

AD-A274 941



NAVAL POSTGRADUATE SCHOOL
Monterey, California



THESIS

S

DTIC
ELECTE
JAN 24 1994

E

D

**DESIGN AND SPECIFICATION OF THE XPRESS TRANSFER
HIGH-SPEED PROTOCOL**

by

David Joseph Sacha

September 1993

Thesis Advisor:

Dr. G. M. Lundy

Approved for public release; distribution is unlimited

94-01975



94 1 21 146

REPORT DOCUMENTATION PAGEForm Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE September 1993		3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE Design And Specification Of The Xpress Transfer High-Speed Protocol (U)				5. FUNDING NUMBERS	
6. AUTHOR(S) Sacha, David Joseph					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/ MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSORING/ MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.					
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited				12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) The use of fiber optics in high-speed data networks has significantly increased throughput and reliability at the physical layer. Consequently, the transport layer has become a bottleneck to the data transfer potential of high-speed networks. This bottleneck has forced an investigation of transport protocols and standards to be used in future networks. The Xpress Transfer Protocol (XTP) is a transport layer protocol designed to perform efficiently in networks where high data rates, densely packed bit pipes and low bit error rates are normal operating conditions. However, XTP is a relatively new protocol which has not yet undergone extensive testing and analysis to verify its ability to resolve the transport layer bottleneck. In this thesis the specification and analysis of the XTP protocol, using the System of Communicating Machines (SCM) model is presented. A comparison is then made with an alternative high-speed protocol called SNR, originated at AT&T Bell Labs. Based on this study, it is concluded that the XTP protocol provides several mechanisms, such as rate control and extended sequence numbering, that should be included in developing high-speed transport protocols. Furthermore, it is concluded that XTPs flexible characteristics allow for multiple paradigm implementations at the cost of some complexity, making a more complete analysis of this protocol difficult. As work on high-speed transport protocols continues many of the XTP mechanisms should be considered for inclusion into evolving standards. It is also concluded that there are some critical features for high-speed protocols that are not in XTP. The ideal transport protocol should include these other features such as implicit timers and data blocking.					
14. SUBJECT TERMS Transport Protocol, Network Protocol, Transfer Protocol, Xpress Transfer Protocol, Data Communications, Protocol Analysis, Protocol Specification, System of Communicating Machines				15. NUMBER OF PAGES 101	
				16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT Unlimited		

Approved for public release; distribution is unlimited

**DESIGN AND SPECIFICATION OF THE XPRESS TRANSFER
HIGH SPEED PROTOCOL**

by
David Joseph Sacha
Captain, United States Army
B. S. General Engineering, United States Military Academy, 1983

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

September 1993

Author:



David Joseph Sacha

Approved By:



Professor G. M. Lundy, Thesis Advisor



Professor Lou Stevens, Second Reader



Dr. Ted Lewis, Chairman,
Department of Computer Science

ABSTRACT

The use of fiber optics in high-speed data networks has significantly increased throughput and reliability at the physical layer. Consequently, the transport layer has become a bottleneck to the data transfer potential of high-speed networks. This bottleneck has forced an investigation of transport protocols and standards to be used in future networks. The Xpress Transfer Protocol (XTP) is a transport layer protocol designed to perform efficiently in networks where high data rates, densely packed bit pipes and low bit error rates are normal operating conditions.

However, XTP is a relatively new protocol which has not yet undergone extensive testing and analysis to verify its ability to resolve the transport layer bottleneck.

In this thesis the specification and analysis of the XTP protocol, using the System of Communicating Machines (SCM) model is presented. A comparison is then made with an alternative high-speed protocol called SNR, originated at AT&T Bell Labs.

Based on this study, it is concluded that the XTP protocol provides several mechanisms, such as rate control and extended sequence numbering, that should be included in developing high-speed transport protocols. Furthermore, it is concluded that XTPs flexible characteristics allow for multiple paradigm implementations at the cost of some complexity, making a more complete analysis of this protocol difficult. As work on high-speed transport protocols continues many of the XTP mechanisms should be considered for inclusion into evolving standards. It is also concluded that there are some critical features for high-speed protocols that are not in XTP. The ideal transport protocol should include these other features such as implicit timers and data blocking.

DTIC QUALITY INSPECTED 3

Accession For	
NTIS	CRA&I ☒
DTIC	TAB ☐
Unannounced ☐	
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	BACKGROUND	1
B.	OBJECTIVES OF THESIS	2
C.	APPLICABILITY OF THESIS.....	3
D.	ORGANIZATION.....	4
II.	EMERGING HIGH-SPEED DATA NETWORKS	5
A.	BASIC DEFINITION AND MODELS OF DATA NETWORKS	5
1.	Definition Of High-Speed Data Networks.....	5
2.	Basic Network Models.....	7
B.	NETWORK AND TRANSPORT LAYER REQUIREMENTS	11
1.	High-Speed Network Protocol Functions	12
2.	High-Speed Transport Protocol Functions.....	12
C.	NETWORKS FOR HIGH-SPEED TRANSPORT PROTOCOLS	12
1.	WANs.....	13
2.	MANs.....	14
3.	LANs.....	15
D.	TRANSPORT PROTOCOLS FOR HIGH-SPEED NETWORKS	16
1.	TCP/IP.....	17
2.	SNR.....	18
3.	XTP	18
E.	PROBLEMS FACING EMERGING TRANSPORT PROTOCOLS	18
III.	THE XPRESS TRANSFER PROTOCOL.....	22
A.	BACKGROUND	22
B.	BASIC CONCEPTS	23
C.	INFLUENTIAL PROTOCOLS.....	25
1.	TCP	26
2.	TP4	26
3.	Delta-T	26
4.	Network Block Transfer.....	27
5.	GAM-T-103	27
6.	VMTP.....	27
7.	Datakit Architecture.....	27
D.	NEW FUNCTIONALITY IN XTP	28
1.	Multi-cast	28

2.	Priorities	28
3.	Rate and Burst Control.....	28
4.	Address Translation	29
5.	Retransmission	29
6.	Acknowledgment Control.....	29
7.	Out of Band Data	29
8.	Data Pipeline Size	30
9.	Connection Services.....	30
10.	Alignment.....	30
E.	XTP STRUCTURE	31
1.	Basic Packet Structure	31
2.	Header and Trailer Segments.....	31
3.	FIRST Packet Structure	33
4.	DATA Packet Structure	34
5.	Control (CNTL) Packet Structure	34
6.	Other Packet Structures.....	35
7.	Timers	36
F.	BASIC PROTOCOL PROCEDURES	37
1.	Association Management Procedures	37
2.	Path Maintenance Procedures	38
3.	Data Transfer Procedures.....	39
4.	Flow Control Procedures	39
5.	Rate Control Procedures	40
6.	Error Control Procedures	40
IV.	FORMAL SPECIFICATION OF XTP	41
A.	SYSTEM OF COMMUNICATING MACHINES (SCM)	41
B.	SPECIFICATION STRUCTURES	43
1.	Communication Structures.....	43
2.	Message Types.....	48
C.	PROTOCOL SPECIFICATION AND PROCEDURES OF XTP	49
1.	Association Establishment Procedure.....	49
2.	Data Transfer Procedure	52
3.	Data Transfer Procedure with Flow Control.....	55
4.	Data Transfer Procedure with Orthogonal Rate Control	60
5.	Data Transfer Procedure with Error Control - Selective Repeat	64
V.	ANALYSIS OF XTP	68
A.	ASSOCIATION ESTABLISHMENT.....	68

B.	DATA TRANSFER.....	71
1.	XTP Error Control - Selective Repeat	72
VI.	COMPARISON TO SNR.....	76
A.	ADDITIONAL XTP HIGH-SPEED CAPABILITIES	76
B.	SUMMARY OF SNR.....	77
1.	Periodic Exchange of State Information	78
2.	Selective Repeat for Retransmission and Data Blocking.....	78
3.	Parallel Processing	78
C.	COMPARISON OF SNR AND XTP.....	79
1.	Increased Performance Through Parallel Processing.....	79
2.	State Information and Lost Control Packet Information.....	80
3.	Error Control and Acknowledgment Of Received Data	81
4.	Connection Management	82
5.	Implementation	82
VII.	CONCLUSION	85
A.	SUMMARY OF RESEARCH.....	85
B.	CONTRIBUTIONS OF THIS THESIS	86
C.	FURTHER RESEARCH OPPORTUNITIES	86
	REFERENCES	88
	INITIAL DISTRIBUTION LIST	92

LIST OF FIGURES

Figure 1.	OSI Reference Model	9
Figure 2.	TCP/IP Implementation Model.....	10
Figure 3.	BISDN ATM Protocol Reference Model	11
Figure 4.	Data/Highway Comparison.....	19
Figure 5.	XTP Host Architecture	24
Figure 6.	XTP Peer to Peer communication Model	25
Figure 7.	XTP Basic Packet Structure.....	31
Figure 8.	Header and Trailer Packet Structures	32
Figure 9.	Middle Segment of a FIRST packet	33
Figure 10.	Middle Segment of a DATA packet	34
Figure 11.	Middle Segment of a Control (CNTL) Packet.....	35
Figure 12.	SCM, two machine behavior representation.....	42
Figure 13.	XTP Association	43
Figure 14.	Association Establishment Timing Diagram	49
Figure 15.	Association Establishment State Transition Diagram	51
Figure 16.	Data Transfer Timing Diagram.....	53
Figure 17.	Data Transfer State Transition Diagram.....	54
Figure 18.	Data Transfer Timing Diagram with Flow control	57
Figure 19.	Data Transfer State Transition Diagram with Flow Control	58
Figure 20.	Data Transfer Timing Diagram with Rate Control	61
Figure 21.	Data Transfer State Transition Diagram with Rate Control	62
Figure 22.	Data Transfer Timing Diagram with Error Control	65
Figure 23.	Data Transfer State Transition Diagram with Rate Control	65
Figure 24.	SCM Model for Association Establishment	69
Figure 25.	SCM, Global Reachability Analysis for Association Establishment.....	70
Figure 26.	SCM, System Reachability Analysis for Association Establishment.....	70
Figure 27.	Selective Repeat Specification.....	74

I. INTRODUCTION

"The role of the federal government remains critical to the national information structure: to establish standards and protocols... (and that)... we establish much higher thresholds of data transmission, so that we can encourage the evolution of new classes of information services that are presently beyond our imagination."

- Al Gore, Vice President of the United States
San Jose Mercury News
11 September 1993

A. BACKGROUND

In today's Information Age there is an ever increasing need to transfer and exchange data as a commodity [MINO91]. Currently high-speed computer communications are somewhat capable of delivering this data transfer service between selective users. However, there is a growing appetite for increased services, accessible to more and more users that range from government and industry to the private homeowner and small business operator [MOLL88]. For example, the use of distributed databases and electronic mail are becoming widely used and accepted [MINO91] and will soon add integrated voice and video applications as well. High-speed networks will naturally create and be forced to solve problems of increased system demand and performance requirements through the optimization of throughput and end-to-end latency, where greater capacity and reduced round trip response times are the goal.

The use of high-speed physical networks and systems alone will not solve the problems described. Although it is true that fiber optics, advanced high data rate twisted pair wire technology, reduced instruction set processors (RISC) machines and optical computing research [ARNO93] promise to deliver faster physical network mediums and hardware processing computing resources, there is still a dependency on sophisticated software and protocols to control and ensure the reliable delivery of data. The hardware component of high-speed systems can transmit and receive data much faster than the software implementing the associated communication protocols [MOLL88]. Several

protocols, designed to operate at higher speeds, have been suggested to replace the current software. Current literature refers to these more efficient protocols for high-speed physical mediums as lightweight or high-speed transport protocols. The approach taken to design or develop high-speed protocols is quite different from that used in developing past protocols in that the design process is centered upon optimizing performance towards successful packet transmission, rather than toward building robustness to compensate for transmission failure. [MCAR92]

Specific proposed applications and high-speed data networks or architectures that may benefit from advanced transport protocol design include: the U.S Navy's SAFENET project using the fiber distributed data interface (FDDI) [HIGH93]; the U.S. Army Battlefield Information Architecture [BREN92]; the Naval Postgraduate School Networked Vehicle Simulator IV (NPSNET-IV), the Defense Simulation Internet (an advanced simulation program funded by the Department of Defense that has evolved from SIMNET), Asynchronous Transfer Mode (ATM) based networks, metropolitan area networks and networks using very small aperture satellite (VSAT) systems. Each of these networks or technologies have potential high-speed data requirements to support advanced user applications and demands for increased performance standards for reliable data delivery.

B. OBJECTIVES OF THESIS

The major objective of this thesis is the formal specification of the Xpress Transfer Protocol (XTP) described in [STRA92b], an analysis of that protocol when modeled using the System of Communicating Machines (SCM) [LUND91b] and finally a brief comparison of XTP to the SNR (named after the protocol authors Sabani, Netravali and Roome, AT & T Bell Labs) protocol described in [MCAR92]. The formal specification will help us show that the transport layer functions of XTP are viable for high-speed data transfer. It should be noted that while XTP also includes network layer functions, the

specification and analysis of XTP network layer functions is intentionally not included in this paper.

A secondary objective of this thesis is to survey emerging high-speed data networks. The survey exposes the breadth of high-speed transport networks and protocols and serves to substantiate the need for continued research in this area. In the survey potential networks for the deployment of protocols such as XTP are discussed. The discussion on transport layer innovations and issues clearly indicates that improved protocols will indeed increase the efficiency of high-speed networked systems.

C. APPLICABILITY OF THESIS

The applicability of this thesis is primarily in three areas. First, background on high-speed data protocols and the description of one such protocol, XTP is presented. Second, the specification and analysis of the XTP high-speed transport protocol. Although simplified this will show that XTP can perform the transport layer functions required of a high-speed protocol suite. And finally, a comparison of XTP to a similar effort described in [MCAR92]. The task at hand is to look at the transport layer functions of XTP and to undertake an investigation of XTPs capability. This investigation will show that XTP, in particular, is one protocol that purposely addresses the issues of high-speed protocols.

A limited formal specification of the XTP protocol, using the SCM model, is developed from two works, [PEI92] and [STRA92b]. The formal specification in this thesis is limited by assumptions that simplify the specification and the complexity of the protocol. These assumptions, in general, are decisions that an XTP system designer or implementor would make, and will be identified. Each assumption has at least one alternative choice that would likely yield a different formal specification. The cognitive summation of all limited formal specifications would yield a complete formal specification for XTP. The limited specification for connection management and data transfer are presented for connection oriented paradigm. The protocol analysis will either identify logical errors or show that the

protocol is free from logical errors, such as deadlock, unspecified reception, unexecuted transitions and blocking loops.

D. ORGANIZATION

This thesis is organized into seven chapters. In the second chapter, we survey specific emerging networks that utilize high-speed data transport protocols and the transport layer problems of these networks. Chapter III describes a single protocol, XTP, and introduces XTP protocol procedures.

Chapter IV describes the SCM model introduced in [LUND91b] and uses it to complete a limited formal specification of the XTP association establishment and data transfer procedures. An analysis based on the previous chapter is conducted in Chapter V.

Chapter VI includes a summary of the SNR transport protocol, with a comparison between it and XTP. Chapter VII is a summary of the research and major contributions in addition to a discussion of potential research opportunities.

II. EMERGING HIGH-SPEED DATA NETWORKS

High-speed networks are rapidly becoming accessible to more and more end users. In this chapter a working definition of high-speed networks is presented with a discussion of models used to describe networks. Then the basic functions of the network and transport layers that must be present in high-speed networks are presented. Next, an overview of evolving high-speed networks gives the reader an idea of just how expansive the need for efficient high-speed network and transport protocols will be. In the transport protocol section of this chapter, three emerging high-speed network protocols are presented to give the reader a sense of the different approaches to same problem. Finally, those problems facing high-speed networks are discussed.

A. BASIC DEFINITION AND MODELS OF DATA NETWORKS

1. Definition Of High-Speed Data Networks

High-speed computer communications and high-speed packet switched networks with extensive switching and gateway operations represent a significant change in the way which computer technology is applied. In the past, ARPANET was once one of the best networks a user could access. By today's standards, ARPANET was a low-speed network and provided data transmission rates on the order of 56kbyte/second without protocol overhead. Today's INTERNET is a conglomerate of many local area networks interconnected through extensive wide area networks with transmission links in the megabyte range. With this underlying transmission rate INTERNET frequently sees end-to-end throughput on the order of 30kbps for common file transfer program executions. This is a substantial increase in throughput over ARPANET considering that this value also includes protocol overhead. Yet even these rates are considered slow when potential rates on fiber optic systems are considered. So ARPANET and INTERNET are good examples of low speed networks. A common characteristic of low speed networks is a transmission path bottleneck, where limited available bandwidth on the physical transmission medium,

among competing users, creates inefficient network usage and decreases overall throughput rates when compared to the underlying available transmission rates. The quest to reach high-speed performance is one of the INTERNET's challenges of the 1990s.

However, the exact definition of a high-speed data networks, in terms of data rates, is imprecise. For purposes of discussion an arbitrary lower bound of 50Mbps is placed on end-to-end transmission at the physical layer for a high-speed network. At the 50 Mbps data rate the first indications of the network outperforming host processing capabilities are experienced. Although this value may seem fast to some, in terms of Gigabit and Terabit rates forecasted for the future it is quite slow. The limit of 50Mbps applies to that level of the network which is transmitting raw data bits, (e.g. the physical layer of the OSI model, described below) and ensures the inclusion of FDDI, DQDB, broadband integrated services digital networks (B-ISDN) and the synchronous optical networks (SONET) OC-1 transmission rate of 51.840 Mbps [MINO91]. A higher threshold could easily have been chosen that drew the high-speed line at advanced fiber optic rates above 100 Mbps. This discussion does not imply that fiber optic transmission is necessary for a high-speed data network, it merely demonstrates that the fiber mode is commonly found carrying these data rates.

Another perspective available to help define high-speed networks is to consider the system from the host processing perspective. If the host processing system (this includes both the operating system and protocols mounted on the host computer) is a bottleneck to system throughput because the network is faster then the network is high-speed. On the other hand, if the transmission media is the network bottleneck, then the network is not high-speed.

These definitions apply equally to telecommunication networks and data networks. The former, telecommunication networks, grew out of the plain old telephone system (POTS) and are typically wide area networks. Data networks grew from direct computer communication links and are considered a natural subset of the telecommunication networks field. In fact, when discussing digital transmissions the

distinction between these two fields is becoming increasingly blurred as networks emerge. This *blur* is actually a strong trend towards the integration of voice, data, and video, referred to as B-ISDN (broadband integrated services digital networks) over interconnected networks of varying geographical characteristics. This completes the discussion of what is meant by high-speed, but what of the definition of a network?

The simplest definition of a network is acceptable for this discussion. A network is a system where, given two or more entities, A and B, there is information transfer from A to B and vice versa. The two entity network can be extended to include additional entities that can then mutually communicate with the existing network. In the case of a telecommunications network there is an added equipment suite consisting of terminating, transmission and switching equipment. Protocols are the agreed upon conventions between communicating entities on how to carry out the mechanics of the communication process. [MINO91]. These protocols exist at varying level within a protocol model. Basic model are discussed next.

2. Basic Network Models

Describing the requirements of a data communications network is a complex task. It is even more so when high-speed data with increased throughput is considered. Abstract models to describe such networks can ease the task of specifying requirements and provide a framework for standardization. Three such models, the Open Systems Interconnection (OSI) Reference Model defined in ISO 7498, the internet protocol suite and a broadband data model are discussed. In general, the models divide the communicating processes into entities or intermediate layers. These models provide a tool with which high-speed data networks can be examined and are of fundamental importance to the study of data and computer communications for providing an architecture within which protocol standards can be developed and explained [STAL91].

a. OSI Model

The OSI model is a seven layer protocol description that provides a common basis for the coordination of standards development for the purpose of systems interconnection. The model consists of the following seven layers: physical, data link, network, transport, session, presentation and application. The following definitions, up to layer four, provide insight into the basic requirements at the particular layer described, layer three and layer four provisions will be applied to our discussion of high-speed data networks. In Figure 1 a logical connection at layer four and a physical connection at layer one are shown.

Layer 1 - Physical Layer. The physical layer is concerned with the transmission of unstructured bit streams over the physical link. It deals with such parameters as signal voltage and electrical and optical procedural characteristics. It is primarily at this layer that the gains of fiber optic technology have initially made the most dramatic effect and increase in raw data bandwidth.

Layer 2 - Data Link Layer. The data link layer provides reliable transfer of data across the physical link. It sends block or frames of data with the necessary synchronization, error and flow control.

Layer 3 - Network Layer. The network layer provides the upper layers with independence from the data transmission. A significant feature of the network layer is the routing or switching capability which directs protocol data units or packets to the next switch, gateway or host on the network.

Layer 4 - Transport Layer. The transport layer provides reliable transfer of data between endpoints (across more than one link) in addition to end-to-end error and flow control. In doing this the transport layer shields upper layers of the protocol stack from the intricate operations of the lower three levels. Additionally, the transport layer provides the means to establish and terminate network sessions.

Layers 5-7. Layers 5 through 7 make use of the transport layer and thus the other lower layers of the model and are titled the session, presentation and application layer

respectively. Transport layer support for upper layer functions as well as the layers themselves is not addressed in this paper.

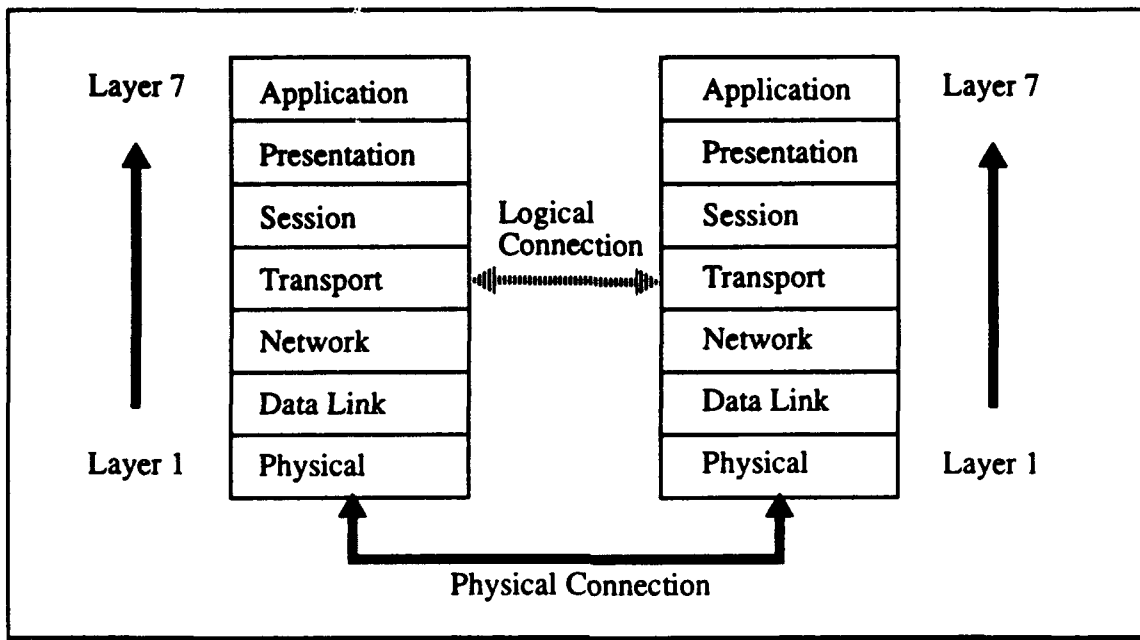


Figure 1: OSI Reference Model

The focus of this paper is on the transport layer protocol, also called the end-to-end or host to host protocol. Well known examples of an OSI reference model transport protocol include the ISO transport protocol class 4 (TP4) protocol and the Transmission Control Protocol (TCP). Both are responsible for reliable end-to-end data communications.

b. TCP/IP Internet Protocol Suite

There are other models in use for data networks that do not subscribe completely to the OSI reference model that may also apply to high-speed networks. One such protocol is the Transmission Control Protocol/ Internet Protocol (TCP/IP) Protocol Suite [COME91]. The TCP/IP Protocol Suite, which is best known by the TCP/IP implementation, must in both the model and application support similar functional requirements for reliable end-to-end communications as the OSI protocol suite. In [CLAR89] it is shown that the TCP/IP implementation can deliver data at rates nearing 530 Mbps, assuming the underlying physical layer could support this data rate. This bodes well

for the model as a potential for further investigation as a high-speed model. The TCP/IP implementation in most communities has become the interoperability standard of choice, due largely to its proven capability, availability and vendors willingness to write off development of TCP/IP technology against their entire product line [ECKE92]. It is therefore a logical starting point for those wishing to upgrade to high-speed capability. A diagram of the model from [COME91] is shown below in Figure 2. However, from an academic point of view the TCP/IP suite and OSI models both provide abstract tool for protocol description. It is the opinion of the author that OSI reference model is the preferred model since it clearly identifies all layers, this helps to describe functions of the protocol.

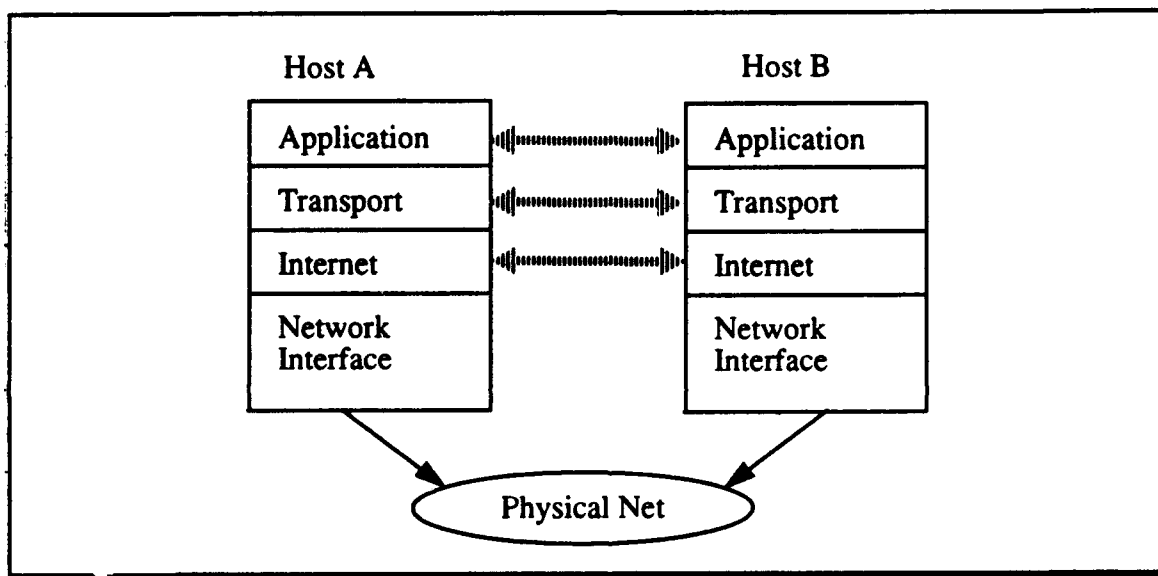


Figure 2: TCP/IP Implementation Model

c. *ISDN and BISDN Protocol Models*

There are emerging high-speed data communication technologies such as Integrated Services Digital Networks (ISDN) and Broadband ISDN (BISDN) that are good examples where the OSI reference model is used to assist in the development of new abstractions to handle the increased complexity of changing communication systems. The ISDN and BISDN protocol models are consistent with the OSI reference models as seen below in Figure 3. The layered approach applies to BISDN bearer services (OSI reference

model layers 1 to 3) and BISDN tele-services (OSI reference model layers 4 to 7).[KANO91] and [MINO91]

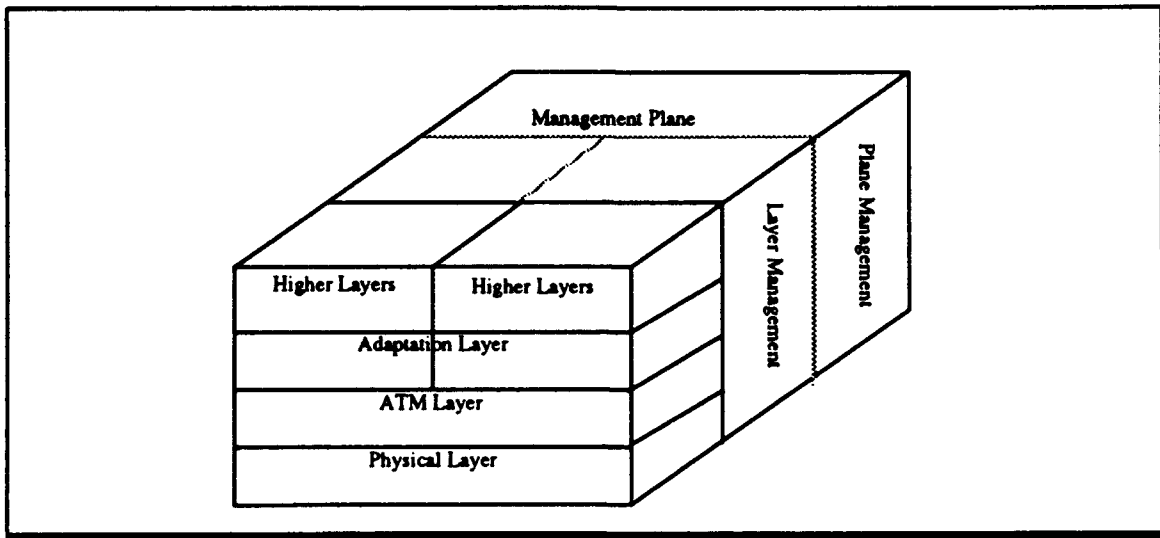


Figure 3: BISDN ATM Protocol Reference Model

The Asynchronous Transfer Mode (ATM) mentioned in the above figure is a high bandwidth, low-delay, packet-based switching and multiplexing technology that will support high-speed data communications at the equivalent to the data link layer.

B. NETWORK AND TRANSPORT LAYER REQUIREMENTS

The OSI model prescribes that the network and transport layer protocols meet certain basic requirements and provide for appropriate functionality. Reliable end-to-end data communications are the most important operations. Other critical functions include addressing, segmentation and reassembly, connection establishment and termination, flow control, rate control, error control and special data transfer services when appropriate [STRA92b]. A look at these requirements, specifically related to the appropriate level follows. When applied to high-speed systems the above listed critical function take on even greater importance when eliminating inefficiencies in a network. The requirements of both of these protocol layers are best seen in their functions.

1. High-Speed Network Protocol Functions

The network level provides the upper layers with independence from the data-transmission and switching technologies used to connect systems. The layer is responsible for establishing, maintaining and terminating connections [STAL91]. Network layer functions include: definition of global addressing space; routing algorithms; inter-networking; congestion control strategy; error handling and fragmentation and reassembly. Some of these functions are similar to those of the transport layer, but are applied at a level in the protocol one lower than the transport layer.

2. High-Speed Transport Protocol Functions

The basic requirement for the transport layer is to shield upper level applications from complexities of the underlying network, data link and physical layers. The transport layer provides transparent transfer of data between endpoints in addition to end-to-end error recovery and flow control [STAL91]. The most critical operation of the transport layer is end-to-end delivery of messages between hosts. This simply stated operation implies other functions that must be successfully performed or the protocol will fail. In [MCAR92], four basic transport layer functions are discussed: connection management, acknowledgment of received data, flow control, and error detection and recovery. As mentioned earlier, [STRA92b] adds protocol functions of addressing, segmentation and reassembly, and rate control to the [MCAR92] list of protocol requirements.

The transport layer is also perhaps the most complex layer. It must satisfy a wide variety of application requirements and at the same time accommodate a wide array of network characteristics. The networks may be reliable or unreliable, connection oriented or connectionless and impose varying degrees of quality of service.[BLAC91]

C. NETWORKS FOR HIGH-SPEED TRANSPORT PROTOCOLS

A short survey of target networks where high-speed transport protocols will be used will indicate just how widespread the need is for improved performance transport protocols. Generally, networks can be described as wide area (WAN), metropolitan area

(MAN) and local area (LAN) networks. The common thread amongst each of these network types is a user base that is demanding interconnected systems with more throughput and faster response time. Surely the user doesn't much care if the "distant-end" computer is across the country (WAN), across the city (MAN) or across the hall (LAN). The user is just concerned with being able to complete the computer task at hand and if that means more throughput and lower latency is required, then so be it (an example that could use all three network types would be the transfer of real-time images on a video phone connection). The role of a high-speed transport protocol would be to give this user the desired performance and to do it transparently to that user. Stated differently this means the transport protocol must provide reliable end-to-end data transfer with a latency that is fast enough (a sometimes hard to value quantify) to service the user application. But today protocols take large amounts of overhead at various layers of the OSI model. At the transport layer there is potential to decrease this overhead and other factors that lead to poor performance. We now look at WANs, MANs and LANs.

1. WANs

Wide area networks are increasingly becoming the survival links of large and small organizations alike. These networks have the potential capability of connecting large number of users across extensive distances. Typically, WANs are fiber optic based systems with large amounts of data carrying capacity. A prime example of high-speed data network is seen in the SONET specification and related broadband technologies.

a. SONET

SONET or synchronous optical network is truly designed to for taking advantage of the high-speed digital transmission capability of optical fiber. The base data rate of SONET is 51.84 Mbps, which can carry the existing DS3 standard. Currently the upper limit for the SONET data rate is 2.488 Gbps. Obviously, at such high data rates there will be increased performance stress on switching, host processors and the communications protocols at each layer of data handling above the physical layer.

b. *BISDN*

Broadband Integrated Services Digital Networks will take advantage of the large physical data highways provided by SONET and fiber technology. As seen in the earlier discussion on protocol models, BISDN, using a layered approach has set the groundwork for high-speed WANs. The BISDN model in Figure 3 clearly shows this layered approach.

c. *ATM layer*

The ATM layer is a critical layer to the BISDN model and future WANs. It is a connection oriented technique that provides virtual channel connections for the underlying physical layer through high bandwidth, low-delay packet based switching and multiplexing. Higher layer mechanisms and protocols must interface with ATM and provide reliable end-to-end communications [HAND91].

It is also interesting to note that ATM is considered for use in third generation LANs [ABSX92]. This increases the blur discussed before and shows a trend where WANs, MANs and LANs are used together in a seamless network to support multimedia applications, a goal discussed in [BREN92] for military communications.

2. MANs

Metropolitan area networks also represent a potential target market for high-speed data networks. Many such networks are in existence today that interconnect major universities as well as corporations with operations spread throughout a single geographical area. Two primary technologies emerging for MANs are the fiber distributed data interface (FDDI) and dual queue dual bus (DQDB). Both systems will support high-speed interconnections between workstations, minicomputers, mainframes and associated peripherals [MINO91]. MANs will interconnect LANs, high-speed computer links and voice and video transmissions.

a. DQDB

DQDB is well suited to public access shared telecommunication facilities in a metropolitan area and supporting high-speed MANs. This type of network, based on fiber optic technology, uses a dual unidirectional bus credit based transmission scheme to gain high throughput and low latency with a shared medium access protocol. The DQDB dual bus in a public network is physically a star but logically a dual bus.

b. FDDI

FDDI is also well suited to high-speed MAN networks, where the customer owns the right of way to lay the fiber optic cable. The FDDI standard covers the first two layers of the OSI model discussed earlier. This leaves the implementation of layers three and four to the network designer. FDDI is designed to interface with both OSI and TCP/IP protocol stacks. Such protocols, improved to handle high-speed data are necessary to create a network environment that will support functions such as electronic video mail and faster file transfer.

3. LANs

LANs represent the third category in which high-speed data networks are emerging. LAN Networks in many cases will be expected to inter-operate with the emerging WANs and MANs already discussed. In a typical LAN, the actual capacity may be as low as 1Mbps, while the physical layer operates at 10 Mbps. The reason for this is the protocol overhead at each layer and the processing time at a host required to handle that overhead. [MINO91]

a. FDDI Broadband LAN

(1) The FDDI high-speed MAN solution is also seen as an excellent LAN solution when applied to a smaller geographical area. It could provide a 100 Mbps backbone to lower speed FDDI LANs. This interconnection of LANs and MANs will

increase the FDDI as a means of networking high-powered computing devices that require a high degree of fault tolerance and data integrity [MINO91].

(2) The survivable adaptable fiber optic embedded network (SAFENET), a Department of the Navy sponsored project is an excellent example of how fiber optics is being implemented as a solution to a current high-speed data network problem. This network, using FDDI at OSI layers one and two and an combination of protocols at the network and transport layers will provide an advanced, survivable high-speed network communications for mission critical computer resources [HDBK92].

b. ETHERNET

As industry looks to new technologies for implementing high-speed data communications there are traditional LAN protocol solutions that, while designed for today's ethernet, may offer performance and data rates meeting tomorrow's high-speed network requirements.

Once such re-emerging network design is the ethernet. The ethernet, a longtime LAN standard based on carrier sense multiple access/ collision detect (CSMA/CD) technology can now attain high-speed data rates comparable to those of FDDI using shorter distances. This advance in Ethernet truly brings it into the world of high-speed networking, while still employing the basic philosophy of keeping the protocols simple [MOLL93].

D. TRANSPORT PROTOCOLS FOR HIGH-SPEED NETWORKS

As discussed, there is a wide range of network types that do or will soon support high-speed data communications. Standards for these networks must evolve to allow for the existence of inter-operable systems. These standards must range from the physical to the application layers of the network user. Focusing on the need of commonality of service to the user requires an investigation of the network and transport layers of the OSI protocol model. It is at these layers that protocols "hide" the physical attributes of lower layers of

the multiple physical high-speed network types possible and present a common interface for application layers above.

Although the OSI model is not required for an investigation of high-speed data communications, it provides an excellent framework from which to discuss performance and capabilities of such networks. There are several high-speed transport protocols either in use or under development today. Three such protocols are chosen to demonstrate the differences in the approach and key features of developing future high-speed transport protocols. The first, XTP, is chosen, as an example of a protocol that has the advantage of looking at existing protocols or those in development as a jumping off point for its own development. The second, SNR, had similar advantages as XTP, but applied differently the mechanisms for implementing solutions to the classical transport layer protocol problems. And finally, the TCP/IP protocol suite, which has become the de-facto standard for much of today's computer communications and has potential to re-emerge with improved mechanisms for high-speed performance [JACO92].

Summarizing the options, the choices are to a) upgrade the existing protocols so as to develop a high-speed solution that modifies the existing features of today's protocol and depends upon skillful implementation to improve efficiency or b) design a new protocol with all the required functions and make execution efficiency an inherent part of the design. [STRA92b] The three high-speed transport protocols are presented below.

1. TCP/IP

TCP/IP is the de-facto medium speed standard for today's internet, it provides for reliable byte stream transmission over a full duplex virtual connection. However, TCP/IP is also an emerging high-speed protocol. It is rapidly becoming a target as an economical solution to the problems of high-speed data transfer. There are a number of studies and papers that describe how the TCP/IP protocol suite can be upgraded to handle the requirements of tomorrow's high-speed data pipes [CLAR89]. However, others within the data communication community feel that there is a window to redesign for the future and

to take advantage of the major shift in transport protocol requirements. This major shift, specifically in data rates, would thus give rise to new designs of transport protocols and the opportunity to design a standard that took advantage of previous work in the field.

2. SNR

The SNR protocol is one such emerging protocol that takes advantage of past protocol designs. The strength of this protocol is found in its ability to accelerate the exchange of state information between communicating entities. This process, independent of significant events that may have occurred on the net, is performed frequently and at a periodic rate. The result is a less complex protocol that still performs the necessary functions described earlier for the transport layer. Additionally, the protocol uses a modified selective repeat retransmission scheme for error recovery. [MCAR92]

3. XTP

The third emerging protocol, XTP is a new protocol that does not adhere strictly to the OSI reference model. This partially due to XTP's combination of the transport and network OSI layers. XTP gets its strength as a protocol from this combined design's ability to handle modern, fast fiber optic networks. This layer combination enables XTP to tightly control end-to-end system parameters such as rate control. Combination of layers three and four into a transfer protocol also facilitates the eventual siliconization of the protocol providing even faster host protocol processing [STRA92b]. Furthermore, XTP is programmable in the sense that data exchange policy is user selected, while the mechanisms to transfer data are provided by the protocol. XTP also preserves many individual features of other protocols where these features were found to be useful, a good example is the credit based window concept used in other protocols.

E. PROBLEMS FACING EMERGING TRANSPORT PROTOCOLS

Emerging protocols, whether new or updated versions of existing protocols, are faced with performance issues that stem from the need for efficient and reliable end-to-end data

service that guarantees high-speed throughput and lower round-trip latency that meets user requirements.

To draw an analogy seen in vehicle traffic control management, think of the high-speed data medium as an advanced modern super highway that allows vehicles (data packets) to travel at very fast speeds. Using the equation below, a change from an ethernet standard at 10Mbps to a FDDI standard of 100Mbps is analogous a highway where speeds are increased to 550mph. If FDDI rates are increased even more to 600Mbps, which is currently technologically feasible, speeds on the highways could increase up to 3300mph. Working with the same analogy, but applying it to bandwidth rather data rates would yield highways that are 80 lanes wide at the FDDI bandwidth of 100Mbps.

$$\frac{100\text{Mbps}}{10\text{Mbps}} = \frac{550\text{mph}}{55\text{mph}}$$

$$\frac{100\text{Mbps}}{10\text{Mbps}} = \frac{80\text{lanes}}{8\text{lanes}}$$

Figure 4: Data/Highway Comparison

Now comes the problem, traffic laws, traffic rules, traffic lights and similar constraints inhibit the flow of any single high-speed vehicle in order to allow more vehicles access to the highway. A good example is the metering lights that exist on highways in Los Angeles. These lights, laws and rules are *forms of protocols*.

The metering lights mentioned limit the number of cars on the highway. If the lights were timed with excessively long reds and short greens, fewer cars would be on the highway, but they could go very fast. This would be a simple traffic solution, but would upset many commuters. Similarly, a very simple transport protocol could allow limited data onto a network that would move very fast. Actual transport protocols often slow the overall delivery of packets by as much as 50%. A 100Mbps data medium, at a 50% transport layer performance would yield only 50 Mbps. This is not to say that the transport protocol is solely responsible slowing down data transmission, only one factor.

To take a look at one solution to the highway traffic problem, a complex protocol that estimated total number of vehicles on the highway based on past history, entry and exit in road sensors, time of day and other factors could be used to estimate traffic loads in those regions. If a region had excessive traffic, the length of the red light would increase, if not longer greens. In the data protocol a more complex protocol can be used to attempt optimal use of the data highway. The similar problem with the transport protocol trying to get as much data onto the network as possible while avoiding congestion has led to the development of complex protocols that monitor and react to data rates, data flow and errors, on regions of "highway" that can extend through multiple sub-regions of a given network.

Other examples can be extrapolated to the highway traffic analogy for transport protocol high-speed network problems. Logically, the analogy does not always strictly hold, but does provide a familiar frame of reference for the reader to understand protocol issues. Other problems that are discussed in [MCAR92] include timers and round-trip delay estimations (how long does it take to drive to City X, deliver a product and return), error recovery (what if there is an accident on the highway? do I send another truck?), flow control tied to error detection and recovery (with an increased number of accidents, reduce entry onto highway or use alternate routes) and non-standard formats (oversized vehicles). Looking at the data system as a highway may enable the development of new highway rules that take advantage of all "80 lanes."

An additional problem that will face high-speed protocols and particularly those using fiber, deals with the natural buffering properties of the medium. This natural property of any transmission media, is exaggerated by fiber's ability to act like a large volatile memory. This is not necessarily an advantage. With SONET large amounts of data tend to be on the transmission media at any given time, which must be accounted for in any protocol.

One benefit of the large amounts of data at high-speeds will allow the network to perform as a virtual backplane for distributed computing applications. This advantage however brings with it its own set of problems, particularly those associated with the guarantees needed for real-time messaging synchronization and transaction processing.

Another problem that faces high-speed networks is a reversion to the stop and wait protocols of the past. Since so much data is travelling so fast, acknowledgments may not return fast enough for the continuing flow of data to occur. A flow control system that recognizes the large amounts of data in transmission at one time is needed.

In this chapter the basics of high-speed data transfer were presented, it is now time to take an in-depth look at one particular protocol, XTP.

III. THE XPRESS TRANSFER PROTOCOL

In this chapter a short overview of the Xpress Transfer Protocol, XTP is presented. This explanation will serve two purposes. First, the reader will gain an understanding of why XTP was developed, what protocols XTP was based on and how it solves the high-speed data transport problems presented at the end of the last chapter. Second, in the later half of the chapter, the specifics of the XTP structure and operation are presented. This knowledge, when combined later with the formal specification of XTP will give the reader an understanding of how the XTP protocol policy and mechanisms work.

A. BACKGROUND

XTP, Express Transfer Protocol, is a high-speed data transfer layer protocol. The XTP transfer layer incorporates functionality from the network (layer 3) and transport (layer 4) layers of the OSI protocol reference model. Additionally, XTP utilizes selected and specific functionality of existing protocols to optimize its own performance. The XTP design effort began in 1987 under the leadership of Greg Chesson. The goal of the original group was to develop a communications protocol that would meet the needs of modern, distributed computing systems and real time control systems. Special attention was placed on the separation of protocol mechanisms from protocol policy allowing the implementor to select the particular mechanisms that support an end user's desired protocol paradigm. The fundamental XTP assumption is that the differences among data service models is directly related to the quantity, reliability, frequency and destination of data transmissions. Thus to implement a particular service model, mechanisms, not policy are chosen [STRA92b]. In this thesis, the connection-oriented paradigm mechanisms (normally seen in the TCP, TP 4 and NETBLT protocols) will be modeled to show end-to-end reliable packet delivery.

XTP is also developed with the idea of eventual implementation in hardware. This would place the first four layers of OSI reference in silicon. Processing time at source, destination and intermediate nodes could then approach that of the network medium, thus

reducing end-to-end latency and increasing overall throughput while decreasing the need for extensive buffering of incoming packets as they wait for processing.[WEAV92b]

The motivation for XTP has grown out of the need to develop a new protocol that will provide distributed applications across a large range of networks that include LANs, MANs and WANs. The XTP communications architecture is designed for these future networks and will scale gracefully from megabits to gigabits while providing interoperability through the open systems concept.

B. BASIC CONCEPTS

There are a couple key concepts in XTP that form the basis of all XTP communications. These concepts are similar to concepts of different names in others protocols and include context, association, path, data stream and packet. The **XTP context** represents state information at one endpoint or host. The context represents an instance of an active XTP communication. The term **association** is given to a pair (or more) of active contexts and the data streams between them. The association is symmetric once it is established, however during establishment it is asymmetric and does not require a traditional handshake process. An association is different from the familiar "connection" description of classical protocols. Although similar in many regards, the XTP association can take on varied paradigm properties, some of which are not connection oriented at all, such as the Versatile Message Transaction Protocol (VMTP) acknowledged datagram paradigm. Therefore, the term association was necessary to emphasize this protocol feature. The third concept is that of a **data stream**. A data stream is a sequence of bytes of arbitrary length that make up packets and messages. In XTP there is sequence number space up to 2^{32} . The sequence numbers provide the basis for flow and error control of an association. Finally, there is an **XTP packet**, that is contained or encapsulated in the MAC layer of the supporting lower layer. Packets can be one of nine types discussed later. With these basic concepts the XTP protocol was built with influence from existing protocols as well new features designed for high-speed systems, both of which are discussed next. An

architectural diagram below shows XTP from a single host perspective [STRA92b]. It is important to note that the diagram shows only one half of an XTP association and that the context records and control block structures may support multiple associations originated and/or terminated within the same host.

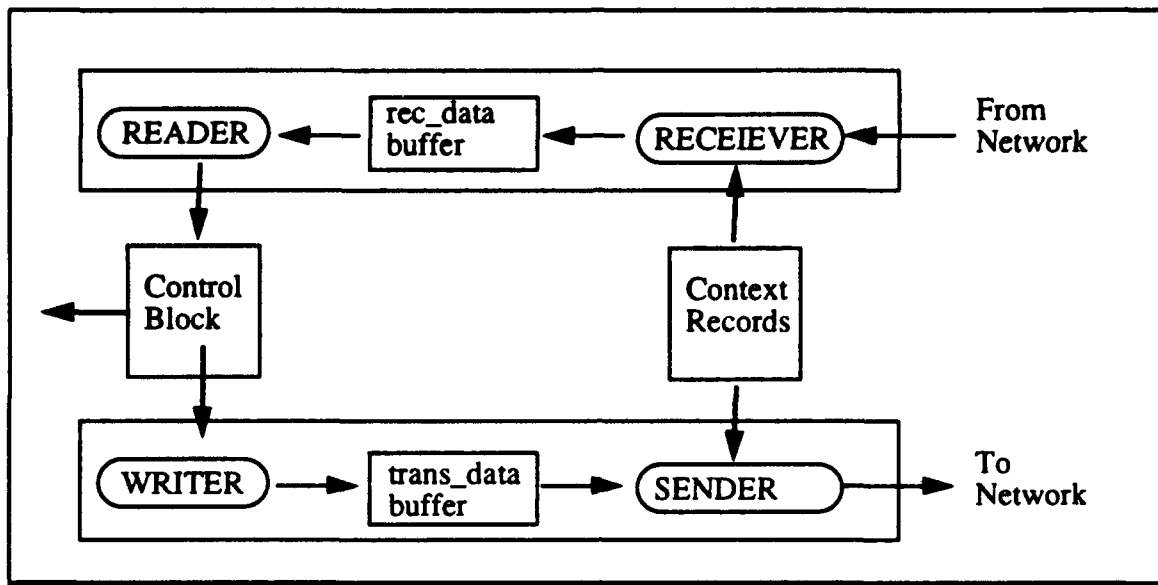


Figure 5: XTP Host Architecture

The above diagram shows the basic architecture for the XTP protocol. This architecture diagram only begins to hint at the flexibility of XTP. The protocol flexibility of XTP is a direct result of many existing transport protocols as discussed in the next section on influential protocols. XTP retains many of the features of these protocols, which may be incompatible if not applied properly. Therefore it is necessary for the XTP implementor to apply a specific requirement to the protocol when designing to the user's needs. If mixing of incompatible XTP options occur, an undefined state may result. (A good example of the problem is seen when the NOFLOW option is toggled to set during the lifetime of an active. The effect on the data stream in this case is to disable the active acknowledgment processes, losing transmission accountability.)

Before looking at influential protocols a look at influential models is appropriate. In Figure 6 below XTP is shown in a manner similar to the OSI resource model framework.

The exact requirements of the OSI reference model are not met due to XTP's combination of the network and the transport layers, however basic concepts of the model are met.

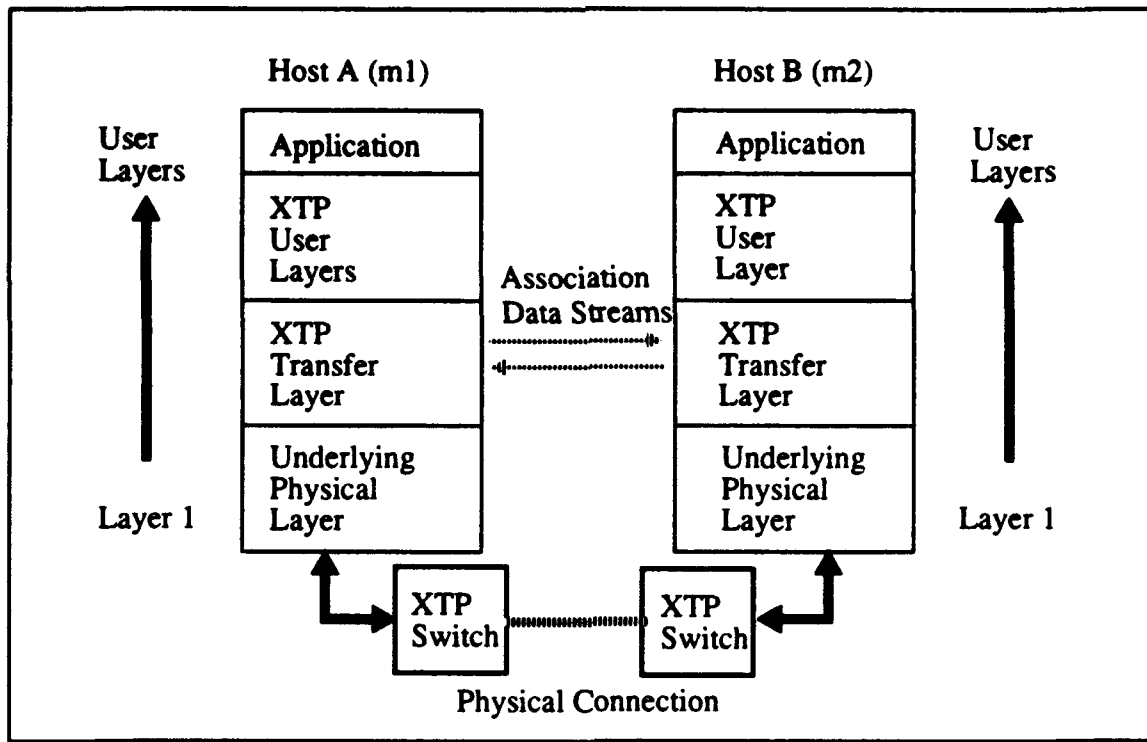


Figure 6: XTP Peer to Peer communication Model

C. INFLUENTIAL PROTOCOLS

As mentioned XTP developers studied extant protocols and incorporated many of the existing protocol features into XTP. The central thought of this effort was to capitalize on the lessons learned from the experience of other protocol designers and implementors. The most influential protocols on XTP are TCP, TP4, Delta-T, Network Block Transfer, GAM-T-103, VMTP, and Datakit. A short summary of which features from which extant protocol follows, this list is not complete but is provided to give the reader an idea of the in-depth protocol investigation that was conducted. For a more complete description see [STRA92b].

1. TCP

The Transmission Control Protocol (TCP) is the first of two widely known standards for reliable transport layer protocols. Although XTP uses a different architectural framework (i.e. a transfer layer) and supports a wider selection of service models, it does use key operational features of TCP. The first feature is the sliding window credit based end-to-end flow control technique. This TCP technique uses control packets to indicate loss due to overflowing buffers at the receiver. XTP uses this basic concept and adds an extended sequence number space. Another example of where XTP followed TCP was in the use byte based sequence numbers for the purpose of tracking transmission errors. In XTP this field is expanded to 32 bits descriptors to handle greater quantities of sequence numbers.

2. TP4

The second widely known protocol that has influenced XTP is Transmission Protocol Mode 4 (TP4), based on the Open Systems Interconnect architecture described earlier. There are similarities between TP4 and TCP, such as the use of checksums and length of field indicators, XTP has used the mechanisms as well. One other significant use of a TP4 concept was in the XTP design for individual packet priorities. In TP4 there are two levels of priority, this concept was expanded under XTP to a 32 bit SORT field.

3. Delta-T

Delta-T is a high performance protocol developed in the late 1970s, to meet the needs of an integrated network within a distributed open architecture. The important Delta-T contribution is the protocols ability to show how timer mechanisms may be used for safe connection management and, through the use of the timer based mechanisms, develop a hazard free connection oriented protocol. XTP's use of connection and path identifiers were designed with Delta-Ts requirements for hazard free communications in mind.

4. Network Block Transfer

Network Block transfer or NETBLT, is specifically designed to handle large blocks of data. XTP designers viewed this as an important service necessary for a transport protocol that would use mediums with greater bandwidths over longer distances. XTP's formulation of a rate control policy was a direct fallout from an investigation of NETBLT's two parameter transmission rate and burst size rate control implementation. The NETBLT ability to send large blocks of data can be seen in XTP's de-coupling of flow and error control by separating the request for reception status from the operation of the window based flow control mechanism.[CLAR89]

5. GAM-T-103

The GAM-T-103 specification, published by the French Ministry of Defence, specifies a Military Real Time Local Area Network based on a hierarchical network similar to the OSI model. XTP investigated this protocol and noted GAM-T-103's emphasis on the need for low latency messages, point to multi-point traffic, multi-point to point traffic and synchronization. The features of the GAM-T-103 service profile were included in XTP.

6. VMTP

The Versatile Message Transaction Protocol (VMTP), was created from an observation that future distributed systems would require the reliability and other features from connection oriented protocols, while at the same time maintaining a desire for short transaction message transfer [CHER88]. XTP designers recognized this need for transaction or client/server type distributed communication. The XTP mechanisms therefore support the message transaction model used in VMTP while maintaining the exact same mechanisms that XTP has for long-lived data transfers.

7. Datakit Architecture

The last protocol reviewed is the Universal Receiver Protocol that was spawned from the Datakit architecture. The Datakit architecture designed at Bell Labs grew from the

need for ubiquitous interconnection for tele-commutations devices. The demands on the UDP receiver were very stringent, and were recognized as a good measure to challenge XTP. Particularly the need to process a received packet within its arrival time.

D. NEW FUNCTIONALITY IN XTP

In addition to drawing on the successes and failures of extant protocols, XTP also incorporates new functionality into its basic design that will meet the requirements of high-speed data networks.[WEAV92b]

1. Multi-cast

Multi-cast is a very important feature for advanced distributed network applications. This XTP feature will allow a user to define a transport layer multi-cast group that allows a single user to communicate with multiple users. This one-to-many architecture includes the capability for receivers to send negative acknowledgments while the group maintains its progressive nature. That is to say the group does not wait for a single receiver to catch up, but simply continues transmission. The receiver that is behind however can request retransmission of any lost data. This will present unique problems for the sliding window protocol.

2. Priorities

The second function that XTP amplifies is a priority scheme. Packet priority has been an issue for transport protocols for a number of years. In XTP a 32 bit SORT mechanism is provided that allows the user to discriminate incoming packets at either endpoints or intermediate nodes. This expanded priority system is extremely useful in real time systems and decreases the likelihood of priority inversion.

3. Rate and Burst Control

In addition to checksums and buffer management techniques used by most protocols XTP adds rate and burst control features. These features together allow the receiver to actively manage the incoming data and prevent buffer overruns at intermediate

nodes as well as at the receiver. The rate specifies a maximum bytes per second threshold whereas the burst control throttles the maximum bytes per burst.

4. Address Translation

XTP implements a hardware hashing function that provides translation of existing addressing types, represented in XTPs address segment and address type of the FIRST packet, to an XTP KEY. Once associated with the 32 bit KEY addressing for subsequent transmission to the same destination on the same as XTP association becomes simpler through the use of a lookup table.

5. Retransmission

A significant feature of XTP is its ability to efficiently execute retransmission of detected errors. The retransmission scheme is implemented through the use of a spans field that identifies correctly received blocks of data after the point at which all data has been received correctly. This mechanism however, is not required and allows a user to revert to a Go-Back-N method of error recovery if desired.

6. Acknowledgment Control

In keeping with the design philosophy of XTP, a mechanism is provided that allows the XTP implementor, from the transmitting host, the ability to control acknowledgments. This new feature further separates mechanism from policy. It however increases the complexity of the protocol since the implementor must now make a choice as to whether or not an acknowledgment is desired. The advantage of selectable acknowledgment, outweighs the disadvantage of increased complexity.

7. Out of Band Data

XTP as a new protocol, was able to review features that were desired by current network users. Once such feature that has great potential, and provides the protocol user added flexibility is the use of out of band data. Out of band data is used to handle the transfer of data without actually embedding it in the transport layer data stream. One

possible use of this feature is the exchange of application layer semantic information about the layer itself. Another use is to time-stamp time-critical data.

8. Data Pipeline Size

One of the most important advanced features of XTP will be its ability to handle large amounts of outstanding data on the physical medium. XTP implements a standard sliding window protocol with a 32 bit sequence number. This allows sequencing of up to 4 giga-bytes of data. Through the use of the synchronization field the sequence number space can be enlarged to 64 bits which will handle terabit/second networks. The expansion of the sequence numbering scheme prevents the protocol from reverting to a simple stop and wait operation. The extended window size is critical in preventing a serious reliability error that can occur if sequence numbers are reused. The problem of sequence number wraparound is that in high-speed networks the available sequence number space can be depleted very rapidly. This error is further explained in [JACO92]. Thus, XTP prevents this problem through the 32 bit sequence number and synchronization extension.

9. Connection Services

One of the most common protocol constructs is the handshaking mechanism. Handshaking is often performed for reasons of reliability on connection oriented communication services. With implicit connection set-up XTP performs handshaking in two packet exchange. This allows shorter overall connect times, especially on single transaction type packets.

10. Alignment

The final added functionality available with the XTP is an alignment operation that uses an offset from the beginning of the data segment and length field to identify where in a packet the data is. This prevents the copying or realignment of data within a packet.

E. XTP STRUCTURE

An abbreviated description of XTP packet structure is provided so the reader may gain a basic understanding of the protocol.

1. Basic Packet Structure

The basic XTP packet structure contains a header, a middle segment and a trailer. The middle segment can be either an information segment or a control segment. The Header and Trailer Fields are consistent between packets, however the middle segment may vary amongst five basic field structures (one for the control segment and four for the information segment) and nine packet types. Figure 7 below shows the basic packet structure.

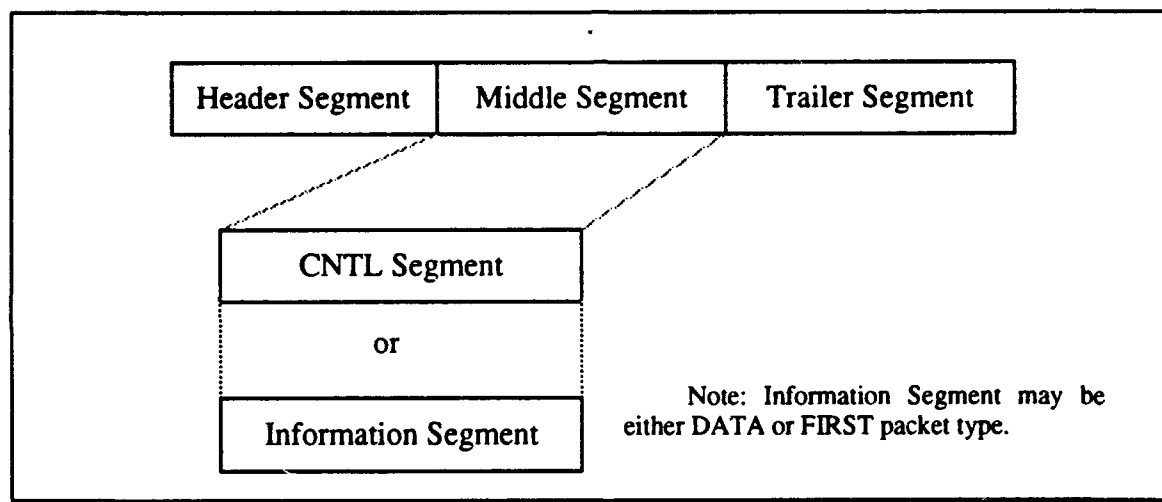


Figure 7: XTP Basic Packet Structure

The nine packet types that use the basic structure are the FIRST, DATA, Control (CNTL), PATH, Diagnostic (DIAG), Maintenance (MAINT), Management (MGMT), ROUTE and Return CNTL (RCNTL). A closer look at FIRST, DATA and CNTL will suffice.

2. Header and Trailer Segments

The header and trailer formats do not change, therefore the bit positions for any given header field are always in the same position. This greatly simplifies host passing.

Within the header there is embedded packet information that may change based on the type of packet being sent or the communications paradigm being implemented. For a more detailed description of the header and trailer packet structure see Figure 8. The trailer consists of only one field, that is used for the data checksum.

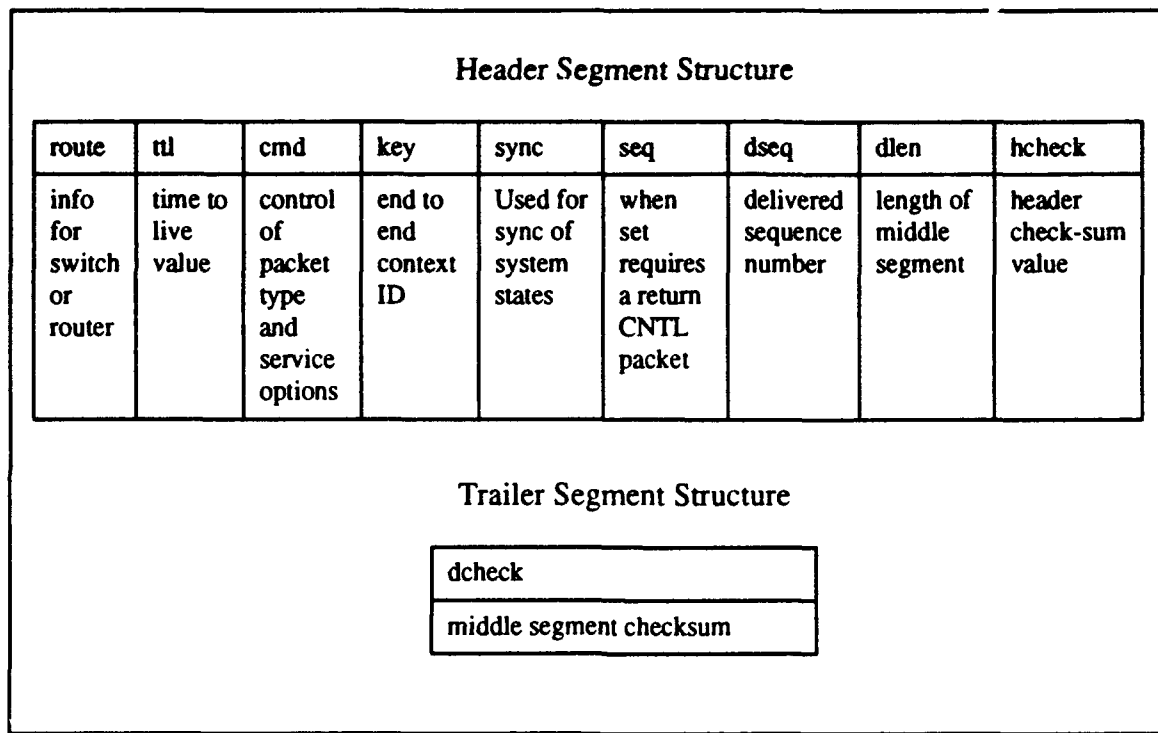


Figure 8: Header and Trailer Packet Structures

- The key value is similar to the connection and transaction ID of other protocols.
- Note: The SORT field has been omitted in this paper.

3. FIRST Packet Structure

The FIRST packet, which is used only once, is the first packet to be sent over an association. The XTP association is likened to a connection described in other protocols.

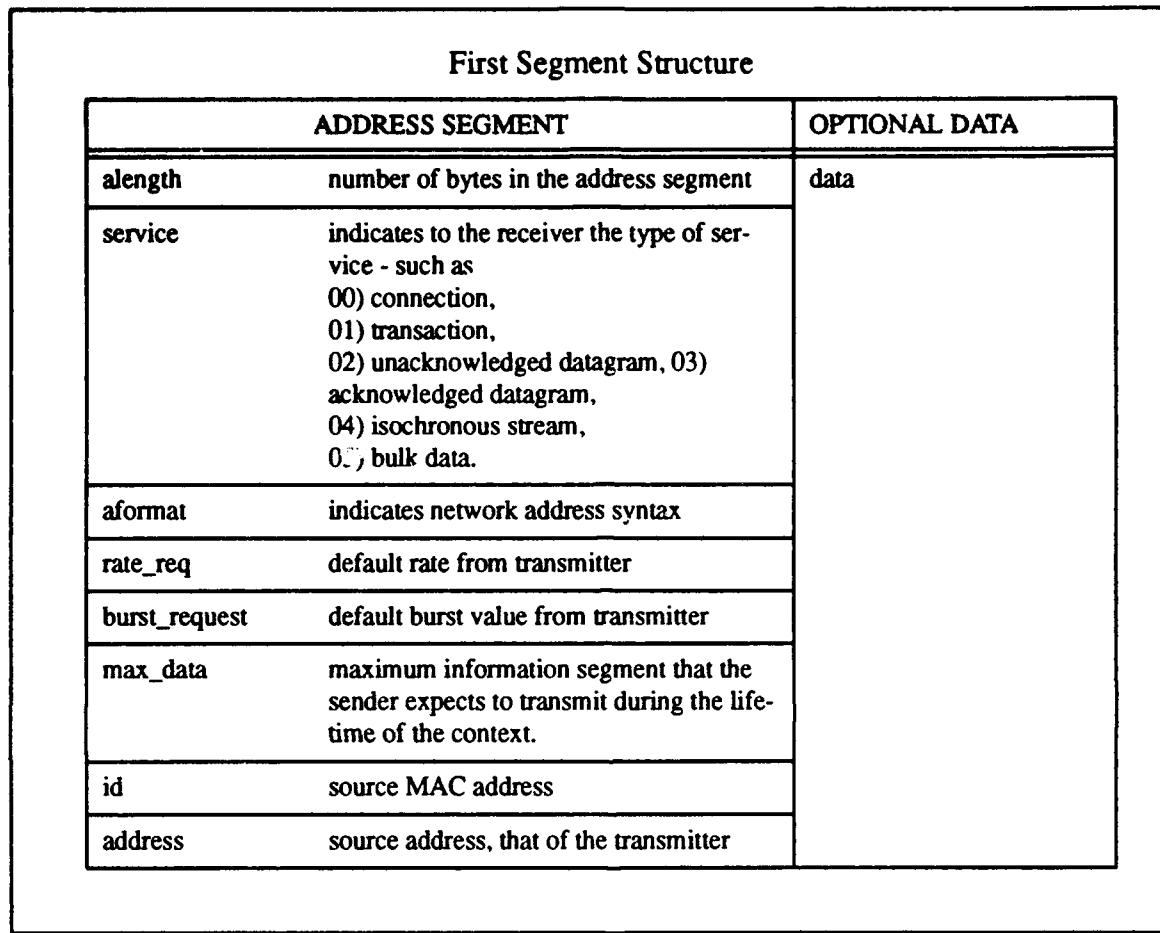


Figure 9: Middle Segment of a FIRST packet

XTP employs mechanisms to ensure that only one FIRST packet per association is allowed. This eliminates the possibility of receiving two FIRST packets for the same context. The association creation process is asymmetric and therefore a FIRST packet need only be sent in one direction, the initiating direction.

4. DATA Packet Structure

The data segment, a type of information segment, always exists in DATA packets and sometimes exist in FIRST packets.

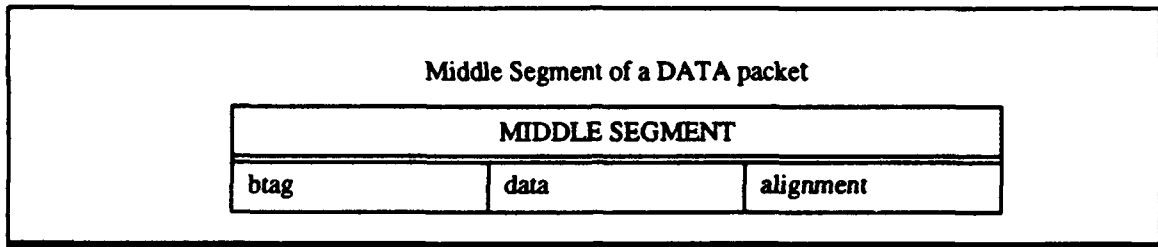


Figure 10: Middle Segment of a DATA packet

Data segments are a constant feature of XTP. The flexibility of controlling mechanisms around the data packet allows XTP to implement varied data transfer paradigms. Classic connection oriented, transaction and datagram paradigms are all supported and may use the basic DATA packet for the transfer of data.

5. Control (CNTL) Packet Structure

The control packet is the XTP mechanism that exchanges state information between the sender and the receiver and thus represents the state of the context. It is sent

upon the receipt of a SEQ or DSEQ indicator in the header of the transmitter. The CNTL

Detailed Control Packet Description	
MIDDLE SEGMENT	
rate	sets maximum rate at which data can be processed from a transmitter. Used to throttle the transmitter.
burst	maximum number of bytes that a receiving node can handle in a single packet burst.
echo	used for synchronization of state machines, returns the value that was received in the sync field of the header
time	current value of time at the sending host and...
techo	with the time field is used to calculate the round-trip timing delay.
xkey	abbreviated context lookup value for address matching of packets to their appropriate contexts
xroute	abbreviated route lookup value for matching packets to their appropriate route on a per hop basis
alloc	used for flow control, a do not exceed sequence number., represent the internal buffer space of the receiving XTP machine.
rseq	indicates sequence number expected on next incoming packet
nspan	number of sequence number pairs
span	ranges of intact sequence numbers of received packets. Bordered by missing sequence numbers.

Figure 11: Middle Segment of a Control (CNTL) Packet

message contains many XTP mechanisms. In addition to the acknowledgment mechanism, CNTL messages also contains state update mechanisms (echo), flow control (rseq), error control (nspan, span) and rate control (rate) information.

6. Other Packet Structures

There are nine total packet structures in XTP, three have been presented. The remaining packet formats and a short description follows. PATH: address information for

re-threading a path or joining a multi-cast association; DIAG: diagnostic information for fault notification; MAINT: network maintenance information; MGMT: network management information; ROUTE: path release information; RCNTL: control information between two switches. These packet structures are used in the information segments of the packet by the same name and contain transport layer messages.

7. Timers

There are five distinct timers used in XTP structures, they are primarily designed to prevent deadlock in a number of situations when an expected event fails to occur. The timers are the CTIMER, CTIMEOUT, WTIMER, RTIMER and the round trip time (rtt). There is also a counter called `retry_count` used with the timers. The full set of XTP timers and counters provides for a robust environment for defining association set up and termination techniques. A brief explanation of each timer is now presented with a description of its use.

CTIMER - The connection timer is a long duration countdown timer that, when expired, represents inactivity of an association. It prevents against waiting indefinitely for an association to progress that has reached an undefined state.

CTIMEOUT - The connection time-out timer is a short duration timer that bounds the time on an attempted message exchanges and inactive association revival. CTIMEOUT is thus used in the state synchronization process and bounds the time for attempted handshakes. The association is considered alive even if CTIMEOUT expires on an attempted message exchange, that is until the `retry_count` is incremented to predetermined maximum value. The second purpose for CTIMEOUT is to prevent the reactivation of a terminated association due to a retransmitted or late arriving FIRST packet.

WTIMER - The wait timer is a countdown timer that is maintained for each XTP context. It represents the maximum amount of time that a context will wait for a CNTL packet once a SREQ status request has been sent.

RTIMER - The refresh timer is a countdown timer that is used in connection with rate control procedures. It defines the period in which a data burst can arrive. When it expires a new allocation of buffer space is permitted

rtt - round trip time. The XTP round trip timer is a context based variable that maintains the elapsed time for a time signal to traverse the network from the sending host to the receiving host and back. Initially, the implementation sets a default rtt value.

retry_count - an implementation parameter is used to count the number of times XTP will attempt message exchange. When **retry_count** count expires the context aborts.

F. BASIC PROTOCOL PROCEDURES

In this section a brief introduction to the basic procedures of the XTP protocol are presented. A more complete description of the data transfer operation is completed in the specification chapter of this thesis.

1. Association Management Procedures

Association management is composed of three primary functions, establishing an association, maintaining an association and terminating an association.

a. Establishment

Association establishment, conceptually the same as a connection set up, consists of just a few basic steps. The sender and the receiver are both initially in the quiescent state. First, the XTP user initiating the association, issues an OUTPUT command to the XTP machine. Receipt of the OUTPUT command generates a FIRST packet to be sent to the intended receiver and puts the sender in the active state. The receiver, at this point, assumed to be in the listening state, accepts the FIRST packet and enters the active state. At the time of establishment a unique association key (or address) value is created that is used identify outgoing packets and to associate incoming packets with the correct context. A route value is similarly assigned to uniquely identify the host exit port and identify the path for the association. This is a traditional network level function.

When both the sender and the receiver enter the active state the association is ready for use. There is no requirement for an acknowledgment of association establishment to be sent to the receiver. However, as seen later in particular XTP paradigms, an acknowledgment may be used.

b. Maintenance

XTP association maintenance procedures include full context lookup, return key and key exchange procedures. A full context lookup is a means by which an association is uniquely identified in a translation table. The table provides a mechanism by which subsequent packets belonging to the same context can efficiently be sent eliminating the need to include the complete address in follow-on packets. The second maintenance procedure, return key, provides for a mechanism in a returning packet that allows for fast and efficient context matching. This eliminates the need for a full context look-up. The last maintenance procedure, key exchange, is a mechanism that allows the same abbreviated lookup process to occur in the forward direction.

c. Termination

The final association management procedure is termination of the association. Termination can be accomplished in four ways: independent graceful close; abbreviated graceful close; forced closed and abort. The degree of data delivery assurance varies from guaranteed with a graceful close to no guarantee with an abort. This is another good example of the separation between mechanism and policy. The graceful close procedure is one example where XTP does not strictly follow the OSI model since graceful close is prescribed as a session layer function under OSI.

2. Path Maintenance Procedures

Path maintenance procedures are responsible for establishing, maintaining and releasing the structures within the endpoints and the intermediate nodes which represent the XTP path. Path establishment occurs concurrently with that of the association and the path

is maintained with the route value in a similar manner that the key value maintains the context. The XTP route procedures perform network level functions.

3. Data Transfer Procedures

End-to-end data transfer begins at the XTP user buffer with the issuance of the OUTPUT command and proceeds to the XTP data buffer, followed by data transmission over a network channel, reception at a receiving XTP data buffer and final delivery to an XTP user buffer. After the issuance of the OUTPUT command each data byte is assigned a sequence number and placed in the XTP buffers. The data is then copied into the data fields of a DATA packet. The data packet carries with it the starting sequence number of the information (seq) and the number of bytes of data (dlen). When this data is delivered to the receiving XTP data buffer it is identified by sequence number, and then delivered to the XTP user buffer at the destination in contiguous sections. XTP uses an end of message indicator in the packet options field to indicate the current packet is the last of a given message.

4. Flow Control Procedures

XTP flow control procedures are based on an end-to-end credit based sliding window flow control algorithm. The windows sliding rate across the transmitters available sequence numbers is controlled by the receivers ability to accept and acknowledge new data. Flow control is operated through parameter passing between sending and receiving machines. Flow control communication is handled, for data bearing packets, by the starting sequence number (seq) and an indicator of the last sequence number successfully delivered to the XTP user buffer (dseq). In CNTL packets the seq, dseq and the window value (alloc) are used for flow control. The seq value of a CNTL packet has a different meaning than in an information packet. In the CNTL packet seq indicates the next byte that is expected on the forward data stream. The dseq, as before indicates the first byte that is unacknowledged. The alloc number, is the highest sequence number that the receiver will accept. This makes alloc the upper end of the window and dseq the lower edge of the window. In summary, the

flow control is constrained by the local policy on buffer space through the alloc field of the CNTL packet. However, flow control can be completely suspended by the setting of the NOFLOW bit in all outgoing packets. In this situation, any alloc value returned by the receiver is ignored.

5. Rate Control Procedures

Rate control is different from flow control and is an attribute of the XTP path and each segment of the XTP path. Rate control limits communications between nodes along an XTP path. This is accomplished through the rate field and the burst field of the CNTL packet. The rate value dictates a recommended maximum bytes/second rate for a given pair of XTP nodes. The burst value dictates a maximum number of bytes that can be consumed in a given burst of packets. These values can be used to control the bandwidth used for a XTP path that may have more than one XTP association.

6. Error Control Procedures

The XTP procedure for error control is responsible for creating a reliable end-to-end data transfer service. Error control checks for and detects lost, corrupted and duplicate data. Once data errors are discovered mechanism for retransmission and data accountability are employed. Retransmission is accomplished through a selective repeat mechanism that identifies successfully received spans of data between the lower edge of the window (dseq) and the upper edge of the window (alloc). The gaps between spans are then retransmitted to the receiver. Data below the dseq field can be released by the transmitter buffers. The second value that indicates acknowledgment is the rseq value in the CNTL packet. The rseq value indicates the data that has been received by the destination endpoint on a particular data stream. This value confirms receipt of XTP data buffer space data, but not necessarily XTP user data buffer space that is acknowledged by dseq. The error control procedure can be simplified to a Go-Back-N protocol if the implementor chooses.

IV. FORMAL SPECIFICATION OF XTP

In this chapter a limited formal specification of the XTP protocol is presented. The model used for this specification, SCM, is presented first, followed by a description of XTP protocol structures used in the specification and then the formal specification.

A. SYSTEM OF COMMUNICATING MACHINES (SCM)

In this section the model used to specify and analyze the protocol is defined. A more detailed description appears in [LUND91b] and [LUND91b].

A system of communicating machines is an ordered pair $C = (M, V)$, where

$$M = \{m_1, m_2, \dots, m_n\}$$

is a finite set of machines, and

$$V = \{v_1, v_2, \dots, v_n\}$$

is a finite set of shared variables, with two designated subsets R_i and W_i specified for each machine m_i . The subset R_i of V is called the set of read access variables for machine m_i .

Each machine $m_i \in M$ is defined by a tuple $(S_i, s, L_i, N_i, \tau_i)$, where

(1) S_i is a finite set of states;

(2) $s \in S_i$ is a designated state called the *initial state* of m_i

(3) L_i is a finite set of *local variables*

(4) N_i is a finite set of names, each of which is associated with a unique pair (p, α) , where p is a predicate on the variables of $L_i \cup R_i$, and α is an action on the variables of $L_i \cup R_i \cup W_i$. Specifically, an action is a partial function

$$\alpha: L_i \times R_i \rightarrow L_i \times W_i$$

from the values contained in the local variables and read access variables to the values of the local variables and write access variables.

(5) $\tau_i: S_i \times N_i \rightarrow S_i$ is a transition function, which is a partial function from the states and names of m_i to the states of m_i .

Machines model the entities, which in a protocol system are processes and channels. The shared variables are the means of communication between the machines. Intuitively, R_i and W_i are the subsets to V to which m_i has the read and write access, respectively. A machine is allowed to make a transition from one state to another when the predicate associated with the name for that transition is true. Upon taking the transition, the action associated with that name is executed. The action changes the values of local and/or shared variables, thus allowing other predicates to become true.

Let $\tau_i(s_i, n) = s_2$ be a transition which is defined on machine m_i . Transition τ is enabled if the enabling predicate p , associated with name n , is true. Transition τ may be executed whenever m_i is in state s_i and the predicate p is true(enabled). The execution of τ is an atomic action, in which both the state change and the action a associated with n occur simultaneously.

The sets of local and shared variables specify a name and a range for each. In most cases, the range will be a finite or countable set of values. For proper operation, the initial values of some or all of the variables should be specified. A conceptual diagram of the SCM model is shown in Figure 12.,

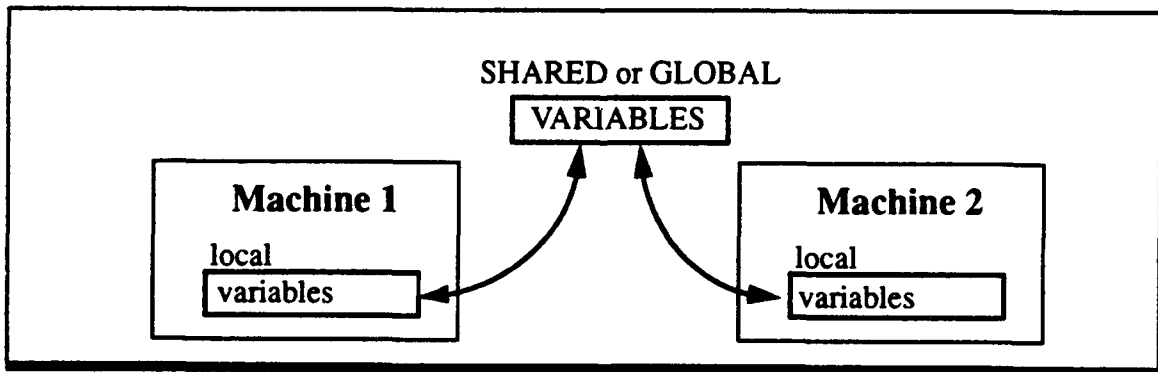


Figure 12: SCM, two machine behavior representation.

B. SPECIFICATION STRUCTURES

Before beginning the specification it is important to identify a set of XTP structures and variables that will aid in the protocol specification. These structures and variables are used in communications, messages and XTP procedures that are involved with the SCM predicate action tables. By convention shared variables, buffers and channels are capitalized, local variables are in lower case and functions or procedure name will have the First Letter Capitalized. Commands to or from the XTP user machine are capitalized. Host A, the transmitter is machine 1, (m1) and Host B is the receiver (m2).

1. Communication Structures

The communication structures in the SCM model are implemented as FIFO queues. The FIFO queues act as communication channels and represent the shared variables of the SCM. Below in Figure 13 we see an XTP association with its global buffers F_CHAN and R_CHAN and local buffers trans_data_buf and rec_data_buf.

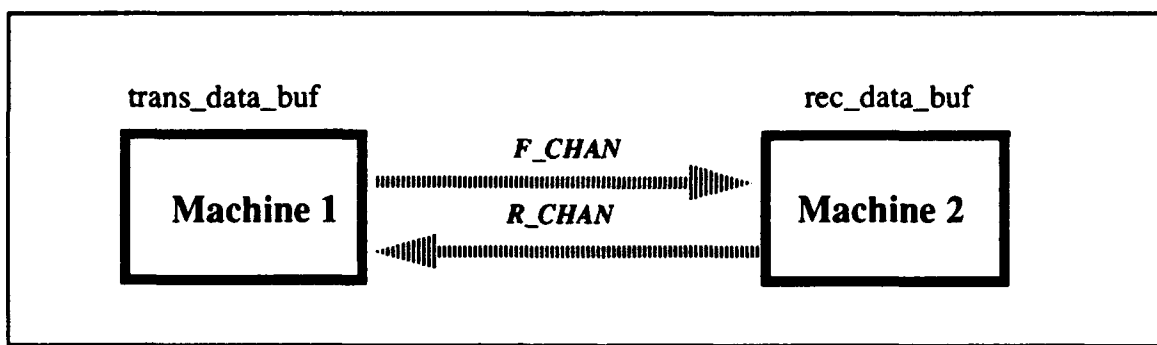


Figure 13: XTP Association

a. Global Variables and Buffers

XTP buffers serve as the logical means for the transportation of data. Virtual buffer size for XTP must be very large to handle the total sequence number space, up to $2^{32} - 1$ are allowed in the protocol. Even though sequence numbers are assigned to individual bytes of information it is rare that a single information byte is sent. Groups of bytes (maximally limited by the physical frame size) are sent from the sender to the receiver. These byte groups are processed from the transmitter data buffer to the receiver buffer

through the forward channel of the data stream. For purposes of clarity local variable record data structures are introduced to condense information representation and data passed on the global channels. Record structures are built from the packet descriptions in section E, XTP STRUCTURE, in Chapter III .

F_CHAN. The F_CHAN is the forward channel of the XTP association. It is modeled as a FIFO queue of size equal to that of the allowable sequence number space. The channel supports all nine packet types and constitutes the forward data stream.

R_CHAN. The R_CHAN is the return channel of the XTP association. It is also modeled as a FIFO queue of size equal to that of the allowable sequence number space. The channel supports all nine packet types.

b. Transmitter Local Variables and Context Variables

The following variables are identified as critical transmitter local variables for the basic XTP procedures specified later in this chapter and are maintained at all times.

(1) *trans_data_buf* - The transmission data buffer represents the XTP working buffer at the sending side of the association. This buffer is used for packet assembly and transmission. Sequence numbers are assigned to the data in the *trans_data_buf* and identified by *seq* the starting sequence number and *dlen*, the length or quantity of sequence numbers. The data buffer represents a FIFO queue where the input is a data stream from a user implementation and the output is the same data, now ready for inclusion in a data packet.

(2) *status_variables* - status variables are local variables that maintain state information or other characteristics of an XTP association or context. They may be implementation dependent. Values of status variables are never read directly into an outgoing record. That is to say status variables are not used for purposes of visibility from one machine to another. Status variables may be input to the enabling predicates of predicate action tables, since the table are maintained locally. In the SCM modeling of XTP

it is attempted to maintain terminology and some implementation detail used in [STRA92b].

TABLE 1: LOCAL STATUS VARIABLES - TRANSMITTER

Flow Control	Rate Control	Timers	Other
<i>t_alloc</i>	<i>credit</i>	WTIMER	<i>t_context_state</i>
<i>bseq</i>		CTIMER	<i>sync_echo</i>
<i>eseq</i>		CTIMEOUT	<i>saved_sync</i>
		RTIMER	<i>output</i>
		<i>rtt</i>	<i>frame_size</i>
		<i>retry_count</i>	

(3) Explanation of Status Variables

t_alloc maintains a value at the transmitter of the receivers allocation (or window size) available.

bseq is the transmitters record of the beginning sequence number of sent, but unacknowledged data.

eseq is the value equal to one greater than the ending sequence number sent but not acknowledged.

credit is an orthogonal rate control value that limits transmission based on the number of bytes transmitted in a given time period.

t_context_state is a multi-state variable that indicates the current state of an XTP machine. It can be quiescent, listening, active or inactive.

sync_echo and *saved_sync* are local maintained values that are used to determine if the receiver state and the transmitter states are synchronized.

output is a boolean variable that is set true when an Output command is issued from the sending host. It indicates that there is data ready for transfer.

(4) *first_rec* - the first record contains first packet information and is built at the transmit host side and written to the *trans_data_buf* upon the receipt of the *Output_cmd*.

- header_rec.cmd.sreq
- header_rec.cmd.ptype
- header_rec.sync
- header_rec.seq
- header_rec.dseq
- header_rec.dlen
- first_rec.descriptor.control.service
- first_rec.descriptor.rate_req
- first_rec.descriptor.burst_req
- first_rec.descriptor.max_data

(5) *data_rec* - the data record contains data packet information and is built at the transmit host side and written to the *trans_data_buf* upon the receipt of a write command.

- header_rec.cmd.sreq
- header_rec.cmd.ptype
- header_rec.sync
- header_rec.seq
- header_rec.dseq
- header_rec.dlen
- data_rec.data(seq...seq+dlen)* - this represent the data in the queue

c. Receiver Local Variables and Context Variables

The following variable are identified as critical receiver local variables for the basic XTP procedures specified later in this chapter.

(1) *rec_data_buf* - Represents the XTP working buffer on the receiver side of the association. This buffer is used for CNTL packet assembly and transmission as well as reordering of received packets from the inbound data stream.

(2) *status_variables* - status variables are local variables that maintain state information or other characteristics of an XTP association or context. They may be implementation dependent. Values of status variables are never read directly into an

outgoing record. Status variables may be input to the enabling predicates of predicate action tables.

TABLE 2: LOCAL STATUS VARIABLES - RECEIVER

Flow Control	Rate Control	Timers	Other
<i>r_dseq</i>	<i>r_rate</i>	CTIMER	<i>r_context_state</i>
<i>r_alloc</i>	<i>r_burst</i>		<i>input</i>
<i>nspans</i>			
<i>spans</i>			

(3) Explanation of Status Variables

r_dseq is a local receiver variable that maintains the receivers value for the outgoing dseq.

r_alloc is a local receiver variable that maintains the receivers value for the outgoing alloc.

nspans indicates the number of spans represented in the spans field.

spans are ordered pairs of sequence numbers that indicate contiguous groups of received sequence numbers.

r_rate is a local receiver's variable for the default rate value.

r_burst is a local receive's variable for the default burst value.

r_context_state is a multi-state variable that indicates the state of an XTP machine. It can be quiescent, listening, active or inactive.

input is a boolean value that is set to true when a receiving host issue the an Input command. It also changes the state of the receiver to listening from quiescent.

(4) *cntl_rec* - the control record contains control packet information and is built at the receiving host side and read to the *trans_data_buf* upon the receipt of a set seq or dseq indicator.

header_rec.cmd.sreq

header_rec.cmd.ptype
header_rec.sync
header_rec.seq
header_rec.dseq
header_rec.dlen
cntl_rec.rate
cntl_rec.burst
cntl_rec.echo
cntl_rec.time
cntl_rec.techo
cntl_rec.alloc
cntl_rec.rseq
cntl_rec.nspan
cntl_rec.spans

d. *Permanent Connection Oriented Defaults*

There are a number of permanent settings that will not change during the life of a particular association. It is important to identify the existence of these values since they may effect local initialization of some variables. However, a complete list of such constants are not presented so that the specification remains less complex. If these values were changed during an association there is a risk of creating an unspecified protocol error. In the examples of this thesis it is assumed that a sustained connection oriented protocol is implemented. In a connection oriented environment the association is maintained until released through an appropriate termination message or association time-out. Initial timer values based on paradigm type are necessary as well specific options in the header command field (e.g. NOFLOW not set, indicating flow control is used.)

2. Message Types

There are nine message types defined in the XTP protocol. They are described in paragraph E, XTP STRUCTURE of Chapter IV . In the simplified protocol specification only the FIRST, DATA and CNTL message/packet types will be considered. This subset of XTP messages adequately handles a simplified connection management and data transfer procedures for the XTP transfer layer.

C. PROTOCOL SPECIFICATION AND PROCEDURES OF XTP

In this section, XTP connection establishment, data transfer and data transfer with flow, rate and error control will be specified using the SCM model using the terms of the communication structures and message types just presented. Local variables effected in the predicate action tables of the SCM will be identified, their use in any of the procedures will also be emphasized. In order to limit the scope of this thesis and simplify analysis the XTP protocol will be specified with an implementation of a reliable end-to-end communication connection.

1. Association Establishment Procedure

The association establishment procedure in XTP is similar to connection establishment of standard transport protocols. However, the only communication between machines that is required to establish the association is the transmission of a FIRST packet. The first packet travels from the association initiator (also referred to as sender, transmitter, machine 1 and Host A) to the association receiver (Host B or machine 2). In Figure 14 below the association establishment is shown.

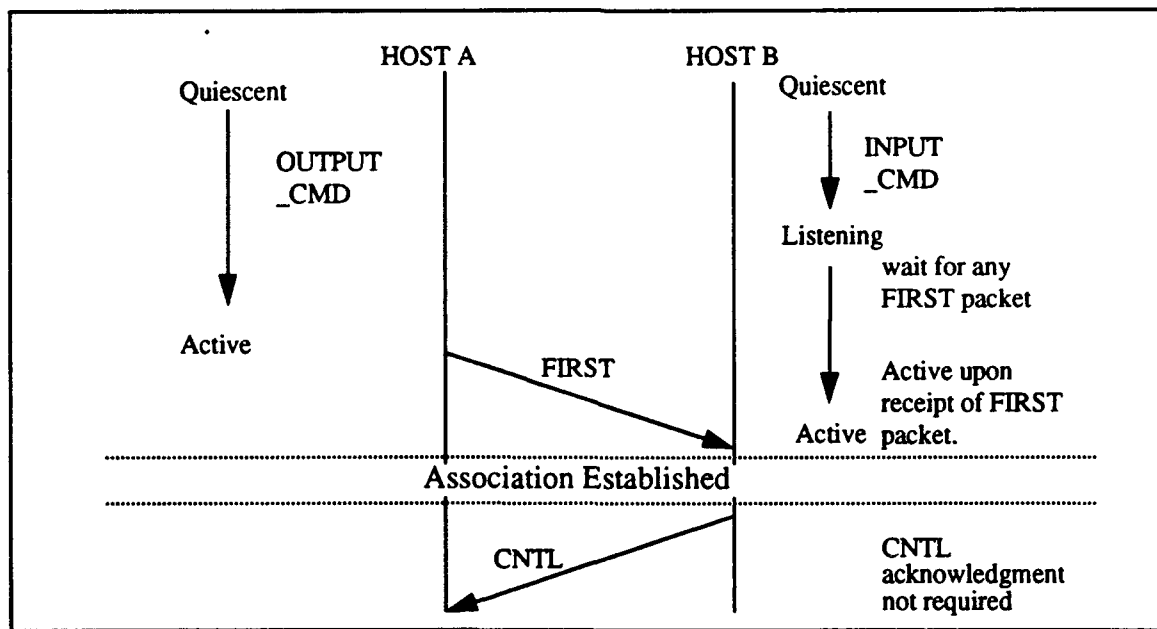


Figure 14: Association Establishment Timing Diagram

Of particular note in this implementation is the presence of a return control packet (CNTL) for the association set-up. There is no XTP requirement for such an explicit acknowledgment (CNTL packet). Rather, the requirement for the CNTL acknowledgment is generated from the chosen reliable end-to-end communication connection paradigm implementation. Thus, the handshaking process shown here is not a burden to the XTP protocol but to the implementation. In XTP the acknowledgment of the FIRST message would come with a later CNTL packet or simply remain implicit. The reliability of the underlying physical network "guarantees" the success of the implicit association establishment. The use of the CNTL packet this early in the paradigm is a simplifying and costly implementation in terms of connection set up time and overhead.

Another significant advantage of XTP is that the transmission of data from Host A to Host B can begin immediately and need not wait for a handshaking CNTL or ACK process to complete. In fact there is a mechanism that allows data to be included in the FIRST packet. This is particularly useful when the entire message can be included in the association establishment (FIRST) packet. And is best used in short lived associations where the entire message is included in the FIRST packet information segment (e.g. VMTP transaction type messages).

XTP can therefore escape from the classical handshake delays of some protocols (e.g. TP 4 and TCP) that require the added network time in low or medium speed environments and do not currently provide for the inclusion of significant amounts of user data.

The state transition diagram presented next in Figure 15, retains the overhead that is evident in handshaking methods of previous transport layer protocols due to the implemented paradigm. The acknowledgment sent back immediately through the CNTL packet, as mentioned, is not an absolute XTP requirement, but in this example where a reliable diagram paradigm is being built, the CNTL acknowledgment allows an immediate confirmation of association establishment.

The finite state machines in Figure 15 thus represents handshaking behavior. This XTP association establishment implementation reverts to this inefficient means of association set up if the confirmation of the association is required before the transmission of data is allowed. The *-FIRST* represents sending an association establishment request, *+FIRST*, receiving the request, *-CNTL*, send acknowledgment of the request, and *+CNTL*, receive the control and acknowledgment information. As per the definition of the SCM model, two channels, one from machine 1 to machine 2 and one from machine 2 to machine 1 are introduced. The S on state 0 of both machines represents the initial/starting state. In the state transition diagram there is a dotted arrow that indicates that data may be sent prior to the *+CNTL*.

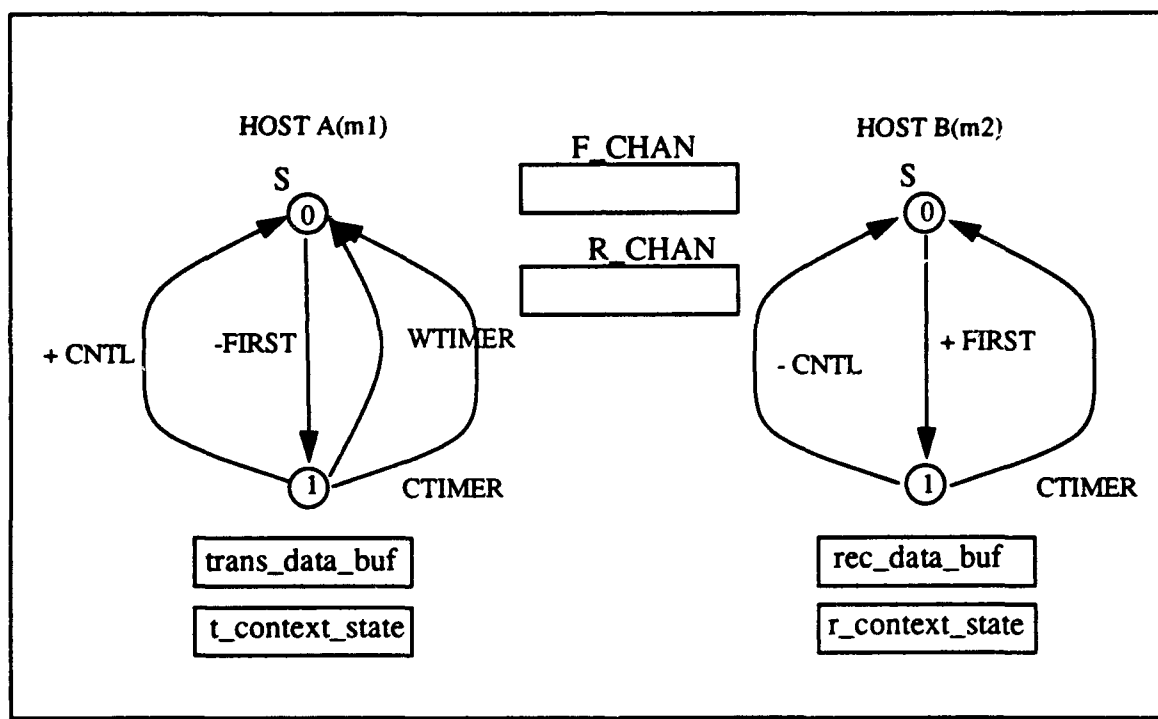


Figure 15: Association Establishment State Transition Diagram

An alternative to the SREQ indicator that forces the *-CNTL* response is the DREQ. The DREQ is similar to the SREQ except that it requires the delivery of data to the receiving implementation prior to sending a CNTL message back to the requestor.

A different global variable, the DSEQ value in the FIRST packet indicates the starting sequence number on the return path data stream. The Sync_Echo function at the transmit end uses the values of sync and sync_echo to perform a state machine handshake operation, which checks to see if state machines are logically together. The predicate action table for the association set up implemented is shown in Table 3.

TABLE 3: ASSOCIATION ESTABLISHMENT PREDICATE ACTION TABLE

Transition	Enabling Predicate	Action
- FIRST	F_CHAN = empty output = true t_context_state = quiescent	F_CHAN <- first.rec trans_data_buf <- data retry_count = k seq <- seq + dlen t_context_state <- active Start(WTIMER) Start(CTIMER)
+ FIRST	F_CHAN = \emptyset rec_data_buf = \full F_CHAN.CMD = FIRST r_context_state = listening	rec_data_buf <- F_CHAN r_context_state <- active Send_CNTL(sreq) reset(CTIMER)
- CNTL	R_CHAN = empty r_context_state = active Send_CNTL(sreq) = true	R_CHAN <- cntl_rec Clear(rec_data_buf) Start(CTIMER)
+ CNTL	R_CHAN = \emptyset CMD.PTYPE = CNTL t_context_state = active	trans_data_buf <- R_CHAN stop(WTIMER) reset(CTIMER)

- if a FIRST message is already sent, a repeat FIRST message cannot be sent on the same association, the implementation prevents this by changing the t_context_state to active.
- when first_rec.sreq is set true, a return CNTL is required, this causes the association establishment process to perform like a stop-and-wait protocol.
- Send_CNTL procedure initiates the sending of a CNTL packet and the needed processing to prepare the return CNTL packet and the rec_data_buf.

2. Data Transfer Procedure

As mentioned, data transfer in XTP is accomplished in two ways, through the DATA packet or through a FIRST packet (that is used only once per context.) To simplify

this specification only the DATA packet will be examined as a means of data transfer. The data transfer process uses the mechanisms present in the DATA packet structure and the return information available from CNTL messages to control data flow. These procedures are separate from the association establishment procedures discussed previously.

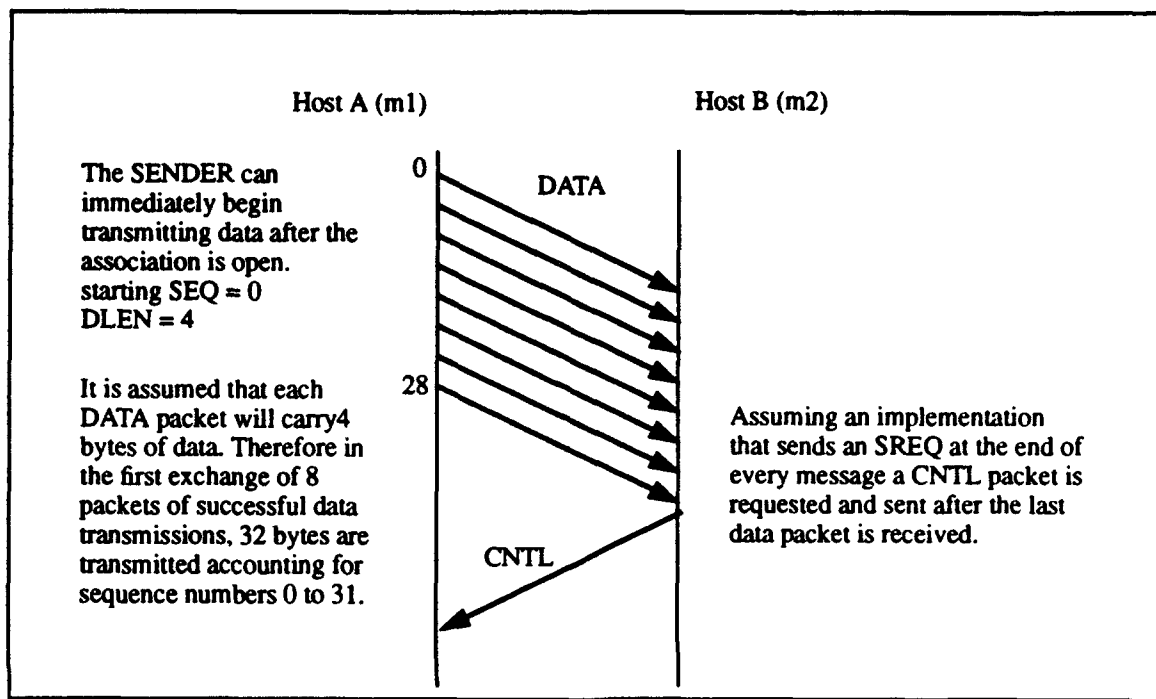


Figure 16: Data Transfer Timing Diagram

In the figure above it is shown how a nominal sized message of 32 bytes could be sent in a burst of eight packets from machine 1 to machine 2. It is assumed that each DATA packet can carry up to 4 bytes, this simplifies the explanation, actual byte quantities are limited by the physical frame size, less the XTP packet overhead. So far in this example no need for error, flow or rate control is needed or introduced, a perfect channel is assumed that has no problems with network congestion or host or buffer space. In Figure 17 the

basic state transition diagram data transfer procedures is shown. The Association Establishment phase is not included.

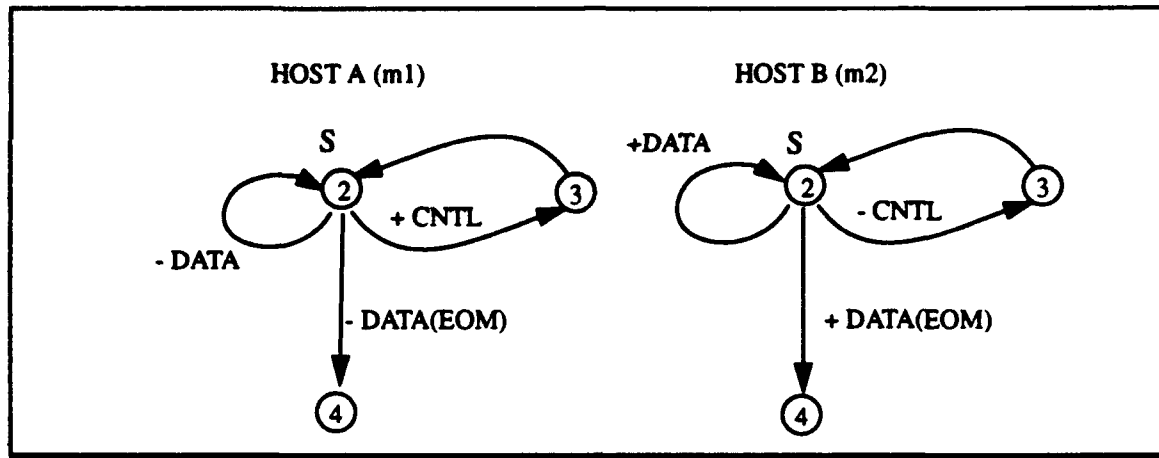


Figure 17: Data Transfer State Transition Diagram

The starting state for this diagram is specifically chosen as state 2 so as to distinctly specify the data transfer procedure from association establishment. The transition for the -CNTL merely indicates that some processing is required for the CNTL packet.

A predicate action table based on this simple data transfer example is now presented.

TABLE 4: DATA TRANSFER PREDICATE ACTION TABLE

Transition	Enabling Predicate	Action
- DATA	F_CHAN = empty t_conext_state = active output = true	trans_data_buf <- header_rec & data_rec(seq...seq + dlen - 1) if EOM = true, then set SREQ F_CHAN <- trans_data_buf seq <- dlen + seq
+ DATA	F_CHAN = \empty rec_data_buf = \full CMD.PTYPE = DATA r_context_state = active	rec_data_buf <- F_CHAN Send_CNTL(sreq) reset(CTIMER)
- CNTL	R_CHAN = empty r_context_state = active Send_CNTL(sreq) = true	R_CHAN <- rec_data_buf

TABLE 4: DATA TRANSFER PREDICATE ACTION TABLE

Transition	Enabling Predicate	Action
+ CNTL	R_CHAN = \emptyset CMD.PTYPE = CNTL t_context_state = active	trans_data_buf <- R_CHAN stop(WTIMER) reset(CTIMER)

The Send_CNTL(sreq) function will return true in one of two cases. In the first case the incoming SREQ global variable is set and transferred to the local variable of the same name. In this case a CNTL packet will be returned to the sender. In the second case, the receiver will process the available state information and determine if a CNTL message should be sent based on available buffer space, rate information and window size. If the receiver determines that the sender needs to be updated a CNTL message will be sent.

Data transfer is not this simple however. The potential for network congestion exists in any network that has more than a nominal load, the probability of congestion increases even more with the increased processing and buffer demands on local hosts in high speed environments. For example, if buffers are undersized or if transmitted data pools at intermediate or destination buffers, due to slower or excessive processing, the likelihood of lost data increases. Flow control is now introduced into our data transfer example. To implement flow control, use will be made of the local variables introduced earlier.

3. Data Transfer Procedure with Flow Control

Flow control is an important part any transport protocol, it is perhaps more important for high speed protocols. Flow control in XTP is accomplished with credit based sliding window mechanisms. Conceptually, the XTP transmitter can look into the receiver, see the window, and determine if more data can be sent. Thus a decision is made to send or not send more data. This mechanism is separate from rate control mechanisms discussed later.

When an association is initiated a default window size is determined by the implementation. This window size is defined by the starting sequence number (seq) and is upper bounded by a default allocation (default_alloc) value. Assuming a window size

larger than the message size produces unrestricted data flow as seen in Figure 16 on page 53.

Flow control mechanisms using the same data stream transmissions as before are now considered (see figure 18 on page 57). In the example, if the window size is restricted to just three packets, demonstrating a reduced start up allocation at the transmitter. The transmitter must now wait for the release of additional window space from the receiver before transmitting additional data. Additional window space is obtained from the receiver in one of two methods. The first is a credit or allocation increase by the receiver. The second is to receive acknowledgment of sequence numbers successfully sent to the receiver's user data buffer space.

The XTP credit or allocation mechanism begins with the transmitter assuming a window size using the `default_alloc` value as mentioned in the previous paragraph. Assuming a small value for `t_alloc` increases the likelihood of first time acceptance of the data packet without error due to buffer overflow. A current context value for this allocation is then maintained in the transmitter allocation variable (`t_alloc`). When a CNTL message is returned to the transmitter a new allocation (CNTL.ALLOC) global variable is sent. This new receiver generated allocation value (from `r_alloc`) is the new credit value applied to the transmitter's available window size. The ALLOC value can be changed with each CNTL packet with the ability to maintain, increase, decrease or cutoff the flow of data thus making the window size a dynamic value as it slides over the sequence number space. This process gives the transmitter visibility of the receivers available buffer space and the upper edge of the sliding window.

The lower edge of the sliding window is now controlled through the delivered sequence number (`r_dseq`) value originated from the receiver. The `r_dseq` value is thus similarly sent to the transmitter in a `HEADER.DSEQ` variable of a CNTL packet. It indicates the lower edge of the sliding window, that is the sequence numbers up to all lesser sequence numbers are acknowledged as delivered.

In Figure 18 below a data transfer with flow control mechanisms in place is demonstrated. The sliding window protocol works in the following manner: host A may send up to its window size, as it sends packets it decreases the available window. Host B receives packets and acknowledges them through the CNTL packet mechanism. Host B can also increase the window size through the allocation value sliding it forward. The process terminates when the end of the message is reached, and the last packet acknowledged. Both acknowledgments and credit allocations may effectively increase the available window size and thus slide the window.

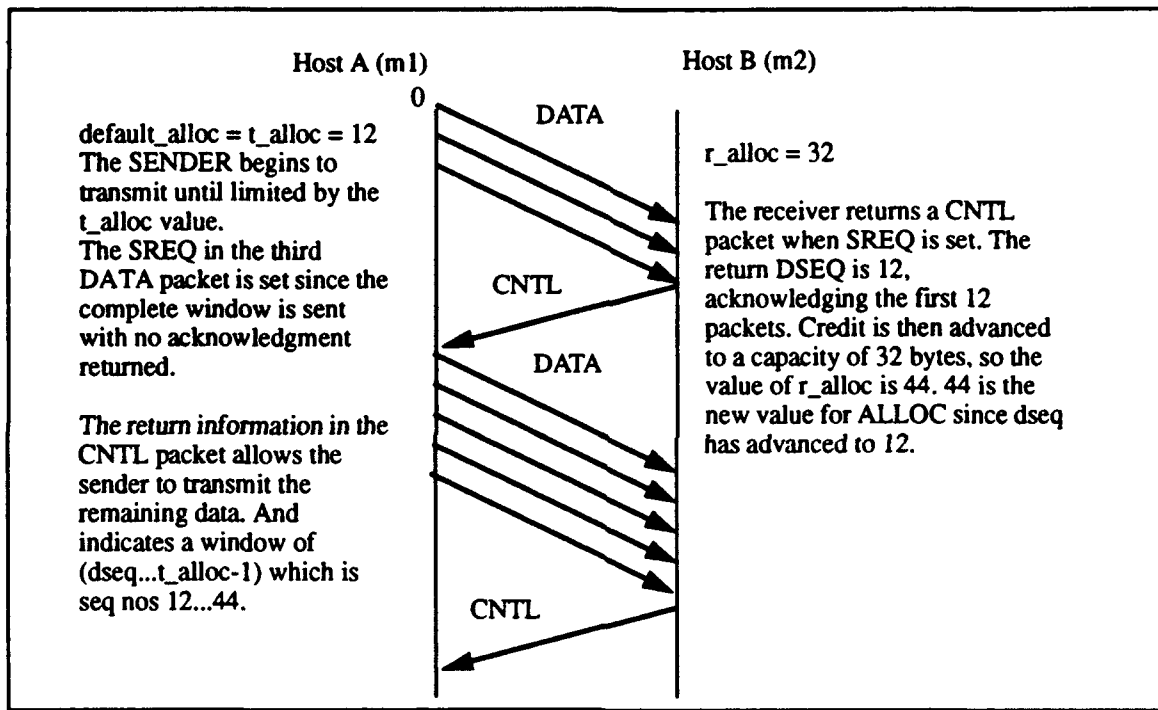


Figure 18: Data Transfer Timing Diagram with Flow control

In the figure above the same nominal sized message of 32 bytes is represented. However, it is assumed that flow control is now needed, this could be due to limited receiver buffer space. Still a near perfect channel is assumed that requires only flow control but so far avoids problems with transmission errors. The larger available sequence number space of XTP reduces the wait for a return control packet since the transmitter can continue to send while waiting.

In Figure 17 the state transition diagram for data transfer is enhanced to include flow control procedures. The Association Establishment phase is included and uses the XTP method that avoids waiting for an explicit start-up handshake.

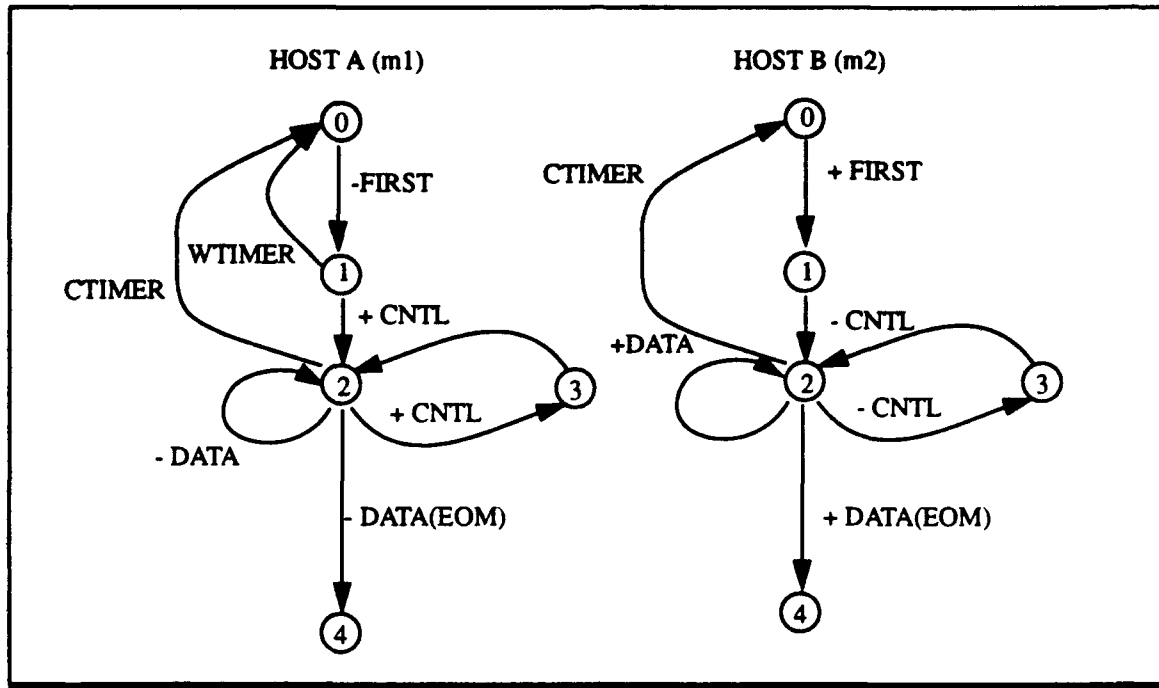


Figure 19: Data Transfer State Transition Diagram with Flow Control

The transition for the -CNTL merely indicates that some processing is required for the CNTL packet.

A predicate action table based on this data transfer with flow control is presented next.

TABLE 5: DATA TRANSFER PREDICATE ACTION TABLE WITH FLOW CONTROL

Transition	Enabling Predicate	Action
- DATA	F_CHAN = empty t_conext_state = active output = true window_open	trans_data_buf <- header_rec & data_rec(seq...seq + dlen - 1) if EOM = true, then set SREQ F_CHAN <- trans_data_buf seq <- dlen + seq t_alloc <- t_alloc - dlen if t_alloc ≤ 0 , then window_closed & set SREQ

TABLE 5: DATA TRANSFER PREDICATE ACTION TABLE WITH FLOW CONTROL

Transition	Enabling Predicate	Action
+ DATA	F_CHAN =\ empty rec_data_buf =\ full CMD.PTYPE = DATA r_context_state = active	rec_data_buf <- F_CHAN r_window = (r_dseq, r_alloc-1) reset(CTIMER) Send_CNTL(sreq)
- CNTL	R_CHAN = empty r_context_state = active Send_CNTL(sreq) = true	R_CHAN <- rec_data_buf
+ CNTL	R_CHAN =\ empty CMD.PTYPE = CNTL t_context_state = active	trans_data_buf <- R_CHAN t_window = (t_dseq...t_alloc-1) stop(WTIMER) reset(CTIMER)

- dlen is limited by the maximum number of data bytes that me place in a DATA packet.
- The initial dseq value at the transmitter is equal to the starting sequence number, seq, in this example it is assumed to be zero(0).
- The initial value of t_alloc is equal to the original transmitter window size, it is a quantity of bytes, in this example it is 12.
- The initial value of r_alloc is equal to the original receiver window size, it is a quantity of bytes, in this example it is 32, it will not be communicated to the transmitter until a CNTL packet is sent.
- - DATA may be repeated until the window size is exhausted, or not closed (dseq +1 =\ t_alloc).

As before the Send_CNTL(sreq) function will return true in one of two cases. In the first case the incoming SREQ global variable is set and transferred to the local variable of the same name. In this case a CNTL packet will the be returned to the sender. In the second case the receiver will process information based on available buffer space, rate information and window size. If the receiver determines that the sender needs to be updated a CNTL message could be sent.

Data transfer even with flow control is not this simple however. The potential for network congestion exists in any network that has more than a nominal load, the probability of congestion increases even more if buffers are undersized or if transmitted data pools at intermediate or destination buffers. As congestion increases so does the probability of errors. But before discussing error control, rate control is discussed.

4. Data Transfer Procedure with Orthogonal Rate Control

As seen in the last section flow control is an important part of any transport protocol, rate control is equally important when operating in high-speed environments. Rate control applies to and limits the production of data for each segment of a transmitting data path and thus by extension the rate control for the overall end-to-end path. In XTP the end-to-end path is the summation of all its sub-paths that make up the association from the initiating host to the receiving host. Rate control on each node-to-node segment or sub-path prevents the overrunning buffers at the next intermediate node and is association independent.

Rate control in XTP is accomplished with a separate credit and timer mechanism that makes rate control orthogonal from flow control. Although the CNTL packet is again used to transfer information, rate information is transmitted through separate rate parameters.

Conceptually, the XTP transmitter (data producer) is prevented from overrunning the next down stream (consumer) node on the path through adherence to a maximum burst limit that is set by the receiver. The burst limit is monitored at the transmitter through a credit value, **credit** (in bytes), and a timer value, **RTIMER**. These values together determine a rate (in bytes/second) that may not be exceeded. The credit value is the maximum number of bytes that can be transmitted in the time period set by **RTIMER**. When **RTIMER** expires, credit is reset. If the credit is used up the node in the path must cease transmitting. [STRA92b]

Since the data rate is controlled by receiving end nodes, but applies to the node-to-node path for all associations using that path, a bandwidth allocation scheme is possible that limits the bandwidth for each association of a network. This use for rate control is an important mechanism in a high-speed network where upper bounds on rate controls may be needed to prevent greedy end users from abusing bandwidth.

When an association is initiated a default rate and burst value are used until a CNTL message indicates the receivers directed values for rate and burst. Additionally, the

transmitter can request through the FIRST packet, a value for the rate and burst fields. These fields are used to directly determine the values of credit and RTIMER and are updated with receipt of each CNTL message.

In Figure 18 below a data transfer with rate control mechanisms in place is demonstrated. Host A may continue to send until its credit value is used up. When the RTIMER value expires credit is reallocated. Each time a packet is sent the rate credit value is decremented. The credit value may be changed by an update through the burst size or rate time period in the returning CNTL packet. Updated values would effect the transmitter on the next expiration of RTIMER.

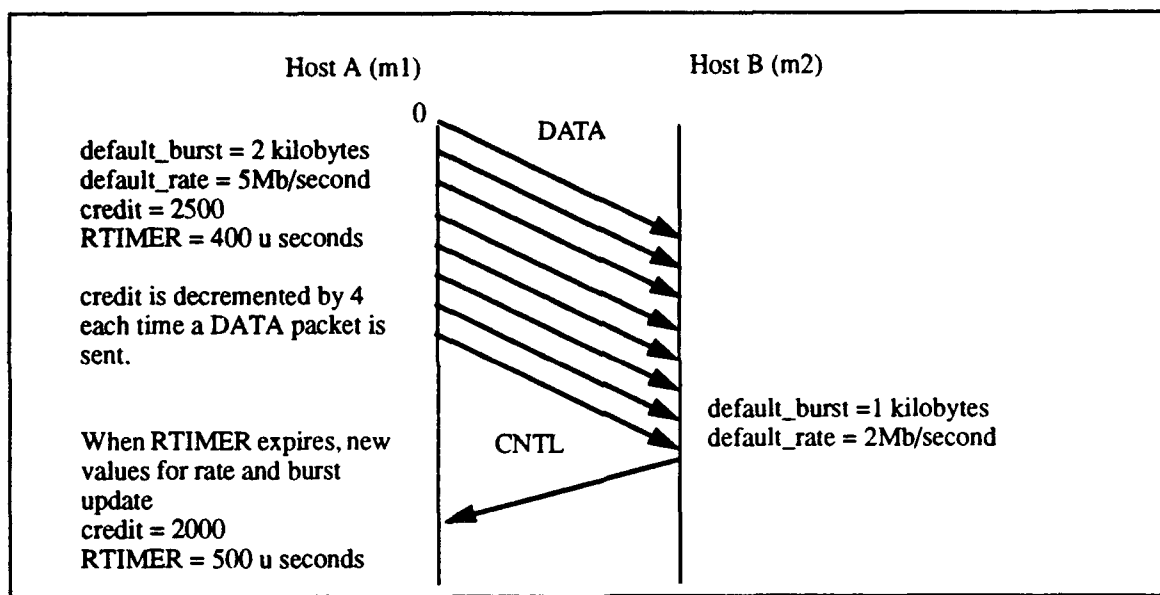


Figure 20: Data Transfer Timing Diagram with Rate Control

In the figure above if it is assumed that a FDDI network is being used with a total bandwidth of 100Mbps. A rate value of 5Mbps is chosen for a single station. A nominal data envelope of 2kilobytes per burst is chosen. The credit and RTIMER are then calculated. A CNTL packet is then sent that reduces the rate of the path, reducing the credit to 2000 bytes per second, and lengthening the refresh timer to 5 microseconds.

In Figure 17 the state transition diagram for the data transfer is enhanced to include rate control procedures.

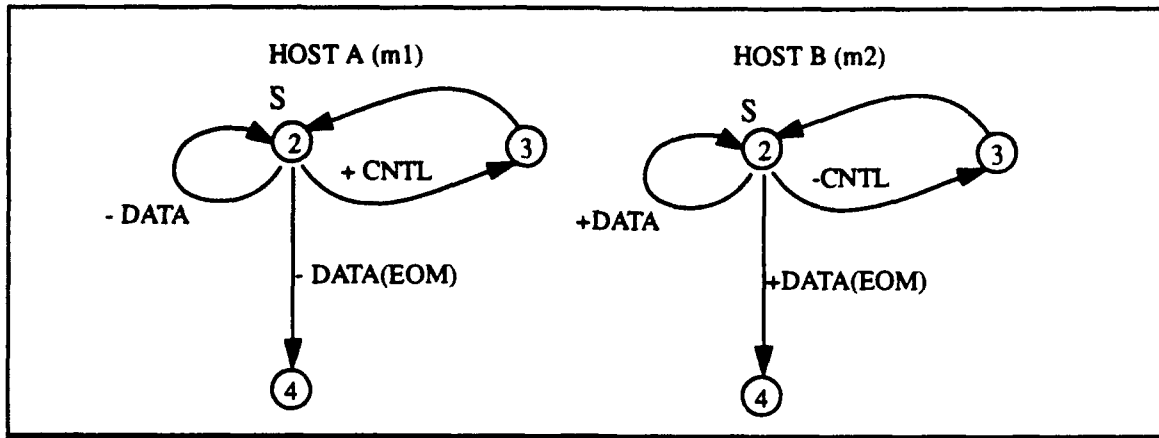


Figure 21: Data Transfer State Transition Diagram with Rate Control

The starting state for this diagram is specifically chosen as state 2 so as to distinctly specify the data transfer procedure from association establishment. The transition for the -CNTL merely indicates that some processing is required for the CNTL packet.

Unfortunately this state diagram does not clearly indicate the benefits or the operation of the orthogonal rate control. Orthogonality is suggested since the state diagram is not altered from the previous data transfer diagram. An investigation of the predicate action table however will show how rate control effects data transmissions. The reason for this is that the rate control constraints are shown as enabling predicates.

A predicate action table based on this data transfer with rate control is presented next. The flow control mechanisms remain in the table.

TABLE 6: DATA TRANSFER PREDICATE ACTION TABLE WITH RATE CONTROL

Transition	Enabling Predicate	Action
- DATA	F_CHAN = empty t_context_state = active output = true window_open credit /= 0	trans_data_buf <- header_rec & data_rec(seq...seq + dlen - 1) if EOM = true, then set SREQ F_CHAN <- trans_data_buf seq <- dlen + seq t_alloc <- t_alloc - dlen if t_alloc ≤ 0, then window_closed & set SREQ Start (RTIMER) (once) credit = credit - dlen if RTIMER = 0, reset(credit) and restart (RTIMER)
+ DATA	F_CHAN = \empty rec_data_buf = \full CMD.PTYPE = DATA r_context_state = active	rec_data_buf <- F_CHAN r_window = (r_dseq, r_alloc-1) reset(CTIMER) Send_CNTL(sreq)
- CNTL	R_CHAN = empty r_context_state = active Send_CNTL(sreq) = true	R_CHAN <- rec_data_buf (updated rate and burst parameters are included in CNTL)
+ CNTL	R_CHAN = \empty CMD.PTYPE = CNTL t_context_state = active	trans_data_buf <- R_CHAN t_window = (t_dseq...t_alloc-1) stop(WTIMER) reset(CTIMER)

As before the Send_CNTL(sreq) function will return true in one of two cases and allow an update through the CNTL packet. The added dimension with rate control is that additional processing is incurred at the receiving host to determine if rate and burst parameters require updating. This determination is beyond the scope of this thesis. Of particular note for rate control is the RTIMER operation. In this model the RTIMER is only started once, it is then restarted only upon each expiration. In the predicate action table this is shown by the restart(RTIMER) operation initiated upon RTIMER expiration.

5. Data Transfer Procedure with Error Control - Selective Repeat

Error control in XTP is initiated through error identification (checksums), lost packet identification and out of order packet delivery. When error conditions exist retransmission can be implemented with a selective repeat method that saves excessive retransmission in high-speed environments. A look at the selective repeat will show how XTP is able to conserve bandwidth by only retransmitting the data that was received in error. When an error occurs, within the active window, the receiver identifies to the transmitter the gaps that require retransmission. Correctly received data is held at the receiver until just missing gaps are resent.

Error control mechanisms using the same data stream transmissions as before are now considered (see figure 18 on page 57). The loss of data creates a gap, this changes the timing diagram and the diagram thus reflects the retransmission of a lost gap. Information, through the RSEQ field, on the successfully delivered data, releases buffer space back to the transmitters sliding window.

The timing diagram below demonstrates how lost data may be recovered. The window size, being arbitrarily small however conceals the ability of XTP to continue to transmit in the presence of errors.

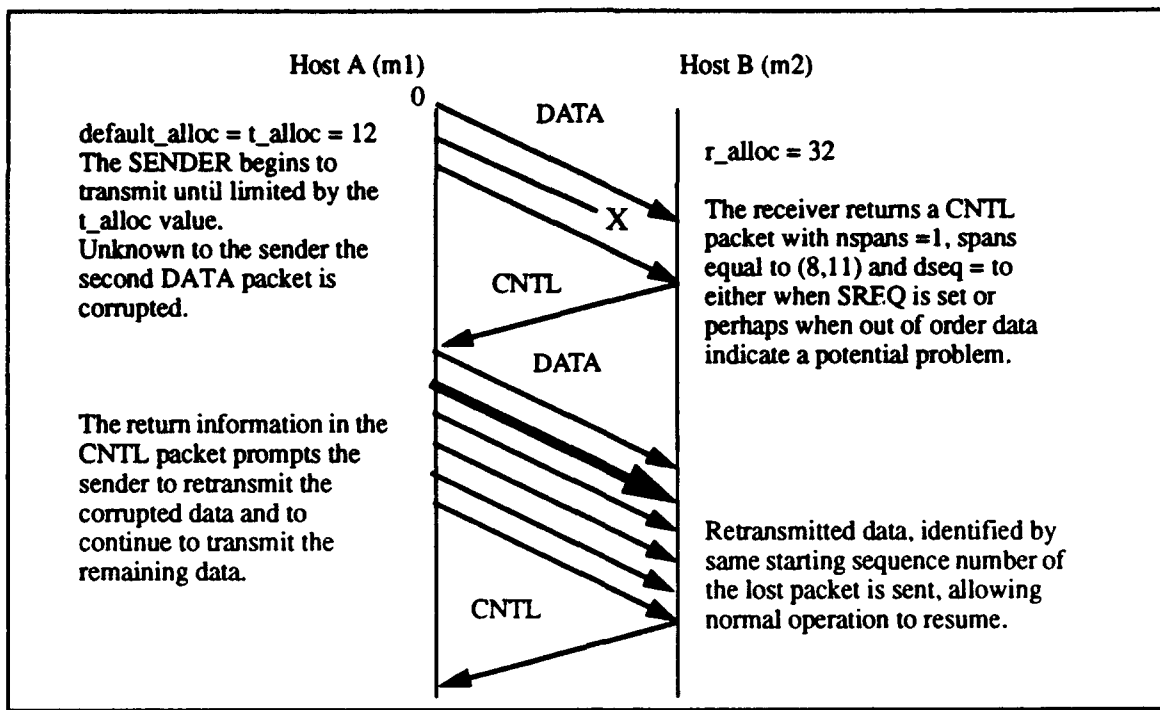


Figure 22: Data Transfer Timing Diagram with Error Control

In the figure above the same nominal sized message of 32 bytes is represented. However, it is assumed that error control is now needed, this could be due to a corrupted XTP key tag that holds addressing information for each context causing the packet to be lost.

The same state diagram applies, shown again below in Figure 23

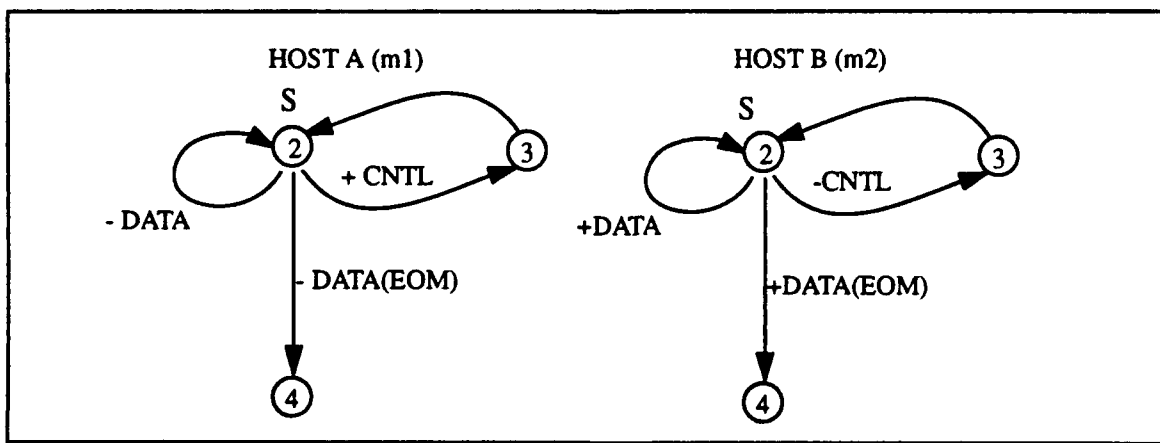


Figure 23: Data Transfer State Transition Diagram with Rate Control

The starting state for this diagram is specifically chosen as state 2 so as to distinctly specify the data transfer procedure from association establishment. The transition for the -CNTL merely indicates that some processing is required for the CNTL packet.

A predicate action table based on this data transfer with error control is presented next.

TABLE 7: DATA TRANSFER PREDICATE ACTION TABLE WITH FLOW CONTROL

Transition	Enabling Predicate	Action
- DATA	F_CHAN = empty t_context_state = active output = true window_open	trans_data_buf <- header_rec & data_rec (gap & seq...seq + dlen - 1) if EOM = true, then set SREQ F_CHAN <- trans_data_buf seq <- dlen + seq t_alloc <- t_alloc - dlen if t_alloc ≤ 0, then window_closed & set SREQ
+ DATA	F_CHAN = \empty rec_data_buf = \full CMD.PTYPE = DATA r_context_state = active	rec_data_buf <- F_CHAN r_window = (r_dseq, r_alloc-1) reset(CTIMER) Send_CNTL(sreq)
- CNTL	R_CHAN = empty r_context_state = active Send_CNTL(sreq) = true	R_CHAN <- rec_data_b critical fields in CNTL for retransmission are RSEQ, NSPANS and SPANS
+ CNTL	R_CHAN = \empty CMD.PTYPE = CNTL t_context_state = active	trans_data_buf <- R_CHAN t_window = (t_dseq...t_alloc-1) CONVERT(npans,spans) to gap info stop(WTIMER) reset(CTIMER)

- In selective retransmission when Send_CNTL(sreq) indicates the need for a CNTL packet, nspans and spans information is included in the packet.
- The returning values of HEADER_REC.DSEQ and CNTL_REC.RSEQ indicate the two forms of XTP acknowledgment. DSEQ signifies successfully delivered data to the XTP user, RSEQ indicates data that has been received at the destination endpoint. Data with a sequence number less than DSEQ may be released, however release of values less than RSEQ cannot be released until the window slides past.
- The nspans value is the number of spans being sent. In the spans field ordered pairs of correctly received bytes exist. Therefore the transmitter tracks outstanding data with local status variable bseq and eseq, which are implementation variables and not

specified within XTP.

- **CONVERT(npans, spans)** is an implementation procedure that uses the returned span information to calculate the gaps in the data stream sent to machine 2. **CONVERT** itself is not part of the XTP definition, but a process like it will be needed in any XTP selective repeat error control implementation.

Finally, it must be noted that there are two alternative methods, within the XTP definition, for error control. The first is **FASTNAK** which is a command options bit set by the transmitter, it indicate that a **CNTL** packet must be returned immediately upon detection of out of order data. This method is useful when out of order data is not likely.

The second is **NOERR**, that allows the disabling of the retransmission mechanism. In this case the **NOERR** bit in the command field would be set, indicating that no retransmission policy would be in effect. This is another example of both the flexibility and the complexity of XTP that provides the user with powerful high-speed data transfer options.

V. ANALYSIS OF XTP

A formal SCM specification of the XTP protocol is given in the previous chapter for association establishment and data transfers with flow, rate and error control. After protocol specification, the next step is to analyze the specification to verify that the protocol is free from logical errors such as deadlocks, unspecified receptions, unexecuted transitions and blocking loops. This chapter presents the XTP protocol analysis.

The system state analysis presented will be performed in much the same manner as in previous works on other protocols by [LUND91b], [MCAR92], [ROTH92] and [TIPI93]. Due to time constraints a software simulation of the SCM analysis, using a tool refined and presented by [BULB93], was not conducted but is recommended as future work. Even though there were time limitations that prevented further in-depth scrutinizing of the XTP protocol, a moderate degree of confidence in the correctness of the protocol was gained. Confidence in XTP was gained through XTPs ability to mimic communication paradigms that already have undergone extensive analysis in their own right, even though multiple paradigm implementations added to overall XTP protocol complexity.

A. ASSOCIATION ESTABLISHMENT

The association establishment portion of the protocol will be analyzed using SCM. This analysis is simpler when using the implemented reliable end-to-end communication connection paradigm than includes the responding CNTL packet. Without this CNTL packet, the acknowledgment of the association is integrated with a responding CNTL packet for a future data transfer. Using a future CNTL packet is more efficient in terms of protocol performance, but representing this in the SCM state diagram is more complicated. Thus the simpler diagram of the reliable connection is analyzed.

The association establishment is the same as specified in Chapter IV. The specification, as represented by the SCM model, is shown as a set of finite state machines and a predicate-action table.

The same two machines as before are shown in Figure 24. Local and global variables are included in addition to the XTP timers. The local buffer variables in Host A and B can have the values of first_rec (a FIRST packet record), cntl_rec (a CNTL packet record), and E(empty). The context state variables maintain the current state of a given machine and in accordance with the logic of the predicate action table will not allow the The initial value for trans_data_buf is FIRST and the initial values for all other variables is E. The system global variable, F_CHAN can have the same values as the local variables.

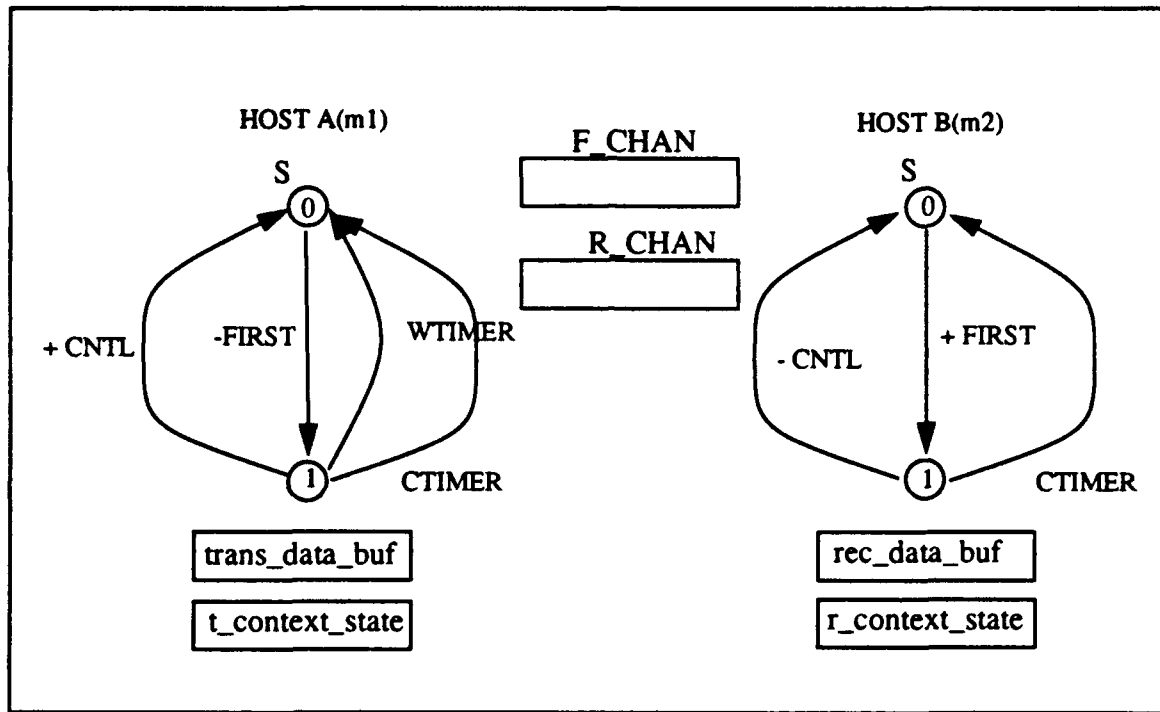


Figure 24: SCM Model for Association Establishment

The predicate-action table is shown in Table 3, "ASSOCIATION ESTABLISHMENT PREDICATE ACTION TABLE," on page 52.

The global state reachability and system state reachability graphs are found in Figure 25 and Figure 26. The procedures for developing the global and system state analysis are prescribed in [LUND91b] and applied in [ROTH92]. A similar approach is taken now.

The format for the global state tuple for the XTP association establishment is:

[Host_A_state, trans_data_buf, F_CHAN, R_CHAN, rec_data_buf, HOST_B_state]

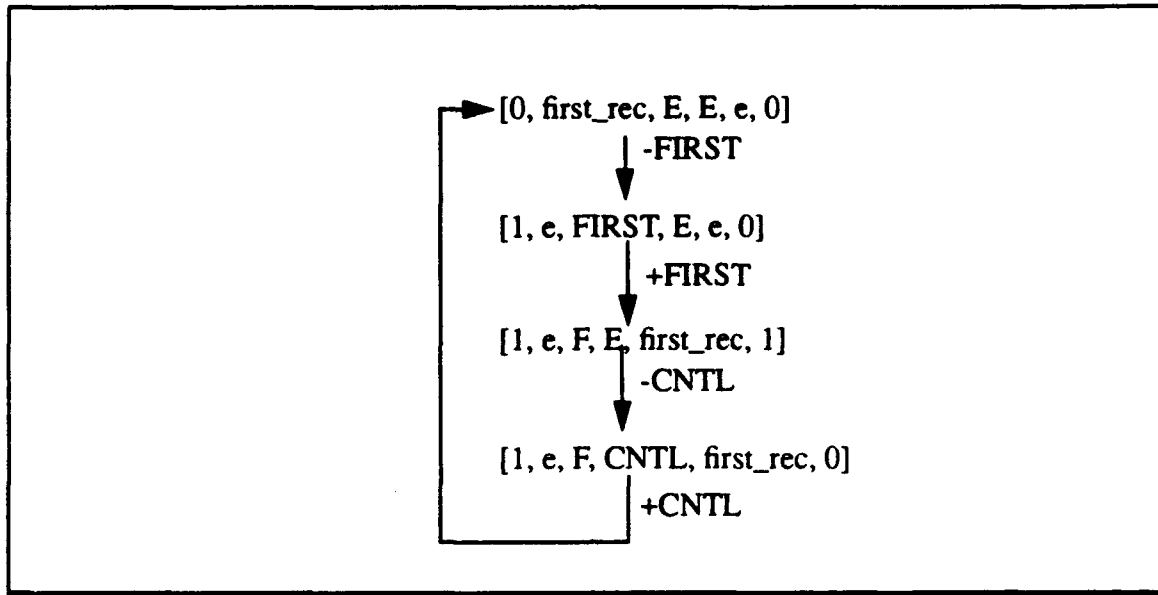


Figure 25: SCM, Global Reachability Analysis for Association Establishment

The format for a system state tuple for all cases of analysis is:

[Host_A_state, Host_B_state]

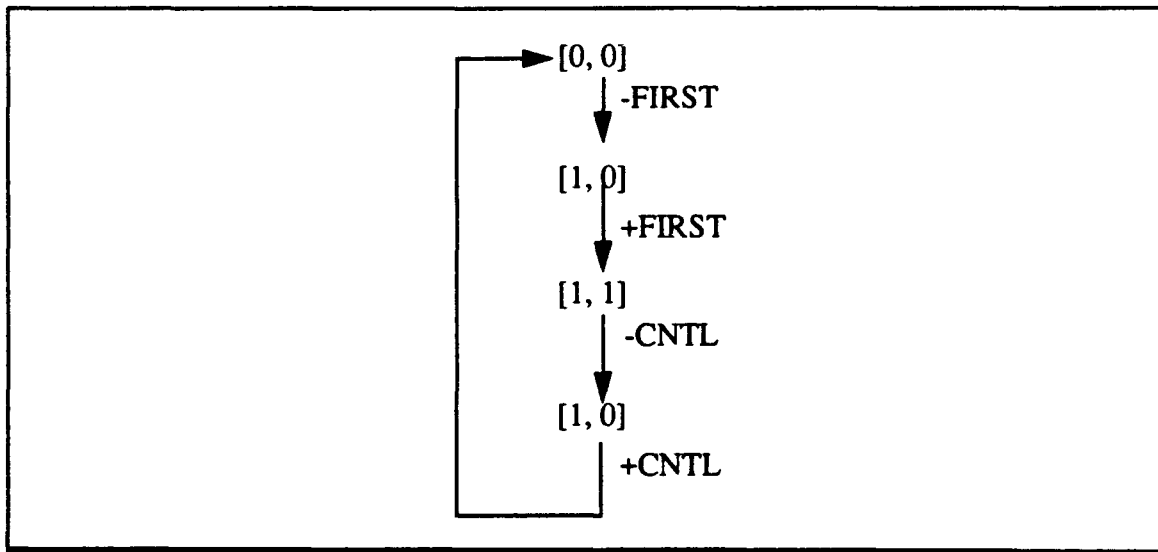


Figure 26: SCM, System Reachability Analysis for Association Establishment

In the SCM model the behavior of the association establishment can be clearly and quite adequately represented, although timers have so far been ignored. When timers are introduced the creation of alternate directed links on the global analysis graph back to the starting state are introduced. The time out links prevents a deadlock in cases where there is a transmission failure of either the -CNTL or the - FIRST.

The flow of the association connection is as follows: the sender places a CNTL packet on the open F_CHAN queue to the receiver. This task is conducted only when the sender is in a quiescent state. Timers are then initiated along with the transition of HOST A to an active context state. It is then assumed that the receiver is in the listening state. If the receiver is not in the listening state, it is then obviously not ready to start an association. In this case HOST A will time out. In normal operation HOST B in the listening state senses a packet on the incoming channel and accepts the message from the channel, removing that message from the incoming channel. Upon reading the command field, and determining that the packet is a FIRST packet the receiver enters the active state. The association is now established.

The receiver now sends an acknowledgment packet to the HOST A. The HOST A senses the acknowledgment packet and is clear to send another packet of information to the receiver. This completes the reliable connection oriented paradigm.

The finite state machines in Figure 24 represent the behavior of the definition of the association establishment procedure. The global reachability analysis graph shown in Figure 25 is free from deadlock, unspecified receptions, and unexecuted transitions. This is evident due to the loop nature of both Figure 25 and Figure 26.

B. DATA TRANSFER

In this section the XTP selective repeat procedure will be analyzed. The selective repeat procedure in any high-speed transport protocol enables greater throughput over previous Go-Back-N methods, that required retransmission of data from the point of the

error. With selective repeat, transmission may continue up to the size of the window, meanwhile only data sent in error is retransmitted.

Selective repeat in XTP uses spans information (explained in Figure 11 on page 35 and in the Error Control section of Chapter III) is used to direct the retransmission operation. Using spans is beneficial if data errors are grouped, since a corrupted group of bytes is easily identified as a missing span or gap of successfully received data and can thus be retransmitted as one block.

1. XTP Error Control - Selective Repeat

The XTP procedure for error control, selective repeat, is chosen as a means to further investigate the performance of the XTP protocol and its freedom from logic errors. Two machines have been defined in the previous chapter, HOST A (m1), a sender and HOST B (m2), the receiver. The initial state for both machines is state 2, this merely suggests that states 0 and 1 are past, since they were already used in the association set up procedure. Two assumptions were made for the analysis. First, all CNTL packets transmitted were received without error and second, only one data packet was lost during the original transmission.

The original specification for the sender is found in Figure 23 on page 65. As the XTP buffer manager places data in the next available sequence number, the sender places the packet on the channel and increments the sequence number for the next packet to be transmitted. As long as the next packet is not tagged "end of message" (EOM), the sender will continue this process until the bottom state on the finite state machine is reached, indicating the transmission of a full window. Acknowledgments in the form of CNTL packets are passed back to the transmitter when status requests bits (SREQ or DREQ) are set. If a CNTL is received then the transmitter must determine, using the rseq and the allocation value, if the window may be opened more and if so, how far. This is accomplished by sliding the window's lower edge to rseq and the upper edge to rseq plus the allocation. The rseq value indicates the highest value for which all lessor sequence

numbers are acknowledged. Additionally, if the CNTL indicates non-continuous reception of sequence numbers starting from the first packet for which an acknowledgment is expected (rseq), span information will indicate what gap of missing data will require retransmission.

The specification from the previous chapter is now enhanced to show more of the detail in the selective repeat procedure. The XTP selective repeat procedure is adapted to the general selective repeat analysis performed in [ROTH92]. The data in this case along, with some packet information in the header is transferred to the receiver. The problem now faced is how to represent either a continuing data reception or the generation of a CNTL packet to acknowledge data sent. In the previous work each data packet received would cause the generation of an acknowledgment packet, that would update the transmitter on the status of data sent and acknowledged.

In XTP the choices at state 3 of the receiver are more difficult to represent and analyze. In the previous chapter a looping receive data was shown. Although an accurate description, this type of behavior is difficult to analyze. Looping tends to simply generate an explosive number of states that quickly becomes unmanageable. With this looping condition in place the only control that could be placed on the system would be an additional local status variable that would halt at some indeterminate point in processing an incoming data stream. In other words the problem looks like the receiver saying "go ahead just keep sending data, and I will acknowledge when you tell me to or sometimes when I want to."

The solution is then to adjust our model to the simpler case of the general selective repeat. In this system an acknowledgment is sent after each reception of data. The only transition possible then, after receiving data is thus an acknowledgment or CNTL packet. This is not how XTP behaves however.

In Figure 27 the referenced selective repeat specification is adapted to XTP. It should be noted that at HOST B the mentioned forced acknowledgment is shown.

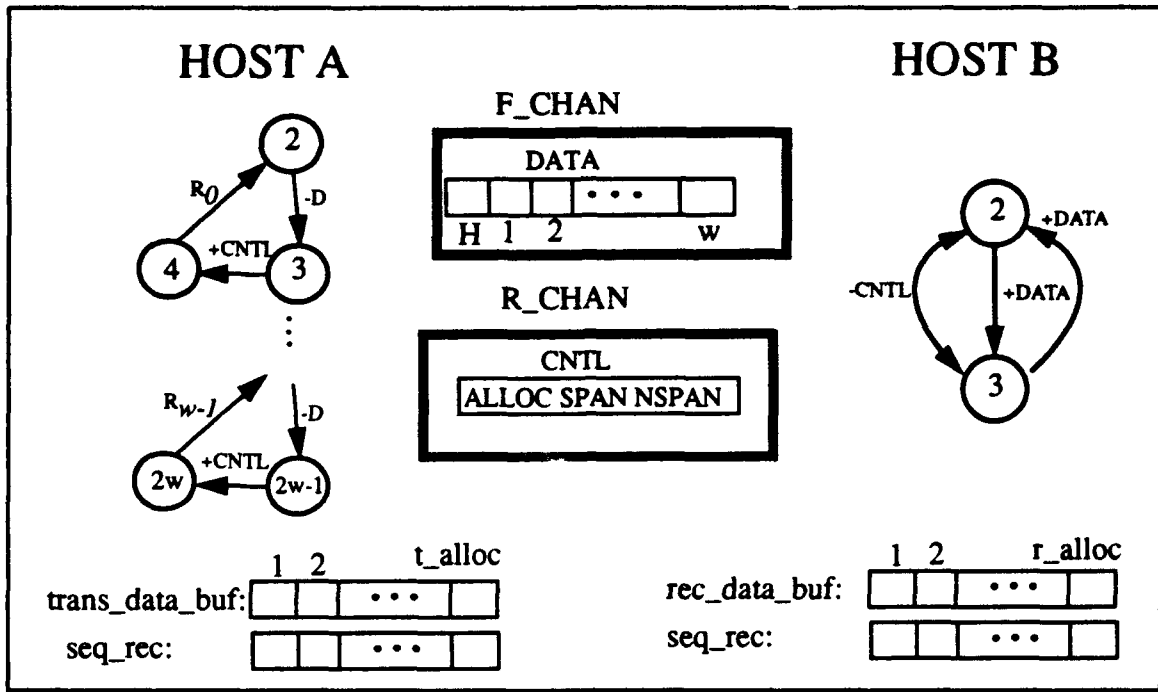


Figure 27: Selective Repeat Specification

- "H" in the diagram indicates the header of the data packet.
- Sequence numbers in F_CHAN range from seq to seq + dlen. That is to from the first sequence number in the packet up to the length indicated.

The initial state of the receiving machine is 2. Any packets that are received with sequence numbers outside of the window are dropped. The initial sequence number is delivered in the header information. If a valid data packet is received then the +DATA is taken, based upon whether the sequence number of the received packet is equal to the next expected sequence number. If the sequence number is not the value expected the packet is stored in the buffer and a CNTL could be sent. If it is the value expected then the window is advanced and this information is transferred to the transmitter through the next CNTL packet.

Further analysis of the XTP protocol was thus not possible due to this problem with the acceptance of the fluid data stream, without a definable control/acknowledgment

policy. The solution to further XTP analysis will require further assumptions for a given implementation.

VI. COMPARISON TO SNR

In this chapter a comparison of the XTP protocol and the SNR protocol highlights many of the problems and proposed solutions to all high-speed data protocols. This comparison investigates only two protocol solutions that have been developed. Network layer functions of XTP will not be discussed since SNR is strictly a transport layer protocol. This comparison is thus limited and will use only the major transport layer functions as the basis of the comparison. Although limited to two protocols, this comparison exposes alternative solutions to the problems facing gigabit networks. It is important in the comparison to focus on what makes each protocol a high-speed protocol. The advantages of each protocol with respect to its ability to transfer data at high data rates will be presented.

A. ADDITIONAL XTP HIGH-SPEED CAPABILITIES

The high-speed capability of XTP is seen in the functional and performance features discussed in this thesis. In this section additional high-speed enhancements to the XTP protocol are briefly discussed for purposes of comparison.

XTP performance gain is achieved through two primary enhancements to the protocol. The XTP header is designed with the header checksum included in the header and the data checksum in the trailer. This allows the transport checksum to be calculated by a hardware checksum unit and to be appended to the data on the fly, thereby eliminating one sequential pass through the data.

The second hardware performance enhancement is the use of fixed length fields. Instead of using variable length headers and trailers, prevalent in many classical protocols, XTP uses fixed length headers and trailers. In a fixed length header and trailer, the positions of each packet control flag is constant, thus promoting efficiency.[STRA92b]

XTP flexibility is enhanced through a conscious effort to separate protocol policy from protocol mechanism. The problem with having a protocol in silicon is that an implementor may feel constrained in the mechanisms the protocol provides to control data

transfer. XTP mechanisms allow for a wide range of communication paradigm implementations. This initially makes the protocol more complex to understand, since there are more decisions an implementor has to make. However, in an actual network design the implementor should know the network design and requirements, thus eliminating any ambiguity in XTP mechanism choice. With the appropriate paradigm implemented the user can design for optimum and efficient high-speed data transfer.

XTP is also capable of protocol performance gains through the application and exploitation of parallel processes. For example, a natural application of parallelism to packet processing would be to separate the receiving, the message parsing and transmitting operations. In [MICH93] a similar idea of off-host processing is presented. In off-host processing the communications architecture provides for the transfer of protocol functions to an attached processor on the system's VME bus. The communications protocol benefits from dedicated processor cycles, specialized hardware and selection and matching of the operating system to the protocol implementation.

B. SUMMARY OF SNR

The SNR high-speed protocol (named after the protocol authors Sabani, Netravali and Roome, AT &T Bell Labs) introduced in [NETR90] is an experimental high-speed data protocol built specifically with the intent of solving data transport problems encountered at transmission rates in the gigabit range. A thorough description and specification of the SNR protocol may be found in [MCAR92] and [TIPI93]. The basic design goal of SNR is to achieve high-speed transport level processing through simplification of the transport protocol, reduction of the processing overhead and utilization of parallel processing. These goals are supported by a simplified protocol design that emphasizes periodic exchange of complete state information, elimination of explicit timers, use of selective repeat for retransmission, data blocking and parallel processing.

1. Periodic Exchange of State Information

SNR allows for the complete exchange of state information between the receiver and the transmitter on a frequent and periodic basis. Although this process increases the total number of control packets transmitted, it simplifies the processing and reduces the need for variable size packet formats. Additionally, the added control packet traffic is negligible in comparison to the very high bandwidth of the fiber. Processing is also simplified by the periodic nature of the control information transmissions. If a control packet is lost, retransmission is unnecessary since the next control packet will update and supersede any previous control information sent.

Another very important consequence of a periodic state exchange is the elimination of explicit timers for error recovery. The timer is not needed when it is known that a control packet will soon update the state machine. In previous protocols this timer would indicate that there was lost data and initiate retransmission of the packets lost.

2. Selective Repeat for Retransmission and Data Blocking

Selective repeat in SNR is implemented with a blocking concept that allows groups of packets to be sent and acknowledged together. The blocking concept reduces the overhead required for maintaining large tables and complex procedures for selective retransmission. The cost of the blocking scheme is seen in the retransmission of some packets in the block that may have been received correctly while other packets in the block required retransmission. The additional bandwidth required for unnecessary retransmissions is expected to be negligible on fiber optic nets where bit error rates are low and bandwidth is high.

3. Parallel Processing

The last design feature implemented in SNR to improve high-speed performance is parallel processing. Parallel processing takes advantage of limited operating system resources and utilizes them in a more efficient manner. SNR is implemented in eight distinct protocol machines that operate almost independently and with only a small amount

of interaction. Thus the protocol lends itself to parallel processing based on the natural separation of eight finite state machines.

C. COMPARISON OF SNR AND XTP

A comparison of the XTP and SNR protocols will highlight respective mechanisms for gaining greater high-speed performance at the transport level. Both protocols are currently undergoing separate testing and research ([McNA92],[MICH93],[NUEF93] and [MCAR92]) to determine what protocol features yield the greatest efficiency and throughput for widespread deployment of high-speed networks. The basis of the comparison and discussion includes just a few areas that are recognized as critical in the development of high-speed protocol features, some of which are summarized in [MCAR92].

1. Increased Performance Through Parallel Processing

In [NUEF93] an architecture for a parallel host interface is presented for increased, stacked communication processing requirements. Parallelism is critical if host systems are going to operate at network speeds. The concept of performance gains through parallelism is not new to either SNR or XTP, although the implementation in each is different. In SNR, parallelism is gained horizontally by separating and assigning distinct transport layer protocols to parallel machines. For example, each of the eight SNR protocol machines could be implemented on a separate processor.

In XTP the parallel machine concept is also seen as a means for increasing performance. In [MICH93] a basic parallel architecture was implemented with XTP and shown to improve high-speed network performance by separating the host operating system and the XTP protocol machine processors. The XTP machine processors, when implemented in silicon, also demonstrate a high degree of parallelism. This is seen in packet pipelining implemented in the highly parallel VLSI design circuits for host interface, buffer control, media access control port and an optional control processor [STRA92b].

A clear advantage of the SNR versus the XTP method of parallel processing is difficult to determine from the literature. However, the author is convinced that some of the XTP protocol complexity could be reduced through the implementation of the protocol in a manner similar to SNR where the protocol itself is subdivided onto distinct machines. This would certainly make analysis easier leaving any performance or efficiency gain to empirical testing.

2. State Information and Lost Control Packet Information

The loss of control packet information is a problem that must be dealt with effectively in all transport protocols, however in a high-speed protocol the problem is exacerbated since even more data can be lost or delayed before communicating machines are re-synchronized. In SNR the problem is solved through an approach best described as "overcome by events." In this method each control packet carries with it the necessary state information to do a complete update with the receiving machine, thus making any previous state information obsolete. To be effective an incoming control packet need only be checked to see if it is the most recent, if so state machines are updated accordingly.

In XTP the loss of control and state information is handled differently. When a control packet is lost a time-out condition will eventually occur at the end of the association that made the status request as the WTIMER (wait timer) expires. This use of timers is both a necessity and inherently troublesome and may lead to poor protocol performance as the state machine waits to time-out. When this occurs a new control packet, CNTL, will be issued. XTP will then implement a synchronizing handshake through the synchronization field, sync, in the forward direction and echo on the return path. Through the synchronization process the states are realigned and a guarantee is made that stale control packets are not acted upon. In this manner XTP maintains accurate and coordinated state information between endpoints exists.

The SNR scheme for handling control packets seems much more reliable than that of XTP. The simplicity of knowing that a control packet will be periodically delivered to

update the state machines of each host would eliminate much of the synchronization processing that is in the XTP protocol. Additionally, the SNR guaranteed control information transmission seems initially more useful in a multicast environment where it is necessary to keep more than two endpoints coordinated.

3. Error Control and Acknowledgment Of Received Data

Acknowledgment of received data and error control is an essential characteristic of transport protocol paradigms that promise end-to-end reliable delivery of data, as in the XTP connection oriented paradigm specified earlier in this paper. SNR and XTP use slightly different means to accomplish the same goal effectively in high-speed environments. In SNR a block acknowledgment is used to acknowledge data blocks and initiate error control. This block acknowledgment indicates the success or failure of transmission of an entire data block. A failure would require the retransmission of the entire block. This may seem inefficient, however since the underlying fiber medium has bit errors rates on the order of 10^{-12} , the number of retransmission is low and therefore not a significant load on the system. SNR uses a reordering buffer to hold and reorder incoming traffic during the amount of time defined by the round trip delay of the connection.

In XTP acknowledgment is provided as part of the control packet (CNTL) sent in response to a status request (SREQ or DREQ) indicator. Usually a range of sequence numbers is acknowledged with each CNTL packet. Unacknowledged data is identified in the form of sequence number gaps, or spans, in the returning control packet. Selective retransmission can then be accomplished. This XTP method uses a 32 bit sequence number that identifies each byte of data, this may seem cumbersome in comparison to block accounting but allows for exact and efficient byte identification and retransmission of errors. Flow control is not significantly effected by this process only precise bytes are required to retransmitted, reducing retransmissions. Thus flow control remains restricted primarily on the amount of buffer space available at the receiver, not the number of outstanding sequence number gaps and retransmissions.

The overhead processing required to assign each byte a sequence number in XTP versus assigning each byte to a block in SNR does not seem to give either protocol an advantage. In either case bookkeeping techniques are required to track large amounts of data being swept through network. The difference between an XTP gap, defined by an ordered pair of bytes is not significantly different from a SNR data block.

4. Connection Management

Connection management is a source of excessive time delay and processing overhead in high-speed networks especially when, due to increased throughput, applications that do not make use of high-speed characteristics are likely to waste time that equivocates to wasted bandwidth.

In SNR, a three way handshaking process takes place before data transfer can begin.

XTP does not use a handshaking process to open an association, although it initially blindly trusts that the connection is made until a CNTL packet is sent later in reply to an SREQ bit used in either flow or error control. In XTP the implementor may decide some form of acknowledgment is necessary but this easily occurs after the connection set up. A failed transmission would not be acknowledged and thus generate a retransmission, it is assumed that the more reliable fiber optic network would make the need for the retransmission less likely.

With reliable fiber optic networks the assumed approach of connection set-up seems that is worth the risk and the preferable method.

5. Implementation

SNR is implemented in software strictly at the transport layer. Whereas XTP is the combination of the network and transport into a single transfer layer with eventual implementation in hardware.

The XTP hardware implementation, without any proprietary restrictions, would be preferable. This way the basic transfer layer protocol could be mass produced cheaply

by many vendors all competing for market share. Changes to the protocol, while undesirable, would eventually come as technical improvements where discovered, requiring cheap silicon replacements for existing XTP chip sets, not something proprietary. From strictly a performance point of view the XTP silicon is preferred. The findings of this chapter are summarized in Table 8.

TABLE 8: XTP and SNR COMPARISON SUMMARY

XTP	Protocol Mechanism	SNR
Parallel processing in XTP tested with off-host protocol processing. Performance increase gained. [MICH93]	Parallel Processing No Advantage	Parallel processing for receiving side, transmitting side and a network interface board. [NETR90]
Implementation for both network and transport layer planned in silicon. Less flexibility if change required, but faster performance possible.	Implementation No Advantage	Protocol implementation planned for hardware. Interface to network layer via a network interface board.
XTP uses 5 different timer mechanisms, although they should perform well in a high speed environment, using this many timers becomes bulky	State Information & Timers Advantage SNR $\Rightarrow$	SNR uses implicit timers, the state machine is frequently and periodically updated. This is a simple control mechanism that should be used in any future high speed transport protocol.
Selective repeat. Uses number of spans and sequence number spans to identify groups of data received, those not received are in error. Sent in CNTL packet.	Error Control and Acknowledgment No Advantage	Selective repeat. Blocks of data designated for retransmission through periodic state updates.
Implicit association establishment possible. Greatly reduces overhead and set-up time. Allows for transaction oriented paradigm.	Connection Management $\Leftarrow$ Advantage XTP	Handshaking process used to initiate connection.
Separate credit and RTIMER mechanism gives an orthogonal rate control process.	Rate Control $\Leftarrow$ Advantage XTP	No unique rate control process
Sequence number on a per byte basis.	Sequence Numbering and Data Blocking Advantage SNR $\Rightarrow$	Sequence numbering done in blocks of bytes. More efficient with less overhead.

TABLE 8: XTP and SNR COMPARISON SUMMARY

XTP	Protocol Mechanism	SNR
New XTP implementation. Potentially very effective in new distributed communication applications	Multi-Cast ⇐ Advantage XTP	Not discussed.

If only one transport protocol could be chosen for deployment into a new high-speed network, based on this thesis and the above analysis the author would choose XTP. This assumes that XTP and SNR were both fully ready for widespread use today and had undergone sufficient testing and analysis to prove their reliability as high-speed transport protocols. Furthermore this choice is based on a technical assessment of SNR and XTP and not on economic or proven performance considerations. It is the flexible capability gained in XTP to operate not only in high-speed networks but with other widely known protocols paradigms that makes it desirable. Certainly, both protocols have features that should be included in the design of the next generation of information services.

VII. CONCLUSION

A. SUMMARY OF RESEARCH

The objective of this thesis has been to present the design, specification and analysis of the Xpress Transfer Protocol, a transfer layer protocol designed for emerging high speed fiber optic networks. It is shown that XTP attempts to overcome the performance and reliability inefficiencies predicted for the transport layer of high speed networks.

However, the success of XTP as a high-speed transfer layer protocol standard at this juncture is yet unknown. Early success of XTP may be inferred from its inclusion as a candidate protocol for the Navy's SAFENET program and for phase two of the Distributed Interactive Simulation (DIS) communication architecture protocol suite, a follow-on effort to DODs SIMNET effort [IST93]. Each of these programs looks to XTP to provide high-speed networking solutions for their respective networks. The determination of which protocol is best, if it can be made, will certainly require further development, investigation, analysis and wide scale testing and acceptance. It is seen in this paper that the XTP approach shows promise as, if not a complete high-speed protocol solution, then as a solution that has many salient points that should be included in future efforts.

As a relatively newly designed protocol XTP has added unique features which provide a high processing speed by simplification of the protocol, introduction of rate control, reduction of the processing overhead, implementation in silicon and the combining of the transport and network layers into a single transfer layer.

To show some of the capability, an XTP connection oriented paradigm was specified with the System of Communicating Machines (SCM) model using the methodology introduced in [LUND91b]. The initial intent was to apply the system state and global state analysis to do a complete XTP specification. However, it was found that the protocol specification is closely connected with the particular paradigm the network designer wishes to implement. Therefore, a straightforward application of global and system state analysis to a complete XTP specification was not possible.

It was seen in the XTP connection oriented paradigm that the protocol could transfer data packets successfully to the receiver tasks without deadlock, unspecified reception, blocking loops or any other kind of logical error using the XTP sliding window selective repeat error correction implementation.

The complexity introduced with expanded XTP mechanism flexibility is minimal when considering the utility of having one protocol that is capable of working in many environments with multiple paradigms. This is not to say that this complexity is desired, in fact it is not. Complexity shows up repeatedly in default parameter selections and in the prediction of packet generation with SREQ or DREQ bits set, that are critical in requiring the receiver to return a CNTL packet. This complexity made specification and analysis difficult. Furthermore, the number of system states remained low, which hid the complexity within local variables. However, the complexity in the protocol is a small price to pay for the projected high-speed performance and reliability that XTP offers.

B. CONTRIBUTIONS OF THIS THESIS

This thesis has the following contributions:

An association establishment specification and analysis for the XTP connection oriented paradigm of the protocol has been done.

The SCM specifications for data transfer, data transfer with flow control, data transfer with rate control and data transfer with error control were accomplished. Analysis was completed for data transfer with error control using system state analysis. Flow control, rate control and error control were specified separately to emphasize their functionality.

A comparison with the SNR high speed protocol was accomplished and identified good qualities of both protocols, certainly a point of departure for further research.

C. FURTHER RESEARCH OPPORTUNITIES

This thesis is a way point for three important areas of research: (1) further research on the XTP protocol performance and reliability, (2) comparison of extant and developmental high-speed protocols and (3) application of the protocol to real high speed networks.

The specification and analysis of XTP is currently limited to a single paradigm. This process could be expanded to other XTP paradigm implementations. The implementations chosen should be considered on a priority basis, with work first being conducted for high throughput implementations. Secondary XTP analysis could then be performed on protocol paradigms designed for low latency, or quick response paradigms, such as the XTP VMTP implementation. Each of these potential research options, in addition to the paradigm presented in this paper, could use the automated analysis methods of [BULB93] to provide a more in-depth analysis of the system and global states. The use of the software tool can provide additional analysis that may uncover protocol logic error that were not discovered in a non-automated approach.

An important question concerned with the XTP protocol is whether the protocol is efficient and reliable enough to provide the high-speed performance which is expected from the lightweight transport protocols. To further answer this question, the protocol needs to be implemented in software and realistic performance tests need to be performed. Some work in this area for XTP has been conducted in [McNA92]. However, additional work in this area is needed to support any high speed protocol standard or attribute preference. An actual transport protocol implementation and subsequent comparison is planned at the Naval Postgraduate School on an FDDI network.

This FDDI network will provide an excellent platform to mount more than one high-speed transport protocol. Experimental performance comparison can then be made between the protocols.

REFERENCES

- [ABSX92] Apple Computer, Bellcore, Sun Microsystems and Xerox, *Network Compatible ATM for Local Network Applications*, Version 1.01, October 19 1992.
- [ARNO93] Arnold, B., "Military Illuminates the Way in Optical Computing," *Military and Aerospace Electronics*, July 1993.
- [BLAC91] Black, Uyless, *OSI, A Model for Computer Communication Standards*, Prentice Hall, Englewood Cliffs, New Jersey, 1991.
- [BREN92] Brendler, J. A., "Tactical Military Communication," *IEEE Communications Magazine*, January 1992.
- [BULB93] Bulbul, Z. B., *A Protocol Validator for the SCM and CFSM Models*, Master's Thesis, Department of Computer Science, Naval Post Graduate School, CA, June 1993.
- [CHER88] Cheriton, D., *VMTP: Versatile Message Transaction Protocol*, Request For Comment 1045, February 1988.
- [CLAR87] Clark, D.D., Lambert, L.L. and Zhang, L., *NETBLT: A Bulk Data Transfer Protocol*, Request For Comment 969, March 1987.
- [CLAR89] Clark, D.D., Jacobson, V., Romkey, J. and Salwen, H., "An Analysis of TCP Processing Overhead," *IEEE Communication Magazine*, June 1989.
- [COME91] Comer, D. E., *Internetworking with TCP/IP, Volume I* Prentice Hall, Englewood Cliffs, New Jersey, 1991.
- [DEMP93] Dempsey, Bert J. and Liebeherr, Jorg, *A New Error Control Scheme for Packetized Voice over High Speed Local Area Networks*, Computer Science Department, University of Virginia, April 1993.
- [ECKE92] Eckerson, W. and Messmer E., "Is OSI Dead?" *Network World*, Volume 9, Number 24, 15 June 1992.
- [HDBK92] Military Handbook, *Survivable Adaptable Fiber Optic Embedded Network (SAFENET)*, September 1992, Naval Ocean Systems Center, MIL-HDBK-220.
- [HIGH93] High, Wayne, *A Formal Protocol Test Procedure For The Survivable Adaptable Fiber Optic Embedded Network (Safenet) (U)*, Master's Thesis, Department of Computer Science, Naval Postgraduate School, CA, March 1993.

- [HAND91] Handel, R., Huber M.N., *Integrated Broadband Networks*, Addison Wesley, Wokingham, England, 1991.
- [IST93] Institute for Simulation and Training, *Rationale, Communication Architecture for Distributed Interactive Simulation (CADIS)*, June 28, 1993.
- [JACO92] Jacobsen, V., Braden, R. and Borman, D., *TCP Extensions for High Performance*, Network Working Group, Request for Comment 1323, May 1992.
- [KANO91] Kano, S., Kitami, K. and Kawarasaki, M., "ISDN Standardization," *Proceedings of the IEEE*, Volume 79, No.2, February 1991.
- [LUND88] Lundy, G.M., *System of Communicating Machines: A Model for Communication Protocols*, Ph.D. Thesis, School of Information and Computer Science, Georgia Institute of Technology, Atlanta Georgia, 1988.
- [LUND91a] Lundy, G.M., "Improving Throughput in the FDDI Token Ring Network," *Proceedings of the Second International Workshop on Protocols for High-Speed Networks*, IFIP, North Holland, 1991.
- [LUND91b] Lundy, G.M. and Miller, R. E., "Specification and Analysis of a Data Transfer Protocol Using System of Communicating Machines," *Distributed Computing*, December 1991.
- [MCAR92] McArthur, R.C., *Design and Specification of a High Speed Transport Protocol*, Master's Thesis, Department of Computer Science, Naval Post Graduate School, CA, 26 March 1992.
- [McNA92] McNabb, James F., and Weaver, Alfred C., *Digitized Voice Distribution Using XTP and FDDI*, Computer Networks Laboratory, University of Virginia, May 1992.
- [MICH93] Michel, Jeffrey R., Waterman, Alexander F., and Weaver, Alfred C., *Performance Evaluation of an Off-Host Communications Architecture*, Computer Networks Laboratory, University of Virginia.
- [MINO91] Minoli, Daniel, *Telecommunications Technology Handbook*, Artech House, Boston, Massachusetts, 1991.
- [MILL90] Miller, R.E. and G.M. Lundy, "Testing Protocol Implementations Based on a Formal Specification," *Proceedings of the Third International Workshop on Protocol Test Systems*, IFIP, North Holland, Oct. 1990.
- [MOLL88] Mollenauer, J. F., "Standards for Metropolitan Area Networks," *IEEE Communications Magazine*, April 1988.

- [MOLL93] Mollenauer, J. F., "The Impact of ATM on Local and Wide Area Networks," *InteNet*, March 1993.
- [NETR90] Netravali, A., Roome, W., and Sabnani, K., "Design and Implementation of a High Speed Transport Protocol," *IEEE Transactions in Communications*, vol.38, #11, Nov 1990.
- [NUEF93] Neufeld, G.W. et al., "Parallel Host Interface for an ATM Network," *IEEE Network*, July 1993.
- [PEI92] Protocol Engines Inc., *Xpress Transfer Protocol, version 3.6*, 1900 State Street, Suite D, Santa Barbara, California 93101 USA, January 11 1992.
- [RAND92] Randall, M. A., *Proof of Fault Coverage for a Formal Protocol Test Procedure*, Master's Thesis Department of Computer Science, Naval Post Graduate School, CA, December 1992.
- [ROTH92] Rothlisberger, M.J., *Automated System State Analysis*, Master's Thesis, Department of Computer Science, Naval Post Graduate School, CA, December 1992.
- [STAL91] Stallings, William, *ISDN and Broadband ISDN*, Macmillan Publishing Company, New York, NY, 1992.
- [STRA92a] Strayer W. D. and Weaver A. C., "Is XTP Suitable for Distributed Real-Time Systems?" *Computer Science Report*, No. 15TR-92-02, January 17 1992.
- [STRA92b] Strayer W. D., Dempsey B. J. and Weaver A. C., *XTP: The Xpress Transfer Protocol*, Addison-Wesley, July 1992.
- [STRE93] Street, Fraser and Weaver, Alfred C., *A Video Mail Distribution System for Networked Personal Computers*, Computer Networks Laboratory, University of Virginia.
- [TIPI93] Tipici, H.A., *Specification and Analysis of a High Speed Transport Protocol*, Master's Thesis, Department of Computer Science, Naval Post Graduate School, CA, March 1993.
- [WEAV92a] Weaver, Alfred C., *XTP: A New Communications Protocol for Factory Automation*, Computer Networks Laboratory, University of Virginia, 1992. Prepared for the IEEE/SICE International Workshop on Emerging Technologies for Factory Automation, North Queensland, Australia.

[WEAV92b] Weaver, Alfred C., *High Speed Communication for Distributed Applications*, Computer Networks Laboratory, University of Virginia, 1992. Prepared for the IEEE International Workshop on Emerging Technologies for Factory Automation, Melbourne, Australia.

INITIAL DISTRIBUTION LIST

- | | | |
|----|--|---|
| 1. | Defense Technical Information Center
Cameron Station
Alexandria, VA 22304-6145 | 2 |
| 2. | Dudley Knox Library
Code 52
Naval Postgraduate School
Monterey, CA 93943-5002 | 2 |
| 3. | Dr. Ted Lewis, Chairman
Computer Science Department
Naval Postgraduate School
Monterey, CA 93943-5000 | 1 |
| 4. | Dr. G. M. Lundy
Code CS/Ln
Assistant Professor, Computer Science Department
Naval Postgraduate School
Monterey, CA 93943-5000 | 3 |
| 5. | Professor L. Stevens
Code CS/St
Associate Professor, Computer Science Department
Naval Postgraduate School
Monterey, CA 93943-5000 | 1 |
| 6. | COMMANDER
Information Systems Software Command
Systems Management Directorate
Attn.: CPT David Sacha
Ft. Belvoir, VA 22306 | 6 |