into a hybrid "transfer" layer. Now every packet requiring the services of the

network layer also gets the flow and rate control services of the transport.

ISO Transport Class 4 uses a sliding window and credits for flow control. GAM-T-103 uses a very simple sliding window without credits. It has a queue at the receiver that rejects packets if an overrun occurs, which induces retransmissions. XTP uses flow control between each node in the network, not just end-to-end. VMTP uses rate control and selective retransmissions. When a server accepts a VMTP request, it knows that the client has reserved the resources for the response.

8.5. Time Responsiveness

The communication system should not inhibit the application. This implies that, for real-time applications, the communication system must be prepared to ensure that delivery is accomplished with the reliability specified, and it happens within the deadline imposed by the application. None of the protocols examined can support this as they now stand.

All protocols miss the actual goal and substitute for it some alternative concept like priority. That may or may not be satisfactory. Controlling communication in a real-time environment requires that all the scheduling and queueing algorithms be cognizant of the message's needs. Only the application, however, is truly that knowledgeable. Perhaps the real-time constraints of messages can be reduced to a priority scheme. Even so, MAC layer protocols do not provide the scheme for ensuring real-time responsiveness. Timed token schemes such as those employed by FDDI and 802.4 do not provide strict priorities; in fact many lower priority messages may be transmitted before a high priority message. Priority reservation schemes such as those found in HSRB and 802.5 are expensive to implement. In the general case, the use of the priority scheme has been shown to increase overall message delay compared with using a single priority for all messages.

Transport protocols that attempt to offer priorities are typically limited and problematic. The expedited service of ISO Transport Class 4, for example, is weak in definition and undefined in implementation. Furthermore, it is unclear how such priorities are mapped to lower layer services, even if the priority is preserved. The implementor of the protocol is encouraged to use the expedited services of the lower layer to provide expedited service to the upper layers when available, but it is not required that the lower layers even have a priority scheme. Thus priority is generally an "intra-layer" concept.

The substitution of speed for time responsiveness is also apparent in the protocols examined, namely in XTP. Presumably, if a communication system is so much faster than the most time-constrained messages ever needed, then no scheduling or queueing algorithm need to be used to ensure time responsiveness. The flaw in this argument is the assumption that real-time applications have a "shortest deadline" time constraint, that it will never constrain a message to a deadline shorter, and that this shortest deadline is orders of magnitude larger than the worst case message latency. It is invalid to assume the existence of the shortest deadline.

8.6. The Irreducible Set of Functions

We cannot identify a single set of functions that in some way defines the core of a real-time communication system. The reason is that functionality required to support real-time necessarily depends on the requirements of the real-time application. Whether a function is absolutely required or is irrelevant is application-dependent. Real-time voice, for example, does not require very robust error recovery functionality; to include one would be ridiculous. If a packet carrying voice data is lost, it is more important to continue the stream rather than wait on retransmission of the packet.

8.7. The Best Real-Time Transport Protocol

It is clear that no one solution to real-time communication exists at present. There are too many details which are application-specific to be able to identify one general real-time communication system. Recalling that none of the protocols presented here are complete or perfect, we feel that two of them have the best chance of providing the functionality required of a communication system in a real-time environment. These two are VMTP and XTP. They are considered good possibilities for different reasons.

As expected, the MAC layer protocols themselves provide a service that is too basic and too low-level to be useful to a real-time system without that system moving a great deal of the functionality into the applications. The communication system can no long be thought of as an abstract functional unit with a well-defined functional interface; part of the functionality is within the application. The protocols based on the full seven layers of the ISO OSI Reference Model are being recognized as too general and too complex to serve real-time applications. The services provided are too sophisticated, and their mere presence may degrade performance. MAP/EPA suffers from both problems — the datalink layer does not have enough functionality and the application layer does not provide the proper kind. It is an attempt to take general purpose functionality in the form of seven layers and make it capable of supporting real-time by removing some of the layers. This approach is neither scientific nor correct. Finally, not enough is known about GAM-T-103. The contributions it has made are in realizing that the seven layer framework imposed by ISO OSI may not necessarily be the best place to start building a real-time communication system. Also, the TRANSFER layer concept has influence at least one other protocol, the XTP. Yet there is no reason to believe that GAM-T-103 actually fuses transport and network layer functionalities into one layer as it professes. It appears to limit the LAN to a single segment, thus simply eliminating the network layer rather than incorporating it into TRANSFER.

VMTP uses a different model for communication — the message transaction, and most of the arguments supporting this departure seem to hold. The message transaction may well be the natural way that real-time application communicate. The idea is simple and the implementation appears to be straight-forward, which also implies low overhead. Sufficient, yet natural, variations on the message transaction make the model an interesting topic for further research.

XTP is the only protocol with a reasonable amount of functionality expressly designed with a VLSI implementation in mind. If the VLSI implementation of XTP can sustain data transfer at the transport layer at the same rate as the underlying physical layer, then the distributed nature of the real-time system becomes transparent. This VLSI implementation is still in the future, and XTP continues to change. The proof is in the silicon.

BIBLIOGRAPHY

ANSA83 Ansaldi, W. and Olobardi, M., "A LAN Protocol for Real-Time Applications", *Proceedings of MELECON '83*, A1.06 Vol. 1, 1983.

CHER86 Cheriton, D.R., "VMTP: A Transport Protocol for the Next Generation of Communication Systems", *Computer Communications Review*, Vol. 16, No. 3, pp 406-415, August 1986.

CHER88 Cheriton, D.R., "VMTP: Versatile Message Transaction Protocol", *Protocol Specification*, Preliminary Version 0.6, 10 January 1988.

CHES87 Chesson, G., *The Protocol Engine Project*, UNIX Review, September, 1987.

CHES88 Chesson, G., *et alia*, "XTP Protocol Definition, Rev 3.1", *Protocol Engines Incorporated, PEI 88-13*, 1988.

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.

FDDI86 ANSI, *FDDI Token Ring Physical Layer Medium Dependent Standard*, draft proposed Standard X3T9.5/84-48, rev. 6, July, 1986; *FDDI Token Ring Physical Layer Protocol Standard*, draft proposed Standard X3T9.5/83-15, rev. 13, August, 1986; *FDDI Token Ring Media Access Control Standard*, draft proposed Standard X3T9.5/83-16, rev. 10, February, 1986; *FDDI Token Ring Station Management Standard*, draft proposed Standard X3T9.5/84-49, rev. 2, September, 1986.

GAMT87 *GAM-T-103 Military Real-Time Local Area Network*, Ministry of Defense, Republic of France, February 1987.

GENE87 General Motors Manufacturing Automation Protocol Committee, *GM MAP Specification*, Ver. 3.0, 1987.

GORA86 Gora, W., Herzog, U., and Tripathi, S.K., "Clock Synchronization on the Factory Floor", *Proceedings of the Workshop on Factory Communications*, National Bureau of Standards, pp. 87-108, March 17-18, 1987.

HWAN87 Hwang, K.W., Tsai, W.T., and Thuraisingham, M.B., "Implementing a Real-Time Distributed System on a Local Area Network", (Abstract), *Proceedings of 12th Conference on Local Computer Networks*, pp. 142, October 5-7, 1987.

IEEE84 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.

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.

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.

LELA85 Le Lann, G., "Distributed Real-Time Processing", *Computer Systems for Process Control*, pp. 69-90, 1985.

MINE87 Minet, P., Rolin, P., and Sedillot, S., "Discussions about MAP Adequacy to Meet Hard Real-Time Communication", *Proceedings of the Seventh IFAC Workshop*, Distributed Computer Control Systems, 1987, pp. 53-56.

MINN88 Minnich, D.W., *Performance Analysis of SAE AS4074.2 High Speed Ring Bus*, Master's Thesis, Department of Computer Science, University of Virginia, expected August, 1988.

MIRA83 Mirabella, O., "Real Time Communications in Local Area Network Environment", *Proceedings of MELECON '83*, A1.07 Vol. 1, 1983.

PEDE87 Peden, J.H., and Weaver, A.C., "Performance of Priorities on an 802.5 Token Ring", *ACM SIGCOMM*, Stoweflake, Vermont, August 1987.

PEDE88 Peden, J.H., and Weaver, A.C., "The Utilization of Priorities on Token Ring Networks", *13th Annual Conference on Local Computer Networks*, Minneapolis, Minnesota, October, 1988.

SAE87 Society of Automotive Engineers, *SAE AS4074.2 High Speed Ring Bus*, Final draft Standard, June, 1987.

SEVC85 Sevcik, K.C., and Johnson, M.J., "Cycle Time Properties of the FDDI Token Ring Protocol", *RIACS report number TR-85.10*, August, 1985.

SIMO88 Simonson, R.A., *Performance Analysis of FDDI*, Master's Thesis, Department of Computer Science, University of Virginia, May, 1988.

SONG87 Song, J.S., *et alia*, "An Implementation of Reliable Network Controller for Real-Time Application", *COMPCON Spring '87*, IEEE Computer Society, 1987, pp. 293-299.

STOI88 Stoilov, E.I., "Coordination mechanisms in Local Area Networks for Real-Time Applications", *Computer Communications Systems*, IFIP, 1988.

STRA88a Strayer, W.T., and Weaver, A.C., "Performance Measurement of Data Transfer Services in MAP", *IEEE Network*, Vol. 2, No. 3, May 1988.

STRA88b Strayer, W.T., and Weaver, A.C., "Performance Measurement of Motorola's Implementation of MAP", *Computer Science Report No. TR-88-7*, University of Virginia, June 6, 1988.

TANE81 Tanenbaum, A.S., **Computer Networks**, Prentice-Hall, Inc., 1981.

ZNAT87 Znati, T., and Ni, L.M., "A Prioritized Multiaccess Protocol for Distributed Real-Time Applications", *Proceedings of the 7th International Conference on Distributed Computing Systems*, pp. 324-331, 1987.