1994-03

# High speed transport protocols: an attempt to find the best solution

Lazaris, Konstantinos A.

Monterey, California. Naval Postgraduate School

# NAVAL POSTGRADUATE SCHOOL
## Monterey, California



# THESIS

**HIGH SPEED TRANSPORT PROTOCOLS:
AN ATTEMPT TO FIND THE BEST SOLUTION**

by

Konstantinos A. Lazaris

March, 1994

Thesis Advisor: G. M. Lundy

# NAVAL POSTGRADUATE SCHOOL
## Monterey, California

August 11, 1994

Thesis Processing
Code 82, Sandra Day

Defense Technical Information Center
Cameron Station
Alexandria, VA 22304-6145

DTIC AD Number: A280989

Please replace Konstantinos A. Lazaris, March 1994 thesis with attached. Thank you.

Sandra Day
Sandra Day

# REPORT DOCUMENTATION PAGE

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED | 1b. RESTRICTIVE MARKINGS |
| 2a. SECURITY CLASSIFICATION AUTHORITY | 3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S) | 5. MONITORING ORGANIZATION REPORT NUMBER(S) |
| 6a. NAME OF PERFORMING ORGANIZATION: Computer Science Dept. Naval Postgraduate School; 6b. OFFICE SYMBOL (if applicable): CS | 7a. NAME OF MONITORING ORGANIZATION: Naval Postgraduate School |
| 6c. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000 | 7b. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000 |
| 8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (if applicable) | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER |
| 8c. ADDRESS (City, State, and ZIP Code) | 10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO |

11. TITLE (Include Security Classification)
**HIGH SPEED TRANSPORT PROTOCOLS: AN ATTEMPT TO FIND THE BEST SOLUTION (U)**

12. PERSONAL AUTHOR(S)
Lazaris, Konstantinos A.

| 13a. TYPE OF REPORT | 13b. TIME COVERED | 14. DATE OF REPORT (Year, Month, Day) | 15. PAGE COUNT |
|---|---|---|---|
| Master's Thesis | FROM 07/93 TO 03/94 | March 1994 | 94 |

16. SUPPLEMENTARY NOTATION 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.

| 17. COSATI CODES | | | 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | High Speed Transport Protocol, Transmission Control Protocol/ Internet Protocol, Xpress Transfer Protocol, AT&T Lightweight Transport Protocol |
| | | | |
| | | | |

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The development and advances in fiber optic technology are leading to major changes in modern telecommunications systems. In short, the transmission of data through optical fiber has become so fast that the computers which the fibers connect have become a bottleneck. The transport layer protocol, which is the software interface between the network and the computer, is one of the most important sources of this bottleneck. The purpose of this thesis is to investigate several "high-speed" transport protocols, evaluate them and attempt to determine which transport protocol or combination of transport protocols is optimal for high speed networks of the future. The approach is to first study the requirements of transport protocols for high speed networks. Then the properties of several specific transport protocols are studied with these requirements in mind. A detailed analysis of the strengths and shortcomings of TCP/IP, XTP, and SNR are presented. TCP/IP, which is in wide use today, was designed when transmission rates were much slower and error rates were much higher than today. XTP and SNR are two new experimental transport layer protocols which have been recently designed with high speed networks in mind. The primary contribution of this thesis is an evaluation of the requirements of future transport protocols. In short, TCP/IP in its present form is simply not adequate; it must change and adapt, or replaced by a new transport protocol like XTP, or SNR.

| 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT | 21. ABSTRACT SECURITY CLASSIFICATION | |
|---|---|---|
| [X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT. [ ] DTIC USERS | UNCLASSIFIED | |
| 22a. NAME OF RESPONSIBLE INDIVIDUAL: G. M. Lundy | 22b. TELEPHONE (Include Area Code): (408) 646-2449 | 22c. OFFICE SYMBOL: CS/ LN |

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE: UNCLASSIFIED

Approved for public release; distribution is unlimited

# HIGH SPEED TRANSPORT PROTOCOLS: AN ATTEMPT TO FIND THE BEST SOLUTION

by
Konstantinos A. Lazaris
Lieutenant, Hellenic Navy
B. S., Hellenic Naval Academy, 1984

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN COMPUTER SCIENCE**

from the

**NAVAL POSTGRADUATE SCHOOL**
March 1994

Author: [redacted]
Konstantinos A. Lazaris

Approved By: [redacted]
G. M. Lundy, Thesis Advisor

[redacted]
L. Stevens, Second Reader

[redacted]
T. Lewis, Chairman,
Department of Computer Science

## ABSTRACT

The development and advances in fiber optic technology are leading to major changes in modern telecommunication systems. In short, the transmission of data through optical fiber has become so fast that the computers which the fibers connect have become a bottleneck. The transport layer protocol, which is the software interface between the network and the computer, is one of the most important sources of this bottleneck.

The purpose of this thesis is to investigate several "high-speed" transport protocols, evaluate them and attempt to determine which transport protocol or combination of transport protocols is optimal for high speed networks of the future.

The approach is to first study the requirements of transport protocols for high speed networks. Then the properties of several specific transport protocols are studied with these requirements in mind. A detailed analysis of the strengths and shortcomings of TCP/IP, XTP, and SNR are presented. TCP/IP, which is in wide use today, was designed when transmission rates were much slower and error rates were much higher than today. XTP and SNR are two new experimental transport layer protocols which have been recently designed with high speed networks in mind.

The primary contribution of this thesis is an evaluation of the requirements of future transport protocols. In short, TCP/IP in its present form is simply not adequate; it must change and adapt, or replaced by a new transport protocol like XTP, or SNR.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# ACKNOWLEDGMENTS

This thesis is dedicated to my beloved wife Vasso, who patiently stood by me during the two hard years of studying at the Naval Postgraduate School. Without her support everything would be even more difficult. I would like to thank my brother Christodoulos, who influenced my decision in selecting Computer Science as my field of study.

I would also like to thank Dr. G.M. Lundy for accepting me as a thesis student, helping me to successfully finish my thesis, and teaching me everything I know in the field of computer communications and networks.

Finally, I would like to express my gratitude to the Hellenic Navy for giving me the opportunity to further my education and professional ability and broaden my social horizons.

# I. INTRODUCTION

## A. CURRENT SITUATION WITH TRANSPORT PROTOCOLS

Optical fiber is employed for a large number of applications including long-distance networks. When first introduced in the early 1980s, optical fiber systems operating at 45 Mbps were considered high speed; currently 500 Mbps systems are common in the public switched network [MINO91]. A number of manufacturers have already introduced systems operating at 1.5 Gbps, or more. This technology is still under development and it will continue to achieve higher data rates.

Prior to the introduction of fiber optic transmission media, the limiting factor in throughput was generally the transmission media itself. The transport protocol at the receiving entity could easily process incoming packets at a rate which exceeded the throughput capacity of the media. The transport protocols that are currently in widespread use, were designed for networks operating at lower data rates and higher error rates. The most widely known transport protocols are TP4 and TCP/IP. TP4 is a term given to the ISO[1] protocol suite that closely resembles TCP. It provides reliable stream delivery service using basically the same techniques of positive acknowledgment and retransmission. It handles the problems of lost data, flow control, window management, and data that arrives out of sequence [COME91]. The TCP/IP architecture, described in chapter III, acquired its current form in the late 1970s. It consists of the Transmission Control Protocol (TCP), which was designed to work over the unreliable network data

1. International Organization for Standardization (ISO). It is an international agency for the development of standards on a wide range of subjects. Its purpose is to promote the development of standardization and related activities to facilitate international exchange of goods and services and to develop cooperation in the sphere of intellectual, scientific, technological, and economic activity [STAL88].

delivery service known as the Internet Protocol (IP). In effect, TCP takes care of the integrity and IP moves the data.

Many protocol designers, thinking of higher data rates and lower error rates, have designed new protocols to provide more and better services than are currently available. The resulting protocols have been termed *lightweight* or *high speed transport protocols*, such as XTP described in Chapter IV, and SNR described in Chapter V.

Other experimental lightweight transport protocols are VMTP and NETBLT. The Versatile Message Transaction Protocol (VMTP), is designed to support remote procedure call and general transaction-oriented communication. It handles the error detection, selective retransmission, duplicate suppression and, optionally, security required for transport end-to-end reliability. It provides multicast, real-time datagrams, and in general is designed with the next generation of communication systems in mind. The protocol is designed to operate on top of a simple unreliable datagram service, such as is provided by IP [CHER88].

The Network Block Transfer Protocol (NETBLT) is a transport layer protocol designed for efficient transfers of large amounts of data. It provides a transfer that is reliable and flow controlled, and is structured to provide maximum throughput over a wide variety of networks. It controls the rate at which data is sent to allow a steady high speed flow. NETBLT's design was driven by several observations about flow control and error recovery algorithms in the conventional transport protocols. This high speed transport protocol is also designed to operate on top of any datagram protocol similar in function to IP [CLAZ87].

Other researchers try to modify and extend the existing protocols [JACO88], [JACO90], [JACO92], in order to speed them up. They feel that the experience gained with the current protocols can be effectively used to improve them. Many have already

demonstrated that improvements can lead to better performance. Especially concerning TCP/IP a considerable amount of work has been done, and most of the written information including its architecture, protocols, and history can be found in RFC's[1].

In brief, the transmission of data through optical fiber has become so fast that the computers which the fibers connect have become a bottleneck. The transport layer protocol, which is the software interface between the network and the computer, is one of the most important sources of this bottleneck. The problem is which of the two approaches should be adapted, in order to achieve the highest performance?

## B. OBJECTIVES OF THIS THESIS

In this thesis, the design considerations for lightweight protocols will be investigated, and the limitations of today's transport protocols will be identified. Changing or extending the current implemented transport protocols is the big question. A detailed analysis of the shortcomings and strengths of each protocol is the major contribution of this thesis. Protocols need to be able to control the larger amounts of data in flight and provide new services like performance guarantees.

This thesis will also include an introduction of the general transport protocol functions, a review of the Transmission Control Protocol/Internet Protocol (TCP/IP), as well as a description of the main services and features of two experimental high speed transport protocols, the Xpress transfer protocol (XTP), and the AT&T transport protocol (SNR).

---

1. Request For Comments (RFC). The name of a series of notes that contain surveys, measurements, ideas, techniques, and observations, as well as proposed and accepted protocol standards. They are available on-line from the Network Information Center [COME91].

## C. ORGANIZATION OF THE THESIS

This thesis is a study of the functions and capabilities of transport protocols. The comparison between a current implemented transport protocol and two new lightweight transport protocols is based in the different services and features that each one provides. A detailed performance comparison of these protocols is a topic for further research since a large variety of operating environments and performance benchmarks exist. Instead some basic examples of transmitted messages in a long-distance network operating at 1 Gbps will be given.

The rest of the thesis is organized into six chapters. Chapter II discusses the main transport protocol functions. Chapter III describes the TCP/IP transport protocol and its features. Chapters IV and V introduce the XTP and SNR transport protocols respectively, and their most important services and features. Chapter VI includes the comparison of the three transport protocols, discussing separately each specific service. Strengths and weaknesses of each protocol will be mentioned. Three basic examples of message transmission will also be presented. Finally, Chapter VII provides the conclusion of the thesis.

# II. TRANSPORT PROTOCOLS

In this chapter we review the purpose of the transport protocols, and discuss the primary transport protocol services. Several examples will be given before the summary.

## A. PURPOSE OF THE TRANSPORT PROTOCOL

A protocol is used for communication between entities in different systems. It is defined as a set of rules controlling the exchange of data between two application programs. Such communication is often called *end-to-end*, and the application programs *endpoints*.

It is essential to achieve a high degree of cooperation between two computers. Instead of a single protocol for performing communications, there is a structured set of protocols that implements the communications function. That structure is referred to as a communication architecture. These elements of the structured set of protocols are layered or hierarchically organized so that there is a continuous flow of responsibility.

The transport protocol (Open Systems Interconnection model, layer four) is the most important concept of a computer communications architecture. Its purpose is to provide the basic end-to-end service of transferring data reliably between endpoints. It ensures that data units are delivered error-free, in sequence, with no losses or duplications. The transport protocol may also be concerned with optimizing the use of network services and providing a requested quality of service [STAL88]. The size and complexity of a transport protocol depends on the type of service it can get from the lower network layer.

Transport protocols are designed to provide a universal interconnection among

machines independent of the particular networks to which they attach. They treat all networks equally. A local area network (LAN) like an Ethernet, a wide area network (WAN), a point-to-point link between two machines or an interconnection through gateways like internet, each count as one network (Figure 1).

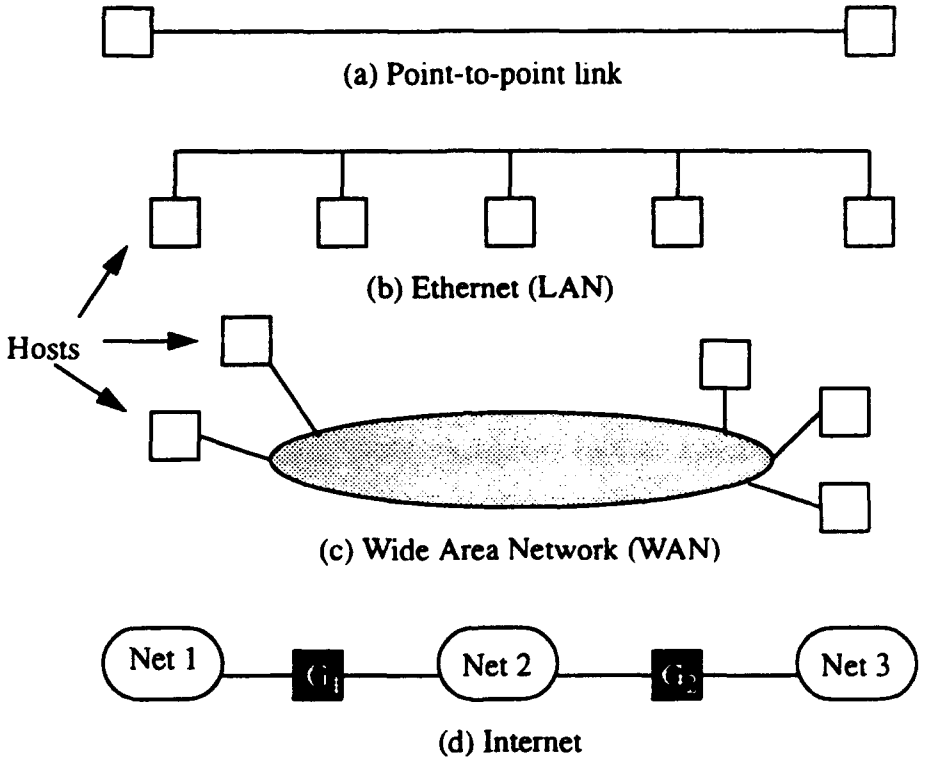



**Figure 1:** Varieties of Systems/Networks Connected by Transport Protocols

## B. FUNCTIONS OF THE TRANSPORT PROTOCOL

In the OSI model, the fourth layer contains the transport protocol TP4. In TCP/IP, the transport protocol is TCP. Transport protocols are responsible for the following main functions:

- Addressing,

- Segmentation and reassembly,
- Connection management (connection establishment, data transfer, connection termination),
- Flow and rate control, and
- Error control (error detection and recovery).

In this section we review some of these transport protocols services. For more information see [COME91] and [STAL88].

### 1. Connection Management

Connection oriented and connectionless are the two basic forms of operation provided by a transport protocol.

Connection oriented service involves a connection establishment phase, a data transfer phase, and a connection termination phase. A logical connection is set up between end systems prior to exchanging data. These phases define the sequence of events ensuring successful data transmission. Connection oriented allows flow control and transparent error handling. The packets follow the same route and arrive in correct sequential order. Traditional telecommunication services are connection oriented.

Connectionless service treats each packet independently from all others. A sequence of packets sent from one end point to another may follow different paths. Connection establishment activities are not required because each data unit is independent of the previous or subsequent one. Each unit of data must contain at least the addressing information and the data. Transmission delivery is uncertain, due to the possibility of error. The main advantage of connectionless service is that it is more robust [STAL88].

Connectionless service is ideal for short messages, because it avoids the overhead of establishing and maintaining the communication route. It is also good for

transient processes (in military, aviation, and meteorological systems, in which frequent and abrupt dissociation often occurs) [MINO91]. For file transfers of the order of megabytes, the connection oriented mode seems to be the appropriate solution, since the transport protocol will not have to manage the sequencing of the different packets. In addition, for large files, the overhead of establishing and maintaining the connection is negligible.

During connection establishment a number of parameters for the transfer of data must be negotiated, as for example buffer space availability at the receiver, flow control algorithms and round trip delay. Connection termination can be either abrupt or graceful. With an abrupt termination, data in transit may be lost. A graceful termination prevents either side from shutting down until all data have been delivered. The nature of the connection will determine the optimal method for controlling the transfer of data, and it is the transport protocol which must establish the connection environment according to the requirements of the connection.

### 2. Flow Control

Flow control assures that a transmitting station does not overwhelm a receiving station by sending data faster than they can be processed. The receiver typically allocates a data buffer with some maximum length. After processing received data, it clears the buffer in order to receive more data. In the absence of flow control, the receiver's buffer may overflow while it is processing old data.

The transmitter divides larger messages into smaller pieces called frames or fragments according the network's maximum transfer unit or MTU. A length field, a sequence number, and a checksum for error detection are usually included to allow the receiver to determine whether the packet arrived with error. When the packet arrives at the ultimate destination, the receiver process the packets removing the length fields, checking

for errors, and determining the correct sequence. Flow control forces the transmitter to limit transmission of data packets based upon the current state of the receiver's buffer.

The simplest form of flow control is known as stop-and-wait flow control. In this method the receiver indicates if it is capable of receiving another frame by sending an acknowledgment to the last received frame. The source must wait until it receives the acknowledgment before sending the next frame. The receiver has the possibility to stop the flow of data by simply holding the acknowledgment. If an error is detected by the receiver, the frame is discarded and a negative acknowledgment is sent causing the source to retransmit the frame. If no recognizable acknowledgment is received during a timeout period, then the frame is retransmitted. The principal advantage of stop-and-wait is its simplicity. Its main disadvantage is that this is an inefficient protocol.

The most common method of flow control is the sliding window protocol. A simple stop-and-wait protocol wastes a substantial amount of network bandwidth because it must delay sending a new packet until it receives an acknowledgment for the previous packet [COME91]. The sliding window is a more complex form of positive acknowledgment and retransmission using network bandwidth better. The sender transmits multiple packets before waiting for an acknowledgment (Figure 2). A small window is placed on the sequence, and all packets that lie inside are transmitted. The window size is limited to a small number, and usually negotiated during the connection establishment procedure.

A sliding window protocol remembers which packets have been acknowledged and keeps a separate timer for each unacknowledged packet. If a packet is lost, the timer expires and the sender retransmits that packet. The performance of sliding window protocol depends on the window size and the speed at which the network accepts packets. A well tuned sliding window keeps the network busy, and obtains higher throughput than

a simple stop-and-wait protocol.

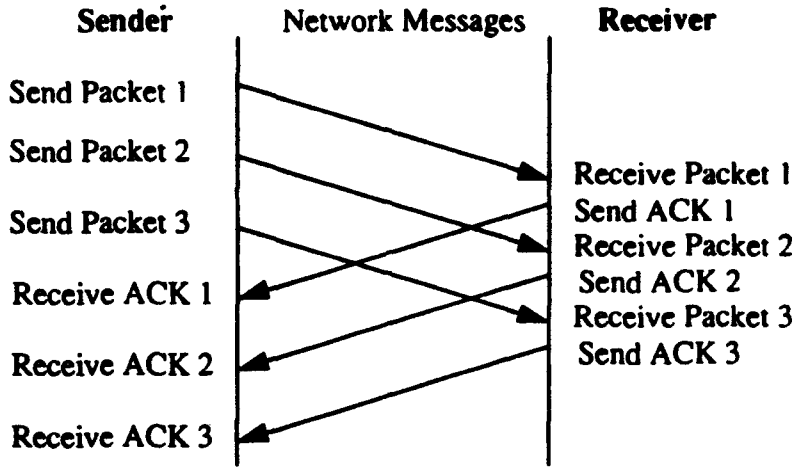



**Figure 2:** Sliding Window Protocol with Three Packets Transmitted.

### 3. Error Detection and Recovery

Messages may be lost or corrupted. In these cases, data lost or delivered with errors is recovered through retransmission from the transmitter. The two most widely known schemes for triggering retransmissions are a positive acknowledgment with retransmission (PAR) and automatic repeat request (ARQ).

Under PAR, if the timer expires before an acknowledgment arrives at the transmitter, the packet is retransmitted. The receiver acknowledges only when the packets have been without error and in the correct sequence. A reliability problem arises when a duplicate packet arrives. In this case, reliable protocols detect duplicate packets by assigning a sequence number and requiring the receiver to remember which sequence number has been received. Positive acknowledgment protocols send sequence number in acknowledgments, so the receiver can correctly associate them with packets. Under ARQ,

packets are retransmitted according to the information contained in the acknowledgment. The sender automatically repeats the request if it does not receive an answer.

Two methods are used to recover from lost packets or packets received with errors:

(1) "go-back-n," and

(2) selective retransmission.

The "go-back-n" method retransmits the packets and all subsequent packets (Figure 3). The receiver acknowledges a correctly received packet by indicating the next sequence number expected. The main disadvantage is that many packets that were successfully received may be retransmitted.

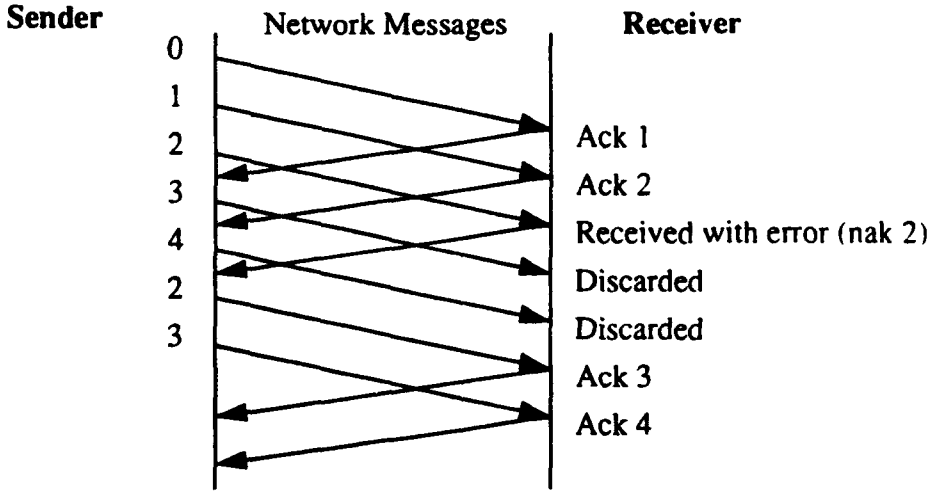



**Figure 3:** "Go-back-N" Method.

The selective retransmission method retransmits only the lost packets or the packets received with errors (Figure 4). This would appear to be more efficient than the

"go-back-n" method, since it minimizes the amount of retransmission. It is desirable in long delay or high bandwidth connections in which a great amount of data may be in transit. The disadvantage is that it increases the processing requirements of the receiver.

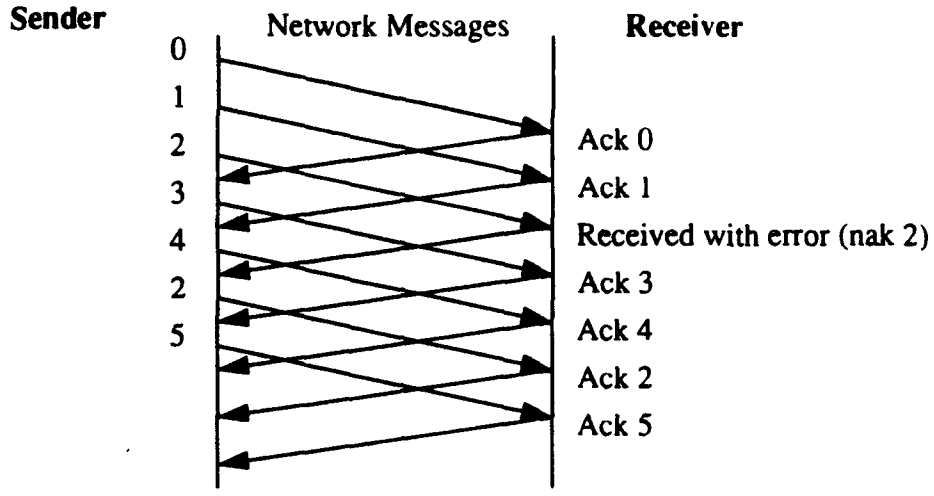



**Figure 4:** Selective Retransmission.

## C. EXAMPLES OF TRANSPORT PROTOCOLS

The two most widely known standards for reliable transport layer protocols are the Transmission Control Protocol (TCP), and the ISO Transport Protocol class 4 (TP4). These protocols offer a set of algorithms and protocol techniques whose power and limitations have been widely studied. Behind these conventional transport protocols, several lightweight transport protocols have been designed in order to improve the performance. Among these high speed transport protocols are the Xpress Transfer Protocol (XTP), the SNR Protocol, the Versatile Message Transaction Protocol (VMTP), the Network Block Transfer Protocol (NETBLT), the Delta-t protocol, and the Datakit protocol.

## D. SUMMARY

At the transport layer, the amount and complexity of protocol processing depends upon the data transfer service provided, and the nature of the underlying network layer service. The reliable and transparent transfer of data between endpoints, as well as the end-to-end error recovery and flow control are the basic functions of the transport protocols. In this chapter, the main services were presented, and several examples of current implemented and new designed transport protocols were given. In the next chapters three of these protocols will be presented in detail, starting with TCP/IP.

# III. TRANSMISSION CONTROL PROTOCOL/ INTERNET PROTOCOL (TCP/IP)

In this chapter the TCP/IP protocol suite is discussed. This suite is a set of several protocols which is named after the two most important. The main functions of the Internet Protocol (IP) and transport protocol (TCP) are primarily discussed.

## A. TCP/IP PROTOCOL SUITE

The TCP/IP protocol suite, developed under the direction of DARPA[1], is designed to allow cooperating computers to share resources across a variety of networks. Many protocols are included to the TCP/IP protocol suite. Some of them provide "low-level" functions needed for many applications. TCP, UDP, and ICMP lie above the Internet Protocol (IP), and below the application layer (Figure 5, and TABLE 1). Protocols, like TCP and IP, give the formulas for passing messages, specify the details of message formats, and describe how to handle error conditions. UDP uses the Internet Protocol to deliver packets and ICMP handles error and control messages. Some other protocols perform specific tasks, as IGMP and ARP (TABLE 1). Later, in this section, each of these protocols will be explained in more detail. In addition to the basic transport-level services, the TCP/IP protocols include standards for many common applications including electronic mail, file transfer, and remote login.

---

1. Defense Advanced Research Projects Agency (DARPA). Formerly called ARPA. The government agency that funded research and experimentation with the ARPANET, and later, the connected Internet [COME91].

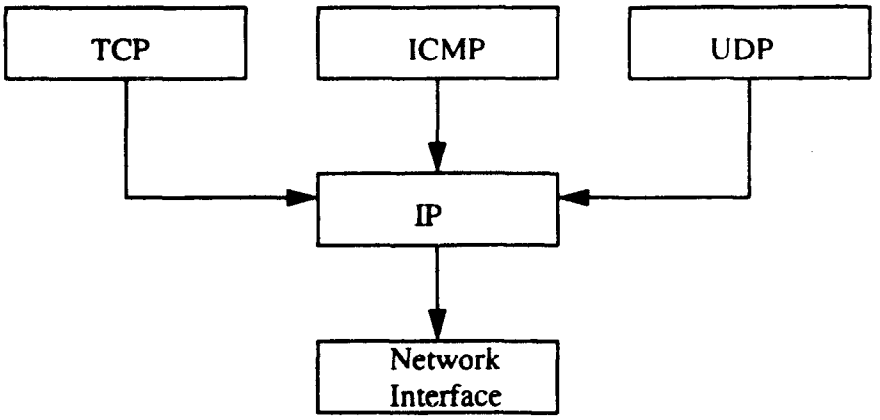



**Figure 5:** TCP/IP Protocol Suite

**TABLE 1: PROTOCOLS OF TCP/IP**

| Abbreviation | Name of Protocol |
|---|---|
| IP | Internet Protocol |
| TCP | Transmission Control Protocol |
| UDP | User Datagram Protocol |
| ICMP | Internet Control Message Protocol |
| IGMP | Internet Group Management Protocol |
| ARP | Address Resolution Protocol |

### 1. Internet Protocol (IP)

Internet Protocol is one of the two major protocols included in the TCP/IP protocol suite. It is an unreliable, best effort, connectionless delivery mechanism of packets across a network. Packets may be lost, duplicated, delayed or delivered out of order. Such conditions will not be detected. Each packet is treated independently from all

others. IP provides three important definitions. Specifies the exact format of all data which passes across an internet, performs the routing function choosing a path over which data will be sent, and includes a set of rules which characterize how hosts and gateways should process packets, how and when error messages should be generated, and the conditions under which packets can be discarded [COME91].

An Internet datagram, or sometimes called an IP datagram, is the basic unit transfer. It is divided into header and data areas. The header contains the source and destination addresses (Figure 6). The first 4 bits field *Vers* contains the version of the IP protocol that is used to create the datagram. All IP software is required to check this field. The current protocol is version 4. The *Hlen* field, also 4 bits, gives the datagram header length, measured in 32 bits words. The 8 bit *Service Type* field specifies how the datagram should be handled, and is broken down into five subfields. Three precedence bits specify the priority of a datagram, with values from zero through seven allowing senders to indicate the importance of each datagram. Three bits specify the type of transport the datagram desires. Low delay, high throughput, and high reliability are the three available options. The last 2 bits of *Service Type* are not used.

The *Total Length* field gives the length of the IP datagram measured in octets including octets in the header and data. Field *Identification* contains a unique integer that identifies the datagram. Its purpose is to allow the destination to know which fragments belong to which datagrams. In order to have a unique identification field for each datagram a global counter in memory is incremented each time a new datagram is created. The low-order 2 bits of the 3-bit *Flags* field control fragmentation. The first control bit specifies whether the datagram may be fragmented. It is called the *"do not fragment"* bit because setting it to 1 specifies that the datagram should not be fragmented. The low order bit in the *Flags* specifies whether the fragment contains data from the middle of the

original datagram or from the end. It is called the *"more fragments"* bit. This bit helps the destination to know if the specific fragment is the last part of the datagram. Field *Fragment Offset* specifies the offset in the original datagram of the data (in units of 8 octets) in order to be easy to reassemble the datagram in the destination.

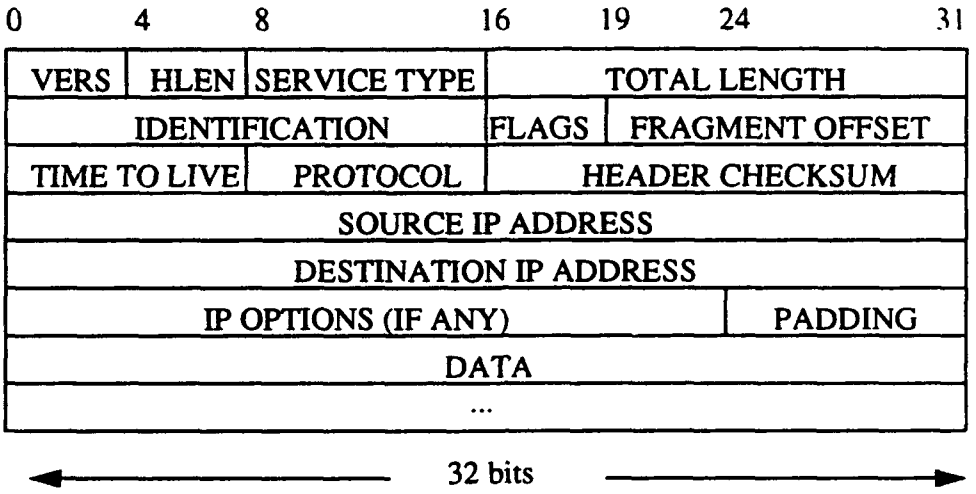



**Figure 6:** Format of an Internet Datagram, the Basic Unit of Transfer.

Field *Time To Live* specifies how long, in seconds, the datagram is allowed to remain in the Internet system, field *Protocol* specifies which high-level protocol was used to create the message being carried in the data area of a datagram, and *Header Checksum* ensures integrity of header values only in the IP header. *Source* and *Destination IP Address* contain the 32 bits IP addresses of the datagram's sender and intended receiver. The *IP Options* field is not required in every datagram. The length of this field varies depending on which options are selected. Record route, source route and timestamp are some of the possible options. For more details see [COME91]. The field labeled *Padding* represents bits containing zero that may be needed to ensure the datagram header extends

to an exact multiple of 32 bits. Finally *Data* shows the beginning of the data area of the datagram.

To make internet transportation efficient each datagram travels in a distinct frame. The idea of carrying one datagram in one frame is called *encapsulation*. TCP, UDP and ICMP messages are always encapsulated in IP datagrams (Figure 5).

### 2. Transmission Control Protocol (TCP)

The Transmission Control Protocol is the second major protocol of the TCP/IP protocol suite. It fits into a layered protocol architecture just above a basic Internet Protocol (Figure 7). TCP is a connection-oriented sliding window protocol which uses byte based sequence numbers, positive acknowledgments and timer based retransmission. Reliable interprocess communication between pairs of processes in host computers is also provided [ISIC81].

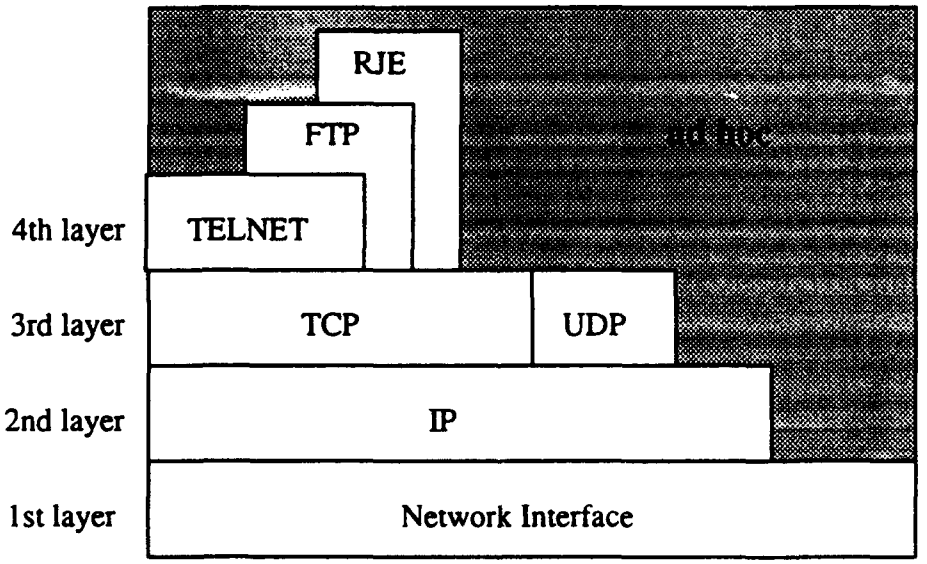



**Figure 7**: TCP/IP Protocol Architecture

TCP specifies the format of the data and acknowledgments that two computers exchange to achieve a reliable transfer, as well as the procedures the computers use to ensure that the data arrives correctly. The unit of transfer between the TCP software on two machines is called a *segment*. Segments are enclosed in internet datagrams, and exchanged to establish connections, to transfer data, to send acknowledgments, to advertise window sizes and to close connections.

In the next section the format of a TCP segment is presented. Later, in this chapter, the main functions of TCP, as connection management, flow and error control will be described.

### 3. User Datagram Protocol (UDP)

The User Datagram Protocol (UDP) is an alternative to TCP. It is designed for applications in which sequences of datagrams are not needed. Each message has a UDP header on the front of the data. Then UDP sends the data to IP which adds the IP header putting UDP's protocol number in the protocol field instead of TCP's protocol number. UDP provides the same unreliable, connectionless datagram delivery semantics as IP. It does not use acknowledgments to make sure messages arrive, does not order incoming messages and does not provide feedback to control the rate at which information flows between the machines. Applications requiring ordered reliable delivery of streams of data should use the Transmission Control Protocol [POST80].

### 4. Internet Control Message Protocol (ICMP)

The Internet Control Message Protocol (ICMP) allows gateways to send error or control messages to other gateways or hosts. ICMP provides communication between the Internet Protocol software on one machine and the Internet Protocol software on another. ICMP only reports error conditions to the original source. The source may not be able to

determine which gateway caused the problem. Host and network administrators will locate and correct it. An ICMP message travels across the internet in the data portion of an IP datagram. Datagrams carrying ICMP messages are routed exactly like datagrams carrying information for users.

### 5. Internet Group Management Protocol (IGMP)

To participate in IP multicast on a local network, a host must have software that allows it to send and receive multicast datagrams. Local gateways contact other multicast gateways in order to establish the appropriate routes. Multicast gateways must use Internet Group Management Protocol (IGMP) to communicate group membership information. IGMP uses IP datagrams to carry messages, like ICMP does.

### 6. Address Resolution Protocol (ARP)

The Address Resolution Protocol (ARP) provides a mechanism to map from a high-level IP address to a low-level physical hardware address. The address resolution used by TCP/IP protocols has been solved without maintaining a centralized database. Another advantage of this protocol is that new machines can be added to the network without changing data or recompiling code. Unlike most protocols, the data in ARP packets does not have a fixed format headers.

## B. TCP SEGMENT FORMAT

A segment is the unit of transfer between the TCP software on two endpoints. Segments are used to establish connections, to transfer data, to send acknowledgments, to specify window sizes, and to close connections [COME91].

The TCP header and the data are the two main parts of each segment (Figure 8). TCP header carries the identification and control information. Fields *Source Port* and

*Destination Port* identify the application programs at the ends of the connection (port numbers). The *Sequence Number* field identifies the position of the first data octet in the transport protocol data unit, since the *Acknowledgment Number* field identifies the next octet that the receiver expects[1]. The length of the TCP header, measured in multiples of 32 bits is contained in *Hlen*. This field is required since the *Options* field varies in length. A 6-bit field *Reserved* is reserved for future use. The field *Code Bits* is used to determine the purpose and the contents of the segment (TABLE 2). The current size of the buffer in octets is advertised in the *Window* field. The *Urgent Pointer* field specifies the position in the window where urgent data ends. This allows the receiver to know how much urgent data are coming. A variable length field *Options* is included depending on which options have been selected. *Padding* represents bits containing zero that may be needed to ensure the datagram header extends to an exact multiple of 32 bits. Finally, a *Checksum* field in the header contains a 16-bit integer checksum used to verify the integrity of the data, as well as the TCP header.

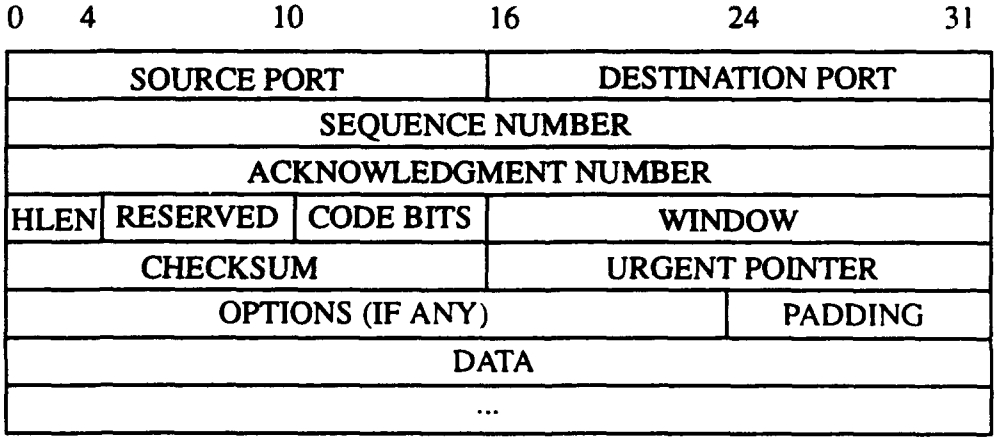



**Figure 8:** The Format of a TCP Segment with a Header Followed by Data.

1. The *Sequence Number* refers to the stream flowing in the same direction as the segment, while the *Acknowledgment Number* refers to the stream flowing in the opposite direction as the segment.

A segment has a maximum size which is negotiated between the endpoints during the connection establishment. The *Options* field allows TCP software at the destination to specify the maximum segment size that it is willing to receive. It is especially important for computers connected by high speed local area networks to choose a maximum segment size that fills packets, otherwise they will not make good use of bandwidth [COME91]. If endpoints do not lie on the same network, the current specification suggests using a maximum segment size of 536 octets, which is the default size of an IP datagram (570) minus the standard size of IP and TCP headers (40). In an internet environment, choosing a good maximum size can be difficult because performance can be poor for either extremely large segment sizes or extremely small sizes.

**TABLE 2: BITS OF CODE FIELD IN THE TCP HEADER.**

| Bit (left to right) | Meaning if bit is set to 1 |
|---|---|
| URG | Urgent pointer field is valid |
| ACK | Acknowledgment field is valid |
| PSH | This segment requests a push |
| RST | Reset the connection |
| SYN | Synchronize sequence numbers |
| FIN | Sender has reached end of its byte stream |

TCP segments travel encapsulated in IP datagrams which are encapsulated in network frames. This means that in addition to the data, each segment has at least 40 octets of TCP and IP headers.

## C. CONNECTION MANAGEMENT

Connection establishment and termination phases of a data transfer can be considered independent of the actual movement of data. These procedures ensure that no false connections are created, that each packet is mapped to its proper destination, and that a graceful close will occur after all data has been received correctly.

For long lived connections the cost of the opening and closing handshakes is reasonable, but handshaking schemes are expensive for short lived connections. The reason is that the propagation delay is added to the transmission time of the message. For long lived connections the additional time will be only a very small percentage of the total connection time. For short lived connections, the propagation delay may be the greatest percentage of the total time required.

### 1. Connection Establishment

To establish a connection TCP uses a three way handshake (Figure 9).

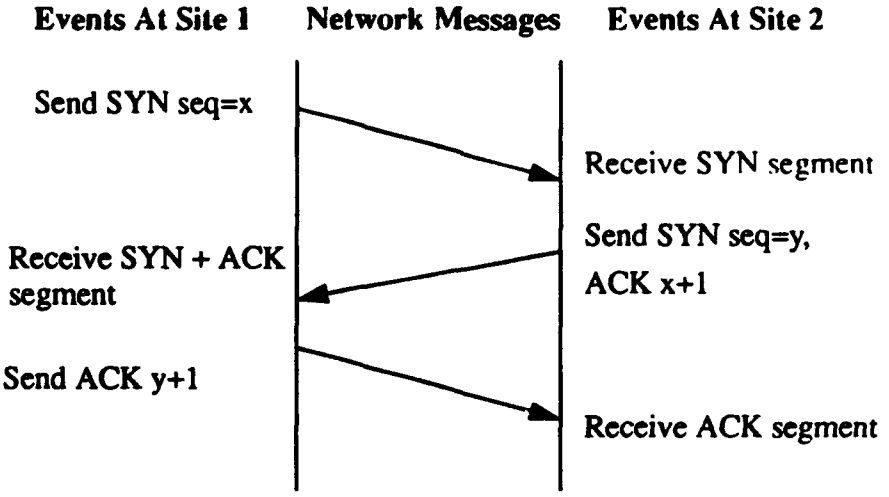



**Figure 9:** The Sequence of Messages in a Three-way Handshake.

The first segment can be identified because it has the synchronization bit set in the code field. The second message has both the synchronization bit and acknowledgment bit set and the final segment is only an acknowledgment to inform the destination that both sides agree that a connection has been established and data can flow in both directions equally well. There is no master or slave, and TCP ignores additional requests for connections once a connection has been established.

### 2. Connection Termination

To terminate a connection TCP uses a modified three way handshake (Figure 10). When one end has no more data to send, TCP closes the connection in one direction by sending a segment with the FIN bit set. The other end acknowledges the FIN segment but it is also able to continue to flow data. Of course, acknowledgments continue to flow back. When the second end decides to close the connection it sends a message with the FIN bit set and an acknowledgment, and when an acknowledgment is received the connection is terminated.

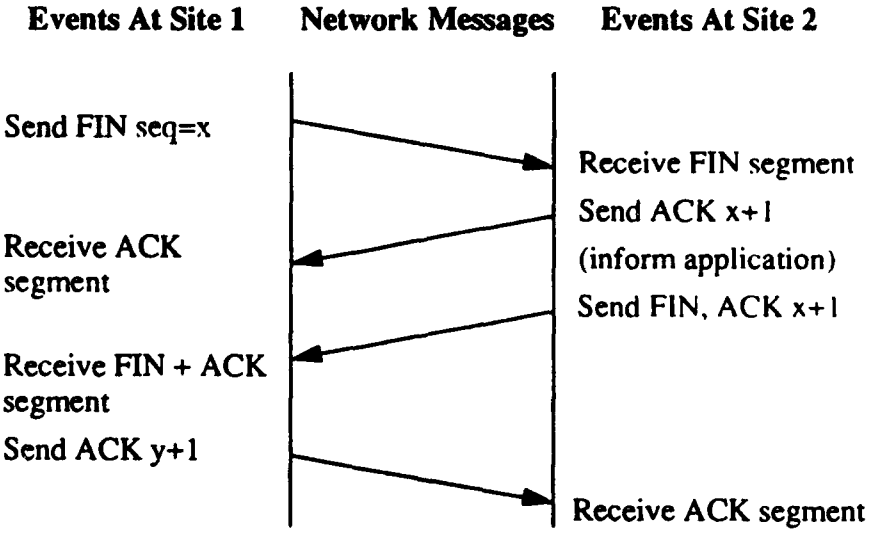



**Figure 10:** The Modified Three-way Handshake Used to Close Connections

The difference between three way, and the modified three way handshake is that in the modified way, the application program is informed after the request of closing the connection. The acknowledgment prevents retransmission of the initial FIN segment since the response may involve human interaction and take considerable time [COME91].

## D. DATA TRANSFER: FLOW CONTROL

TCP uses a sliding window to handle efficient transmission and flow control. The TCP window mechanism may send multiple segments before the arrival of an acknowledgment increasing the total throughput. The receiver is allowed to restrict transmission according to its available buffer space.

The protocol has a window size which determines the amount of data that can be transmitted before an acknowledgment is required (Figure 11). This amount of data is a number of bytes (octets). Octets are numbered sequentially and pointers separate octets that have been sent and acknowledged (1,2), that have been sent but not yet acknowledged (3,4,5), that have not been sent but will be sent without delay (6,7,8), and octets that can not be sent until the window slides (9,10). In order to provide flow control as well as reliable transfer, TCP uses a variable size window preventing the receiver's buffer to overflow decreasing the size of the window, or when the receiver's buffer has a lot of space, increasing the size of the window.

It is essential to have a mechanism for flow control, since machines of various speeds and sizes communicate through networks and gateways of various speeds and capacities. TCP uses its sliding window scheme to solve the end-to-end flow control problem, but it has no explicit mechanism to address data rate problems when intermediate systems (e.g. gateways) become overloaded (congestion control) [COME91].

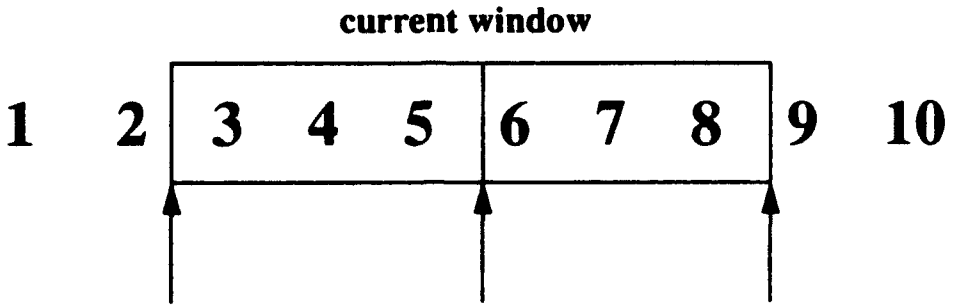



**Figure 11:** TCP Sliding Window. Pointers Show Limits of Different Groups of Octets

Congestion is an "excessive" number of packets in the network. When congestion occurs, delays increase and the gateways begin to enqueue datagrams until they can be routed or dropped. Endpoints, ignoring the existence of congestion in a gateway, timeout and retransmit the message. This will increase traffic and delays.

TCP must reduce transmission rates when congestion occurs. "Multiplicative decrease" and "slow-start" are the two methods that help to avoid congestion [COME91]. TCP assumes that most datagram loss is due to congestion and uses "multiplicative decrease" congestion avoidance. After a segment is lost, the window is reduced by half, down to a minimum of one, and the retransmission timer is doubled. Hence whenever congestion occurs, the traffic is reduced exponentially and the rate of retransmission time always increased. If loss continues, TCP limits transmission to a single datagram and continues to double timeout values before retransmitting until congestion has been avoided. TCP uses "slow-start" to recover from congestion. It starts the window at the size of a single segment and increases it by one segment each time an acknowledgment arrives. In fact this method is not very slow. When the first acknowledgment arrives, the window is increased and two segments are sent. When the two acknowledgments arrive they each

increase the window by one, so TCP can now send four segments. The window size is doubled every time the acknowledgments arrive correctly. For this reason, once the window reaches half of its original size, TCP reduces the rate of increment (congestion avoidance). It increases the window by one only if all segments in the window have been acknowledged. "Slow start" and "multiplicative decrease congestion avoidance" improve the performance of TCP without adding any significant overhead to the protocol software.

## E. DATA TRANSFER: ERROR CONTROL

Transmissions are made reliable through the use of sequence numbers and acknowledgments. Each octet of data is assigned a sequence number. Segments carry an acknowledgment number which is the number of the next octet that the receiver expects to receive. When the transmitter sends a message, TCP puts a copy of the transmitted segments in a retransmission queue, and a timer is started. When the acknowledgment is received the segment is removed from the queue. When the timer runs out, the segment is retransmitted. TCP uses an adaptive retransmission algorithm that monitors delays on each connection and adjusts its timers. Timers are both a necessity and a weak point of TCP. In [ZHAN86], the poor performance of timers in the TCP standard is explicitly detailed.

Segments travel in IP datagrams, and they can be lost or delivered out of order. The receiver reorders the segments according their sequence numbers. Acknowledgments are easy to generate, and lost acknowledgments do not force retransmission of segments, but a major disadvantage is that the sender receives information only about a specific position in the stream that has received and not about all successful transmissions. TCP uses the "go-back-n" method. Retransmission begins with the lost segment, making the protocol less efficient since there is no way for the receiver to inform the sender that most of the data

has been received correctly. TCP adopts a positive acknowledgment with retransmission (PAR). Under PAR, if the timer expires before an acknowledgment arrives at the transmitter, the transmitter retransmits the packets. The receiver acknowledges only when the packets have been received without error and in the correct sequence.

For example, assume a window that spans 6000 octets starting at position 601 in the stream, and suppose that the sender has transmitted all data in the window by sending five segments (Figure 12). If the first segment is lost, but all others arrive intact, the receiver will specify octet 601 as the next highest octet it expects to receive. After a timeout the sender retransmits all five segments. When the receiver receives the lost segment it specifies 6601. That acknowledgment may not reach the sender quickly enough to prevent the unnecessary retransmission of the other segments. If the sender sends only the first segment, it must wait for an acknowledgment before it can decide what to do. In this case it looses the advantages of having a large window.

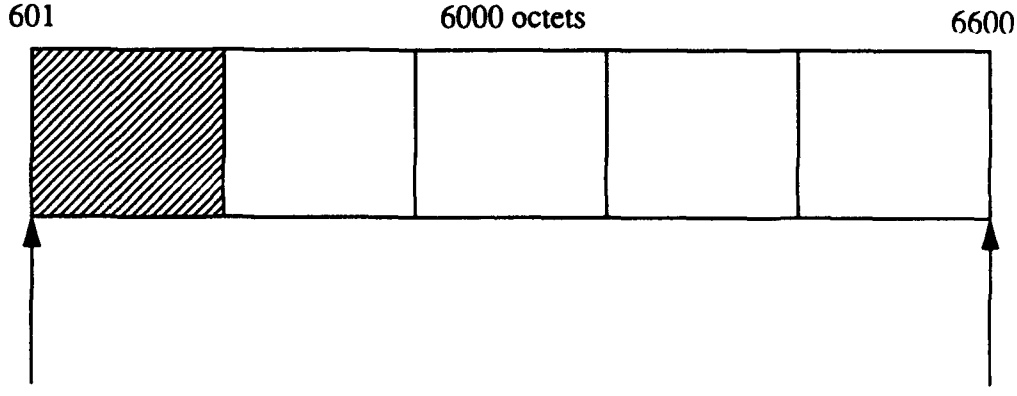



**Figure 12:** Retransmission After a Loss of a Segment.

## F. OTHER TCP/IP FEATURES

Until now, the main functions of TCP have been described. In this section, we present several features of the TCP/IP protocol suite. Some of them as multicast and fragmentation occur at the second layer of the TCP/IP protocol architecture (Figure 7).

### 1. Multicast

IP multicasting allows transmission of an IP datagram to a single multicast group, where its members are spread across different physical networks. Each multicast group has a unique multicast address of class D. Multicast gateways are used to forward multicast datagrams to their destination. TCP/IP defines IP multicast addressing, describes the protocol that gateways use to determine a multicast group and specifies how multicast datagrams are sent or received. Hosts communicate their group membership to multicast gateways using IGMP.

### 2. Address Resolution Cache

One of the protocols included in TCP/IP is the Address Resolution Protocol (ARP), which allow a host to find the physical address of a destination host on the same physical network, given only its IP address. Hosts, using ARP maintain a cache of previously acquired IP-to physical addresses in order to reduce communication costs. Receivers also update the IP-to physical address information in their cache before processing an ARP packet making address translation more efficient. In this way, a host always checks its cache before sending broadcast requests.

### 3. Fragmentation

TCP/IP software chooses a convenient initial size and arranges a way to divide large datagrams into smaller pieces called fragments, when the datagram needs to traverse a network that has a small maximum transfer unit (MTU). Fragmentation usually occurs at

a gateway, and the fragment must be reassemble to produce a complete copy of the original datagram before it can be processed at the destination. The fragment size must be a multiple of eight, and usually the last piece is shorter than the others.

In TCP/IP, once a datagram has been fragmented, it travels as separate datagrams all the way to its destination. The disadvantages are that after fragmentation, networks with large MTU are traversed by only small fragments, and if a fragment is lost the datagram can not be reassembled. If the reassembly timer expires before all fragments arrive, then the receiver discards all the pieces without processing the datagram. The advantage is that intermediate gateways have not to store or reassemble fragments.

**4 Service Type**

An 8-bit SERVICE TYPE field specifies how the IP datagram should be handled. Senders may indicate the importance of each datagram from values ranging from zero through seven (3 bits), and specify the type of transport they desire. Low delay, high throughput or high reliability are the options, but unfortunately, internet does not guarantee the type of transport requested. It is only a hint to the routing algorithm that helps to choose among various paths to a destination [COME91].

**5. Out of Band Data**

TCP allows the sender to specify data as urgent. The receiving program should be notified immediately of such an arrival regardless of its position in the stream. The receiving TCP then notifies the application program to fall into "urgent mode." The exact details of how TCP informs the application program depends of the computer's operating system. This is very useful especially when TCP is used for a remote login and the user wishes to interrupt or abort the program at the other side.

## G. SUMMARY

The TCP/IP Internet Protocol Suite, commonly referred to as TCP/IP after the names of its two main standards, was presented in this chapter. It can be used to communicate across any set of interconnected networks. The Transmission Control Protocol that provides the reliable, full duplex transmission of packets was described in detail. TCP allows a process on one machine to send a stream of data to a process on another. Software implementing TCP usually resides in the operating system and uses the IP protocol to transmit information across the internet.

TCP/IP is currently in widespread use, but it is not considered as a high speed transport protocol. Many protocols designers trying to improve the performance of the currently implemented protocols have designed new high speed transport protocols. In the next two chapters two of the new experimental lightweight transport protocols will be presented.

# IV. THE XPRESS TRANSFER PROTOCOL (XTP)

In this chapter the Xpress Transfer Protocol (XTP) is discussed. The basics of this protocol, the packet format, and its main functions are presented. XTP is a new high speed transport protocol designed in 1992 and implemented in experimental environments. For more details see [SDEW92], and [WEAV92].

## A. XTP OVERVIEW

XTP combines the functionality of transport and internet protocols to produce a protocol that requires minimum processing. A new technology for implementing communications protocols in hardware, the *Protocol Engine*, is described in [SDEW92]. XTP is designed to be implemented in hardware as a VLSI chip set. The Xpress Transfer Protocol represents the synthesis of the best ideas developed by several protocols and then extends those ideas with new services designed to support distributed applications and embedded real time systems [SDEW92].

XTP requires the underlying data delivery service to provide a simple framing capability, and a data delivery service between hosts. XTP packets must be placed in the protocol data units. They must also be encapsulated into frames of a well known standard as the Internet Protocol.

In a packet switching network, communication protocols are based on the exchange of data units called *packets*. In order to discuss the procedures and the main functions of this protocol we first describe the packet format used within XTP.

## A. XTP PACKET FORMAT

An XTP packet is divided into three parts: The header, the middle segment, and the trailer (Figure 13). The header specifies how to process the packet, including service options information, length and sequencing information, priority information and validity checks. The middle segment can be a control or an information segment. In the first case, it carries protocol state information, since in the second it contains addressing information and the user's data. Finally, the trailer consists only of the validity check over the middle segment.

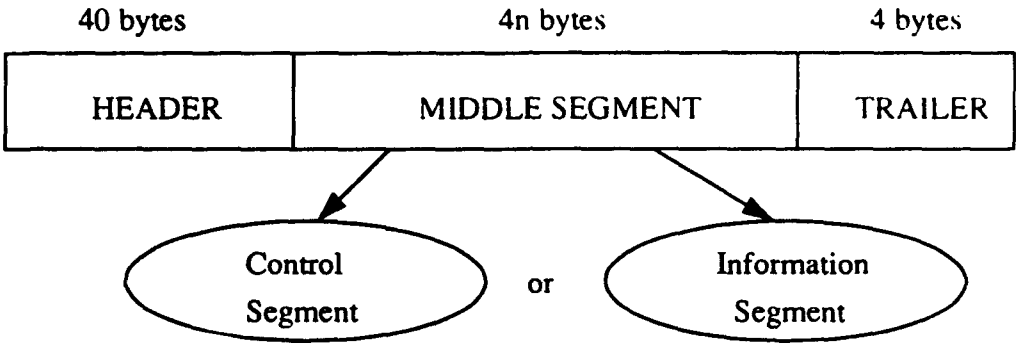



**Figure 13:** General XTP Packet Structure

### 1. XTP Header

The header is present in all XTP packets, and contains ten 4-byte fields (Figure 14). The *route* field identifies the host exit port used for sending this and all subsequent packets. A *time-to-live (ttl)* field indicates the time after which the packet will be discarded after an unexpected long delay. Packet's format and other command information is contained in the *command (cmd)* field. The *key* field is used to identify the packet and associate it with a particular context. The *synchronize (syn)* field associates

status request with the status responses. The *sequence number (seq)* field identifies where the data should be placed in the data stream. The *delivered sequence number (dseq)* indicates the amount of data received and delivered to the user. The *sort* field indicates the priority of the packet and the *data length (dlen)* field gives the length of the middle segment. Finally, the *header checksum (hcheck)* field contains the checksum over the header.

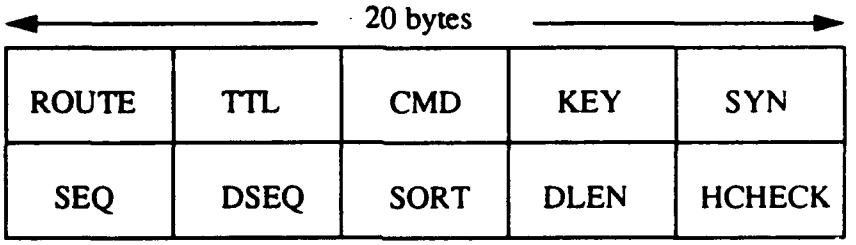



**Figure 14:** XTP Header

### 2. Middle Segment

The middle segment can be a control or information segment. Its length is variable depending on the spans information (control segment), or the user's data (information segment). The fields within the control segment represent the mechanisms used to exchange protocol state information between protocol state machines. Such information includes data reception status, rate and flow parameters, route identifiers, and values used to synchronize state machines. Twelve 4-byte fields are included in this segment in addition of a variable length spans field. The information segment provides the mechanisms to allow users transfer data. Four different forms of the information segment exist. It is intended to avoid getting into the details of each field format, since it is outside the scope of this thesis. For more details see [SDEW92], and [STAL88].

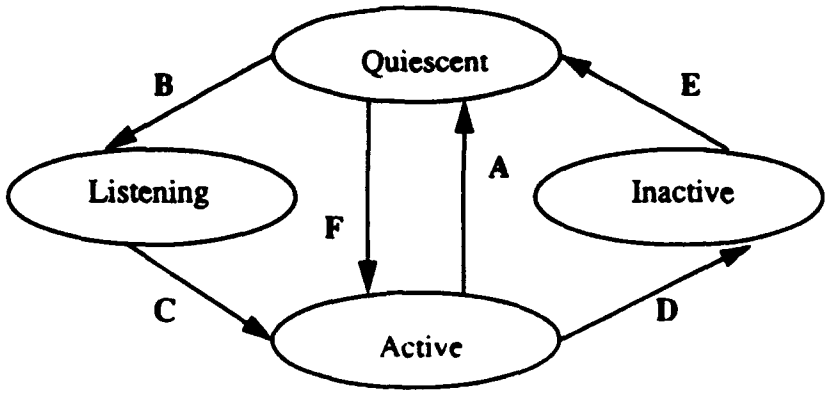
### 3. XTP Trailer

The packet trailer, like the packet header, is a component of all XTP packets. It consists of a single 4-byte data checksum field. When a packet is received at the destination, the checksum procedure is performed and then checked against the value in the trailer.

## C. CONNECTION MANAGEMENT

The connection management procedures enable two or more XTP users to establish a connection. When the connection is established, it uses these procedures to maintain the connection as long as the users desire. XTP implements an implicit way to establish, and a three way handshake to terminate a connection. The connection establishment is a different way than the currently implemented way of TCP. The implicit connection establishment that XTP adopts may save a significant overhead over the total transmission time of a message, especially for short lived connections. For this reason, we also describe the connection management procedures in terms of finite state machines.

Four states exist for establishing and terminating a connection: Quiescent, listening, active and inactive (Figure 15). When it is quiescent the shell is empty, and there is no connection. When the context moves to the listening state then it waits for the first packet which will establish the connection. As soon as the context moves into the active state, then it is mapped to some connection. In the inactive state the context is not allowed to receive or transmit data although the connection still exists. In this state it may only exchange information packets. Finally, the context becomes quiescent again when the connection is terminated. We discuss the establishment, maintenance, and termination of a connection by describing the actions that force a state to change.




A: Context activated - caused by an output command
B: Context "listens" for FIRST packet - caused by an input command
C: Context activated - caused by receipt of FIRST packet
D: Close of READER and WRITER
E: Connection terminated
F: Connection aborted

**Figure 15:** Context State Machine

### 1. Connection Establishment

Establishing a connection between two XTP users involves one user initiating the connection while the other listens for the indication of the start. The user issuing the input command will cause XTP to listen for any incoming "first" packet that is addressed to this specific address. The connection is initiated by the side that issues the output command. When the input command is issued, a quiescent context moves into the listening state. Then the first packet is assembled and sent. When the listening context receives the first packet, it moves into the active state and the connection is established (Figure 16).

The "first" packet contains an address segment which is used to identify the destination host. The connection establishment may require acknowledgment of success.

XTP user controls the degree of reliability required on the connection establishment The first packet can carry data. Thus although the main purpose of the first packet is to establish the connection, it can also be a data packet. If no buffer space is available to the receiving host, the first packet still establishes the connection and error controls recover the loss of data.

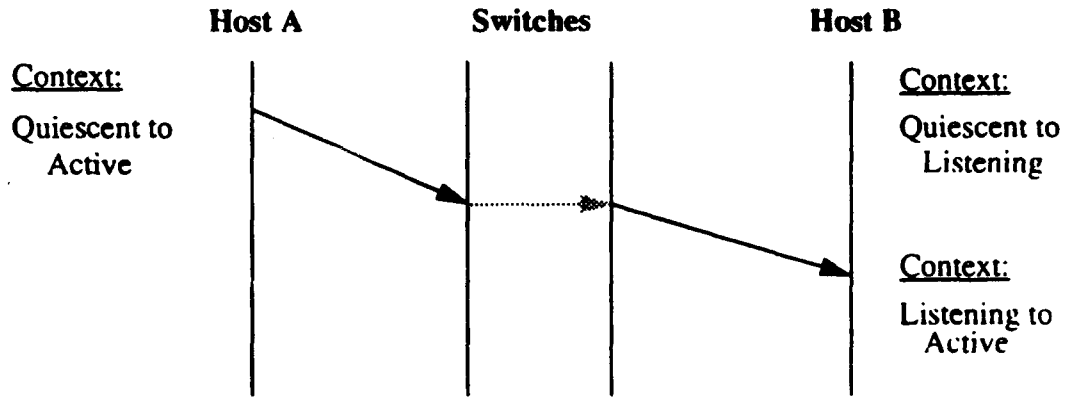



**Figure 16:** Connection Establishment Packet Exchange

### 2. Connection Termination

For terminating a connection XTP provides the mechanisms from an independent graceful close to an immediate abort. An independent graceful close ensures that all data in either direction of the connection is reliably delivered before the connection is terminated. An abort immediately closes the connection and data may be lost. An arbitrary length of time and an arbitrary number of messages can be exchanged. One or both of the endpoints start the connection termination procedures. In contrast to the connection establishment, terminating a connection gracefully requires a series of carefully guarded state changes. Contexts in the connection must assure that all data has

been delivered correctly before returning to the quiescent state. Three bits in the header of XTP packets indicate the progression of a connection termination. The WCLOSE which indicates that the sender's write process is closed and no new data will be sent. Retransmissions, if necessary, will be allowed. The RCLOSE which indicates that the sender's read process is closed and no new data will be read from the network. The receiver will not accept any packets after this bit is set. Finally, the END bit which indicates that no other communications of any kind are allowed and the connection is terminated (Figure 17).

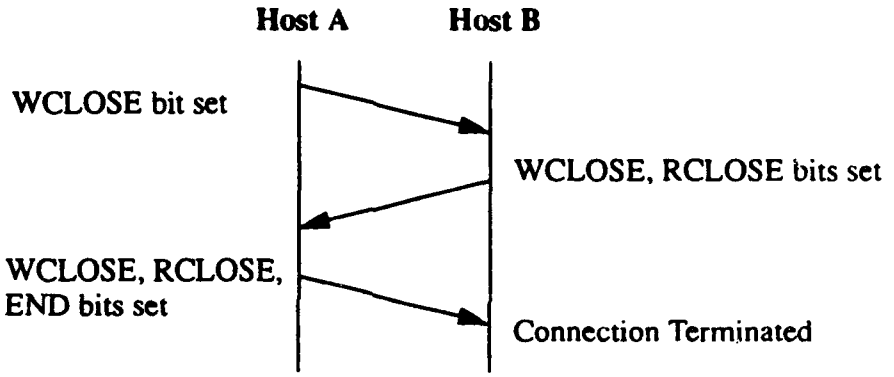



**Figure 17:** Graceful Termination

XTP achieves reliable transmission with the exchange of only two packets. Unlike TCP, connection close mechanisms enable the implementation of a variety of connection close methods. The connection release procedures include bidirectional graceful close, unidirectional graceful close with the initiating side forcing the close of the reverse stream, and forced terminations (aborts). Only graceful close ensures that all data sent will be delivered before the connection is terminated [WEAV92].

## D. DATA TRANSFER: FLOW CONTROL

Flow control regulates the volume of data flowing between the endpoints according to policies enforced by the receiver. In XTP is achieved through the use of parameters which provide visibility of the receiver's buffer to the transmitter. These parameters indicate which packets have been passed to the receiving host, which are currently in the buffer, which have not been received, and the current buffer space available to accept new packets. XTP provides the mechanisms and the procedures for an end-to-end based sliding window flow control algorithm. As the receiver is able to accept more data, more sequence numbers in the data stream are allocated to the transmitter, moving the upper edge of the transmitter's window forward. As the receiver acknowledges data received, the lower edge of the transmitter's window is moving forward. Thus the window slides across the sequence number space of the data stream at a rate dictated by the receiver's ability to accept and acknowledge new data [SDEW92].

The window in the transmitter represents the sequence number space of data that the receiver is willing to accept. The receiver must inform the transmitter the bounds of the window, sending the values of the upper and lower edges. As the receiver can accept more data, the value of the upper edge increases. Since the receiver may change these values with each control packet, the control window may be expanded, reduced, or even closed.

An example will make the flow control procedure easier to understand (Figure 18). Assuming that host A begins the transmission sending the "first" packet with 100 bytes of data (initial window size 0..99). A control packet is generated by host B. This packet will determine the window size, shown to be (0..199). The transmitter receives the control packet, updates its window, and sends 100 more bytes of data. At the other side, 60 bytes of data are delivered to the XTP user, and the receiver updates its control packet specifying the new upper and lower edges of the window, which are (60..259).

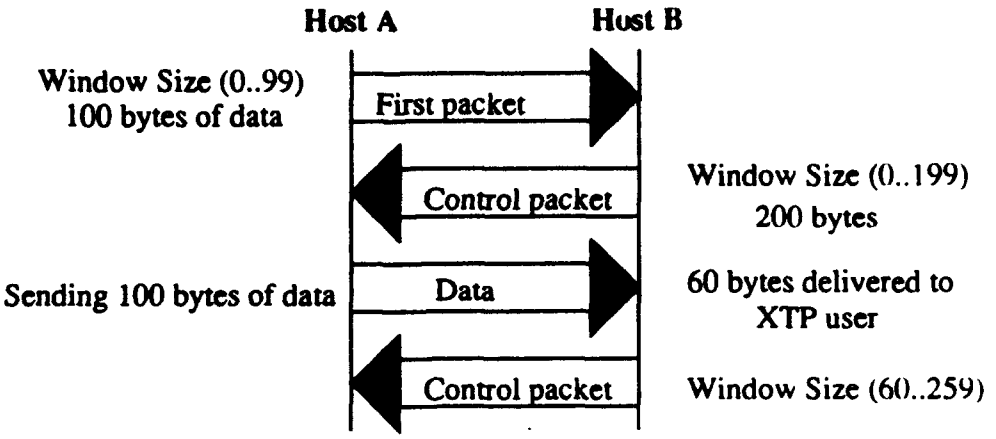



**Figure 18:** Packet Exchange Showing Flow Control Window

Another policy may be set by the transmitter in order to turn off the flow control procedures on the outgoing data stream. The receiver may accept this policy of no flow control or reject it. If the receiver refuses to accept the no flow control it must also reject the connection

## E. DATA TRANSFER: ERROR CONTROL

The error control procedures detect and recover from lost data. Received data are acknowledged and lost data retransmitted. The acknowledgment and retransmission procedures are related to the flow control procedures in that the window for flow control also represents the amount of data waiting for acknowledgment. Data from this window is retransmitted either from lack of acknowledgment or explicit negative acknowledgment.

XTP provides two kinds of acknowledgments. The first moves the lower edge of the flow control window indicating that data prior to this sequence number has been reliably

delivered and buffers holding this data may be released. The second indicates that the data has been received at the destination endpoint, but the XTP buffers can not be released until the lower edge of the window slides and passes the data they hold. Error control procedures are primarily concerned with the second kind of acknowledgment. XTP also allows the user to decide whether and when acknowledgments are desirable. They are requested by the transmitter by setting a status request bit.

Three methods of error recovery are possible. First, when the receiver receives an out of order sequence number, it can determine that a gap exists and packets may be lost. It informs the transmitter of the last correctly received packet and the sender retransmits all data from that point. This policy is known as "go-back-n". It may be easier for the transmitter to retransmit more than the amount required to cover the range, and hence some duplicate data may be sent. Second, the transmitter may request from the receiver a list of gaps for which sequence numbers are missing. These gaps in the data stream are determined from the spans of correctly received data. The span is a contiguous group of data. It is represented by a pair of sequence numbers indicating the beginning and the end of the group. When the sender receives that information, all the knowledge necessary to conduct selective retransmission is present. Using selective retransmission policy the transmitter will fill the gaps of the receiver with the missing data. Third, if the transmitter does not receive any indication from the receiver, it will time-out, prompting an additional request for gaps [LUND92]. Three timers and two context variables are used in an association to help the error detection and recovery procedures [SDEW92].

An example of selective retransmission is shown in Figure 19. The transmitter has just received a control packet. The beginning and the ending sequence numbers of sent but not acknowledged data were 5, and 26 respectively. The receiver indicates that the transmitter may retransmit data packets covering the range (6..25). The number of spans is

2, and the spans are (10,14), and (19,22). In this case the transmitter will retransmit three data packets: (6..9), (14..18), and (22..25).

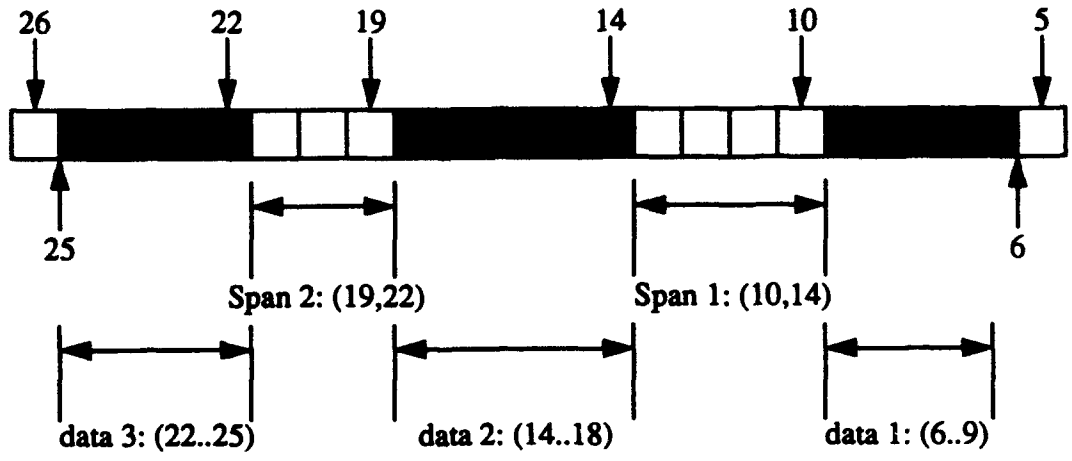



**Figure 19:** Selective Retransmission

The transmitter may also choose to turn off retransmission by selecting the no-error mode. In this case, the receiver ignores data stream errors and does not request retransmissions. Control packets continue to flow, but error control actions are not taken.

## F. OTHER XTP FEATURES

The main functions of XTP have been described in this chapter. However, XTP provides many features designed for high throughput, low latency, and better performance. In this section, several features will be mentioned. Especially rate control is one of the most important. It could be described in the same section with flow control. Nevertheless, it will be more effective to be presented as an entity of itself.

### 1. Multicast

XTP provides a powerful mechanism for group communication. A multicast transmitter may address an arbitrary large receiver group, replacing a large number of transmissions with a single multicast. Data transmission is reliable. Receivers may ack for retransmission of lost data. A multicast connection operates under one of two error control policies. If error control is not requested, the XTP multicast connection operates under the no-error mode. The transmitter has the possibility to request reception status, flow and rate control information from the receiver group.

### 2. Address Translation

Connection establishment requires the resolution of a network address to a connection identifier. Sending large addresses repeatedly reduces network efficiency. A form of address translation, once the connection is established is used by XTP in order to minimize this overhead. The first XTP packet contains an address segment and an address format field. This address segment holds all of the information necessary to deliver a packet from end-to-end across the network, including location of the destination host, and establishment of a path in intermediate nodes between the two endpoints. Through a hashing function, initial addresses are resolved into a 32 bit key field and subsequent packets can be identified by a simple table lookup.

### 3. Rate and Burst Control

While flow control applies to data transfer in a connection and reflects the flow of XTP buffer space at the endpoints, rate control is an attribute of a path, and each XTP node along the path participates. Each node receives packets and forwards them, hence each pair of nodes in the path forms a producer/consumer pair. Rate control allows the receiver to specify in bytes per second a maximum rate at which data can be consumed.

This helps synchronize not only the transmitter and receiver but the intermediate nodes as well. Burst control allows the receiver to specify the maximum number of bytes which can be consumed in one burst of packets, that is, packets sent in rapid succession. The transmitter is allowed to transmit the number of bytes in the burst field as a single continuous transmission of one or more packets. Then the transmitter must wait for a period of time (burst/rate seconds) before another such burst of data may be transmitted. In this case flow control is increasing and buffer starvation is avoided.

The receiving end-node sets the rate and burst values by determining how much data it can process over a period of a time. All the intermediate nodes along the path may participate in the rate and burst control procedures. Each node is permitted to reduce the values of the rate and burst field of any control packet, depending of its capabilities of consuming data. Thus the data generated by the transmitter is the minimum acceptable rate of all the nodes along the path. [SDEW92]

An example will show how rate control may affect the transmission of a 250 Mbytes message between two end-points. We assume that the transmitter operates at 200 Mbps, and the receiver can process only 100 Mbps. In this case, flow control will assure that the receiver's buffer will not be overwhelmed, and the sender will transmit the message at 100 Mbps. This means that the message will be transmitted in 20 seconds without any problem at the receiver.

Assuming, that an intermediate node with a 60 Mbytes buffer can process only 60 Mbps, then after 12 seconds, the buffer of this node will be full (Figure 20). Packets will be lost, since this intermediate node is not able to operate at such a high speed. If rate control was applied, then the minimum acceptable rate at which the transmitter should operate would be 60 Mbps, resulting in the correct reception of the message.

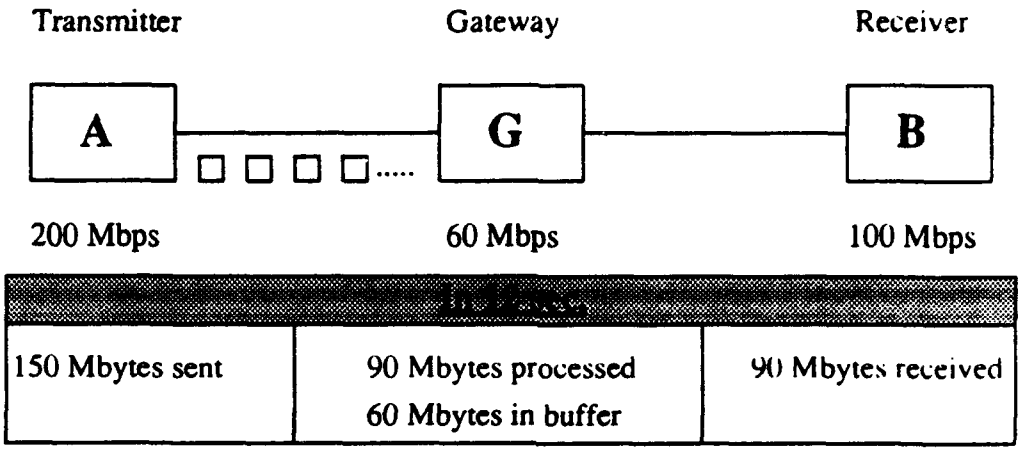



**Figure 20:** Rate Control

### 4. Priorities

XTP includes a sort field in the header segment, which permits packets to be assigned different priorities. Higher priority packets receive preferential treatment. Packets with the sort bit set are always processed first. The sort value is a 32-bit field, with over than 4 billions priority levels ($2^{32}$). Smaller values denote higher priority. Users may optionally encode the sort value to indicate a message's relative priority. At each transmission opportunity, a node selects its most important packet and operates on it.

### 5. Out-of-Band Data

XTP provides a small channel for out-of-band data. Each XTP packet may carry an 8 byte tag of data at the beginning of the packet. When a tagged packet is received XTP notifies the user of the tag's appearance. Tags can carry management information for the purposes of higher layer protocols, information about the data itself, or to timestamp time critical data. [WEAV92]

## G. SUMMARY

An overview of the Xpress Transfer Protocol (XTP) was presented in this chapter. The components used to build XTP packets as the header, the trailer, the control segment and the information segment were shown. The protocol procedures of establishing, maintaining and terminating XTP connections, controlling data flow and rate, and detecting and correcting errors were discussed. Finally, several special features of XTP were described.

XTP is among the lightweight transport protocols designed in the last few years, in order to improve the performance of the current conventional transport protocols. In the next chapter the basic ideas of the SNR high speed transport protocol will be presented.

# V. SNR: A HIGH SPEED TRANSPORT PROTOCOL

In this chapter the SNR protocol is discussed. The SNR packet format is shown; control and data packets are included. Finally the main functions of the protocol are described, and special features mentioned.

## A. PROTOCOL SPECIFICATION

The SNR transport protocol is an experimental lightweight protocol introduced in [SABN90], capable of high throughput with the evolving high speed networks, based on fiber optic transmission lines. Its purpose is to overcome the insufficiencies experienced by the current transport protocols.

In SNR, packet sizes are fixed for each connection, but can vary for different connections. The protocol provides error recovery, sequenced delivery, flow control, multiplexing/demultiplexing and different modes of operation. SNR uses the selective repeat method of retransmission, and the concept of blocking. Blocking reduces the overhead of maintaining large tables and complex procedures that are required for selective repeat procedures. Control packets are used for periodic and frequent transmission of complete state information of the communicating entities.

The SNR protocol specification consists of eight machines (four in the transmitter and four in the receiver), each of which performs a specific function [TIPI93]. These machines operate almost independently with a small amount of interaction between them (Figure 21).

Machine T1 selects a new block and schedules it for transmission provided that the

buffer at the receiver can process it. It is also responsible to determine if an old block needs retransmission. Machine T2 establishes the connection with the receiver by a three-way handshake and thereafter processes the incoming receiver control packets, and updates related tables and variables as the blocks are acknowledged. Machine T3 sends transmitter control packets to the receiver at regular intervals. In case of conflict, control packets get priority over data packets. Machine T4 is the host interface of the transmitter. It inserts the incoming data stream into the buffer for transmission by machine T1.

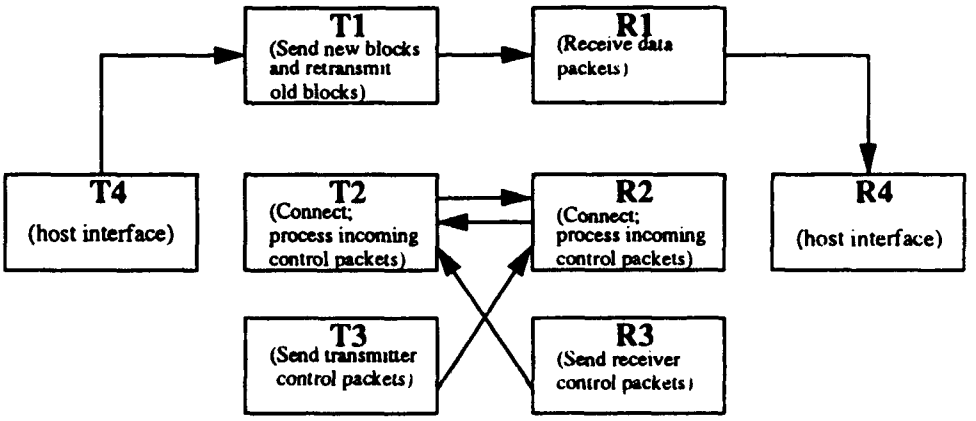



**Figure 21:** Machine Organization

Machine R1 removes the data packets from the transmitter channel and inserts them into the buffer in order according to their sequence number, and updates the related variables and tables in the receiver. Machine R2 sets up the connection in cooperation with T2 in the transmitter. After the connection establishment, it receives the transmitter control packets and is responsible for terminating the connection. Machine R3 sends the receiver state information periodically to the transmitter through the receiver channel.

Finally, machine R4 is the host interface of the receiver. It retrieves the data packets from the buffer, if they are in sequential order, and passes them to the host. For more details of the protocol see [SABN90] and [LUND93].

## B. SNR PACKET FORMATS

The packets exchanged between the transmitter and the receiver can be divided into four categories: connection control packets, transmitter control packets, receiver control packets and data packets.

### 1. Connection Control Packets

These packets are used for connection management purposes. Connection request, connection acknowledgment, connection confirmation and disconnect are some of these connection messages. They do not carry any user data and their format is similar to that of the current implemented transport protocols.

### 2. Transmitter Control Packets

Transmitter control packets are used for periodic transmission of complete state information of the transmitter, even when the data transfer ceases for a while. The packet format is shown in Figure 22.

The first field contains the logical connection identifier (*LCI*), which is a unique sequence number assigned to each logical connection during the connection establishment phase. The *type field*, identifies that this packet contains the transmitter's state (type=1). The next field (*seq #*) is used only for datagram networks and contains a sequence number, since out of sequence control packets are discarded. The control packet transmission period $T_{in}$, is calculated by the formula $Tin$ = max (RTD / *kou*, *IPT*), where RTD is the estimated round trip delay, the constant *kou* is typically a power of 2, such as 32, and *IPT*

is the average time between two data packet transmissions [TIPI93]. The variable $k$ is the interval between two control packet transmissions, expressed in units of $T_{in}$. The number of blocks queued for transmission, and the maximum sequence number of the block below which every block has been transmitted but not necessarily acknowledged as known at the transmitter ($UW_t$) are also contained in this packet. Finally, the last field contains an error detection code (*error check*).

| LCI | Type = 1 | Seq # | $k$ | $UW_t$ | No. of blocks queued | Error check |
|---|---|---|---|---|---|---|

**Figure 22:** Transmitter Control Packet Format

The queue length, as explained in [SABN90], can be used for a variety of purposes, such as congestion control within the network, to decide whether the receiver should accept another connection, etc. It can also be used when the transmitter does not have enough packets to complete a block. In such a case the transmitter sends a partial block and the receiver does not classify this as an outstanding block.

### 3. Receiver Control Packets

Receiver control packets are used for periodic transmission of complete state information of the receiver. The packet format is shown in Figure 23.

The first field contains the *LCI*. The *type field*, identifies that this packet contains the receiver's state (type=0). *Seq #* field contains the sequence number of the packet which is separate than the sequence number of the transmitter control packets or the data packets. The variable $k$ is again the interval between two control packet

transmissions of receiver in units of $T_{in}$. The maximum sequence number of the block below which every packet in every block has been correctly received as known at the receiver ($LW_r$) is contained in the next field. Buffers available at the receiver in units of blocks, and a bit map representing the number of outstanding blocks (*LOB*) are also contained in this control packet. Finally, the last field contains an error detection code (*error check*).

| LCI | Type = 0 | Seq # | $k$ | $LW_r$ | Buffer_available | *LOB* | Error check |
|---|---|---|---|---|---|---|---|

**Figure 23:** Receiver Control Packet Format

### 4. Data Packets

A data packet, as its name implies, is used to send user's data. Its format is shown in Figure 24. The purpose of the first three fields is the same as the transmitter and receiver control packets. This time, *type field*, identifies that the packet contains data (type=2).

The sequence number (*seq #*) of a packet contains $s = n + k$ number of bits, where the first $n$ bits represent the sequence number of the block which contains the packet, and the next $k$ bits represent the packet number in the block. If a block consists of $2^k$ packets, then the message can have $2^n$ blocks.

As mentioned earlier, the length of the data packets is negotiated during the connection establishment, and then remains fixed. If a message does not fit exactly in a multiple number of packets, the space left in the last packet is padded with null characters. This simplifies the packet processing in the receiver.

| LCI | Type = 2 | Seq # | Data | Error check |
|---|---|---|---|---|

**Figure 24:** Data Packet Format

## C. CONNECTION MANAGEMENT

In SNR, the transmitter is responsible for establishing and terminating a connection. The initial connection establishment phase is based on the standard three way hanshake similar to that of TCP/IP. During the connection establishment the following parameters are negotiated between the transmitter and the receiver: mode of communication (0, 1 or 2), peak bandwidth per connection, packet size, block size, and buffer required at the receiver in units of blocks. An estimate of the round trip delay (RTD) is also calculated during the connection establishment. After all the data have been received correctly from the receiver, the transmitter starts the process for terminating gracefully the connection.

### 1. Connection Establishment

Two machines are involved in connection establishment. The first is the transmitter's machine T2, and the second is the receiver's machine R2. The transmitter (machine T2) sends a connection request message which includes the parameter values that the transmitting host specifies. The receiver (machine R2) evaluates the requested parameters and responds by sending a connection acknowledgment message that includes the modified values for the parameters under which the receiver can operate (Figure 25). If the transmitter does not receive a connection acknowledgment in a certain time, it retransmits the connection request again. After certain number of unsuccessful attempts,

the transmitter notifies the host of unsuccessful connection.

When the transmitter receives the connection acknowledgment, it evaluates the parameters and if the values are acceptable by the transmitter, then it transmits a connection confirmation message and the data transfer phase begins. The transmitter rejects the connection and notifies the host if the connection failed. The receiver enters the data transfer phase when the confirmation message or the first data packet is received.

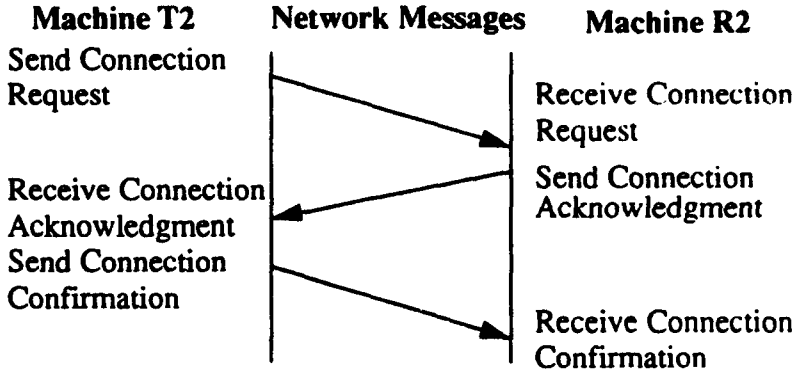



**Figure 25:** Connection Establishment

### 2. Connection Termination

For terminating a connection, a disconnect message is sent by the transmitter, after all the data packets have been transmitted. The receiver acknowledges the request for closing the connection, and finally the connection is terminated when the transmitter sends back the final acknowledgment (Figure 26).

An abnormal termination of the connection may occur if control packets are not received for a certain period of time. In this case, the transmitter sends a disconnect message and leaves the network, and the receiver aborts the connection after it timeouts. Machines T4 and R4 are responsible to notify their hosts of the abnormal disconnection.

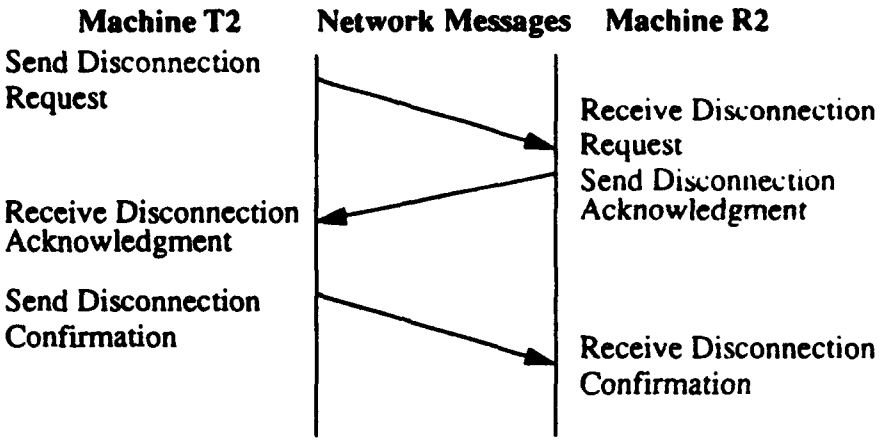



**Figure 26:** Connection Termination

## D. DATA TRANSFER: FLOW CONTROL

The SNR protocol uses the concept of blocking. A group of packets (typically 8) is called a block. The receiver writes the available buffer space it has in units of blocks into a specific field (buffer_available) of the receiver control packets. As the window size increases, the throughput increases, because the larger window size lets the transmitter to keep the pipeline full. On the other hand, the transmitter and receiver must reserve approximately one window's worth of packet buffers for each active connection, so the larger the window size, the more buffer memory is required at the receiver [SABN90]. because the state is expressed in blocks rather than packets, the window size must be a multiple of the block size. The maximum window size in units of blocks is chosen to be slightly larger than:

(round trip delay) x (maximum bandwidth) / (number of bits in a block)

Every time the transmitter completes the transmission of a block, it increments the number of outstanding blocks which have been transmitted but not acknowledged yet.

Every time it receives a receiver control packet, it decrements it by the number of acknowledged blocks. The transmitter starts the transmission of a new block only if the number of outstanding block not acknowledged is less than the maximum window size, and less than the buffer's available space. In this way, the transmitter keeps track of the acknowledged blocks avoiding to overflow the receiver's buffer. The above check is not necessary for retransmissions, since the retransmitted block has already reserved buffer space in the receiver during the first transmission. The outstanding blocks which have been transmitted but not yet acknowledged have already been incremented [TIPI93].

## E. DATA TRANSFER: ERROR CONTROL

The SNR protocol uses a modified selective repeat error recovery mechanism to perform packet retransmission efficiently. This service is provided only in mode 2 operation. In this mode, if any packet in a block is delivered incorrectly, the entire block is retransmitted. Blocking makes throughput almost independent of the variations in round trip delay, while keeping the processing within reasonable limits [SABN90]. It also reduces the overhead of the large tables and complex processing that are required if packets were acknowledged or retransmitted each one separately. The receiver acknowledges blocks, and not the individual packets. If a block has not been acknowledged for a predetermined amount of time, then all the packets of the block are retransmitted.

Since there is no error recovery in mode 1, a problem is encountered when the packets get lost during the data transfer in this mode. A solution to the problem of how long should the receiver wait for the lost packets and what action should be taken is suggested in [LUND93].

The predetermined amount of time for waiting an acknowledgment depends on a specific counter. After the transmission of a block of packets, an entry is made into a table whose index is calculated from the block sequence number. The counter takes the value

(round trip delay) x (control packet transmission period) + constant

(where constant may have the value 2). Therefore the block will not have to be retransmitted if the acknowledgment arrives a little late, since the time out period was made larger than the round trip delay. When the block is acknowledged, the block entry is removed from the table. After every reception of a receiver control packet the counter is decremented. A block is scheduled for retransmission only if it is not acknowledged and the counter is zero. This error recovery scheme eliminates the following: the need for explicit retransmission timers through the use of counters and periodic transmission of the control packets, the need for recalculation of round trip delay in order to adjust the retransmission timeout values, and the unnecessary control packets transmissions [SABN90].

An example is shown in Figure 27. The receiver has already acknowledged blocks 2, 3 and 4, but since some packets (4 and 6) of block 1 have not been correctly received, block 1 has not been acknowledged. Block entries 2, 3 and 4 have been removed from the transmitter's table, and the counter is decremented. When the counter will reach the value zero, block 1 will be retransmitted, since it has not been acknowledged.

In connectionless data transfer, is also possible that data packets will arrive out of order, since they may traverse different paths through the network. Sequenced delivery is provided in modes 1 and 2. The receiver is responsible for reordering the data packets according to their sequence numbers. This service also includes the detection of duplicate packets. A check is made to the sequence number of the received packet. If the packet is a duplicate, it is discarded.

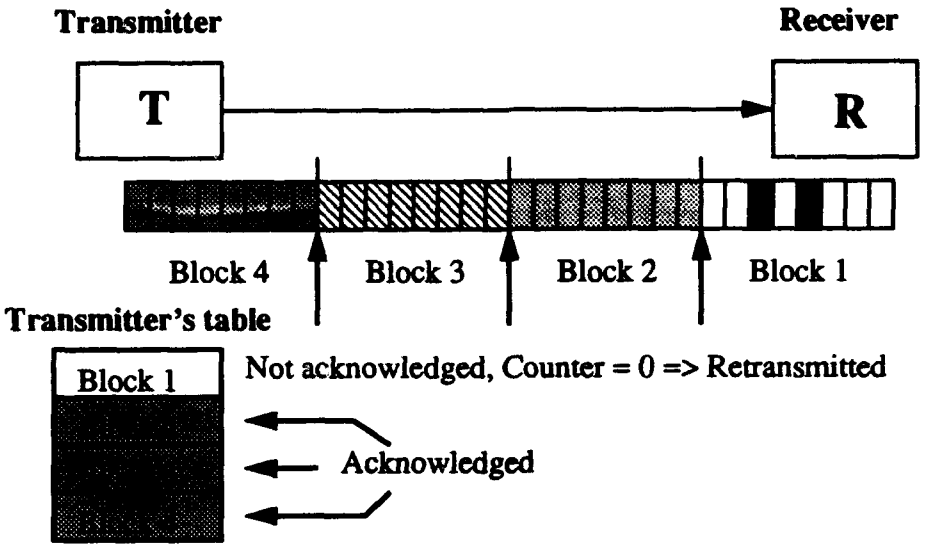



**Figure 27:** Selective Retransmission and Blocking

## F. OTHER SNR FEATURES

The packet format and the main functions of the SNR transport protocol have been described. Additionally, in this section we present two more important features of the protocol.

### 1. Modes of Operation

The SNR transport protocol has three different modes of operation. Mode 0 has no error or flow control. It is suited for virtual circuit networks in which quick interaction is desired [SABN90]. Mode 1 has no error control. Real time applications can be supported by this. Mode 2 has both error and flow control. For that reason, it is better for large file transfers.

### 2. Multiplexing/Demultiplexing

Multiplexing and demultiplexing are provided in all three modes. The front-end

processor implementing the protocol is thought as connecting several hosts and many logical connections within each host to the network.

## G. SUMMARY

The specification of the SNR transport protocol was reviewed in this chapter. The format of the different packets exchanged between the transmitter and the receiver were shown. The protocol main functions of connection management, flow and error control were presented.

SNR is the final transport protocol that was described in this thesis. In the next chapter the comparison of the three protocols will be presented (TCP/IP, XTP, and SNR), and the strengths and weaknesses of each protocol will be discussed.

# VI. COMPARISON: XTP, SNR, AND TCP/IP

In this chapter we compare three different transport layer protocols - TCP/IP, XTP and SNR. The TCP/IP architecture and protocols acquired their current form in the late 1970s [MINO91]. XTP and SNR are new experimental transport layer protocols designed with high speed networks in mind. Since there is a simultaneous need for richer services and more efficient execution two basic approaches are available:

(1) Select TCP/IP and modify it as necessary to add the required new features to meet timing requirements; or,

(2) Implement a new protocol with mechanism specifically designed for high-bandwidth, low-delay and low-error communications.

We summarize the strengths and weaknesses of these protocols. We discuss rate and flow control, error control, packet formats, connection management, simplicity and overhead factors, and special features. Finally, we also mention some of the economical and political factors.

## A. RATE AND FLOW CONTROL

Flow control assumes an increased importance in high performance networks. It has been noted that the major source of packet loss in high performance networks is a node congestion causing packets to be discarded [CHER89]. The performance of the overall system is degraded if data is transmitted faster than the data rate of the system or faster than the processing rate of the receiver. Rate control is also an important method for controlling the sender's data flow. The sender must know the rate at which data can be

transmitted and the burst size, which specifies the maximum amount of data that may be sent in a burst of packets [CLAZ87].

TCP/IP provides end-to-end control as well as reliable transfer using a variable size window. The receiver specifies the maximum amount of data that the sender may transmit and hence overrun is not possible. One problem with TCP/IP is the lack of rate control. For that reason TCP uses estimates of the round-trip delay as an indication of network congestion (multiplicative decrease congestion avoidance) and uses the "slow-start" recovery to increase the window. Thus, this congestion control scheme implements an implicit form of access control. TCP/IP does not provide rate control. The slow-start recovery was also developed to overcome the problem of the initial large window. The initial window is small, and increases as long as acknowledgments arrive properly. Data is transmitted at the rate at which it can successfully travel through the network and be accepted by the receiver. Nevertheless, for short connections, throughput is reduced due to the initial small window [DOER90].

XTP implements both end-to-end flow, rate and burst control. Initially, default parameters are used to control the transmission rate. However, during an exchange of control information the receiver specifies flow control parameters which may be modified by network nodes to control congestion. A big advantage of XTP is that rate control and window flow control may be used together. In this case, packets are transmitted under the restrictions of the rate control algorithm as long as the window remains open [DOER90].

SNR also provides end-to-end flow control and reliable transfer using a variable size window. The size of the window is selected to be a multiple of eight blocks. A major advantage of SNR is the implementation of blocking. The transmitter sends a new block only if the number of outstanding blocks, is less than the maximum window size, and less than the buffer's available space. In this way the sender avoids overflowing the receiver's

buffer. SNR does not provide rate control.

In our view, rate control is an excellent way to help control both flow and congestion; so from this standpoint XTP has a decided advantage over both SNR and TCP. However, SNR's implicit timers, discussed in Chapter V also help to control congestion by expanding timeouts when the network is congested; so SNR has an advantage also. It seems clear that both of these are an improvement upon the simple flow control scheme of TCP, but it is not clear to us which is better, or if both should be used.

## B. ERROR CONTROL METHOD

A transport protocol must perform error detection, reporting and correction or retransmission. Sequence numbers, length fields and checksums are responsible for error detection. Selective reject or negative acknowledgments inform the sender about errors detected by the receiver, and retransmission provides error recovery. The biggest difference between these three protocols is the procedure used for error correction.

TCP/IP uses a type of "go-back-n" protocol which is called positive acknowledgment with retransmission (PAR). Every byte has a sequence number; the receiver acknowledges packets by sending the sequence number of the next expected byte. If a message is lost, it and all subsequent messages must be retransmitted. Timers are used to force retransmissions. When these timers run out the segments are retransmitted. This method minimizes the processing requirements at the receiver. However it is difficult to determine the timeout values for automatic retransmission, especially in networks where the round trip delay varies [JACO88]. The poor performance of timers in the TCP standard is detailed in [ZHAN86]. Acknowledgments are very easy to generate and lost acknowledgments do not force retransmission, but the disadvantage is that the sender does

not receive information about all successful transmissions. The retransmission uses "go-back-n" scheme and some correctly received segments may be retransmitted. This scheme makes the protocol very simple, but also less efficient. The cost of using "go-back-n" may cause many goods packets to be retransmitted, causing a significant loss of bandwidth especially for high speed networks with long propagation delay. [DOER90]. An additional disadvantage of the existing transport protocols, and especially TCP/IP, is that flow control mechanisms are commonly tied in with error detection and recovery. High speed networks require the separation of the two functions. Flow control mechanisms have to address how fast packets may be transmitted, rather than how many may be outstanding at a given time [MCAR92].

XTP adopts automatic repeat request (ARQ), where data is retransmitted according to the information contained in the acknowledgment. This protocol may use "go-back-n" forcing the sender to retransmit all data from a specific point or selective retransmission, where only missing data are retransmitted. Spanning provides the necessary information of which messages have been correctly received. Timers are also used, prompting additional requests when the transmitter does not receive any indication from the receiver. In XTP, a synchronization mechanism is also available between transmitter and receiver [SDEW92]. Selective retransmission is used assuming that the receiver has a reordering buffer in which out of sequence packets can be handled. This increases the processing requirements of the receiver, but it is desirable in long delay or high bandwidth connections in which great amount of data may be in transit.

SNR uses a slightly different retransmission policy. It uses a modified selective repeat error recovery mechanism to perform packet retransmission efficiently. In order to avoid large tables and complex processing it uses the concept of blocking reducing the overhead of such tables. The receiver acknowledges blocks (groups of eight packets) and

when a packet is received incorrectly the entire block is retransmitted. SNR also allows exchange of messages that contain complete state between the transmitter and the receiver (control packets). Blocking, coupled with the periodic transmission of state makes throughput almost independent of the variations in round trip delay, while keeping the processing within reasonable limits [SABN90]. Another big advantage of SNR's error recovery scheme is that eliminates the need for explicit retransmission timers through the use of counters and periodic transmission of the control packets. This eliminates the need for recalculation of round trip delay in order to adjust the retransmission timeout values.

Error control method is a major consideration in every protocol. SNR seems to have the best procedure, taking advantage of the high bandwidth and avoiding processing overheads. Using constant size packets and performing block selective repeat, many implementation difficulties have been overcome [SABN90].

## C. PACKET FORMATS

In a packet-switched network, traffic is divided into small pieces called packets that are multiplexed onto high capacity intermachine connections. A packet carries identification that enables computers to establish connections, to transfer data, to send acknowledgments and other important functions. These fields carrying control information are referred as the packet formats. Every protocol has a specific packet format and a common header to indicate the layout of the remaining portion of the packet.

The size and the alignment of control fields in a packet header has a significant impact on the speed at which these headers can be constructed and parsed [DOER90]. Data alignment is also a very important notion. Additionally, variable-length control fields need more memory access. With the order of magnitude increase in raw bandwidth made

available by fiber optic media, variable-length packets may take excessive processing time. One of the most significant performance measures of any protocol is its ability to avoid buffer overflow by passing incoming packets on to the receiving host with a minimal amount of processing. By using standard format packets, parallel processing method will significantly reduce the time necessary to process a data packet [MCAR92]. Thus fixed-length fields seem to be a better choice. The placement of fields is also important. As an example, checksum is better to be placed at the end of the packet providing the possibility for parallel processing.

TCP/IP uses a rather complicated packet format with checksum field before the data. This implies that the whole packet must be processed to compute the checksum before transmission can begin. Thus, every byte of every packet must be handled at least twice. First for computing the checksum, and second for transmitting it. Parallel computation is difficult or impossible. A second disadvantage is that many fields are only 16 bits long, as for example the window field, and this may become a severe limitation in the near future.

In XTP, most of the header fields are 32 bits long. This sacrifices communication bandwidth, but makes header processing easier. XTP fixes the length of the header and trailer and the p[illegible] of all flags within the header. The disadvantage of XTP is the use of complex packet formats.

SNR, in our opinion, uses the best approach. Its packet format is very simple. The length of the data packets are constant throughout the connection, which is determined during the connection establishment phase. If a message does not fit into an integral number of packets, then, depending on the implementation, the space in the last packet can be padded with null characters [TIPI93]. This is intended to simplify the packet processing in the receiver. Another advantage of SNR is that it uses just three kinds of packets: Transmitter control packet, receiver control packet, and data packet. This makes

processing of a packet more efficient.

## D. CONNECTION MANAGEMENT

A connection is the path between two protocol modules that provides reliable stream delivery service. During a connection three phases occur: connection establishment, data transfer and connection termination. During the connection establishment phase, two entities agree to exchange data, and a number of parameters for the transfer of data must be negotiated. These parameters (sequence numbers, window size, round trip delay) form the state information required to transfer data reliably. Some of them may be renegotiate due to changing conditions. During the data transfer both data and control information is exchanged. Finally, one of the sides terminates the connection by sending a termination request.

TCP/IP uses a three way handshake to establish and terminate a connection. The problem in this scheme is that exchanged messages are required to open and close a connection. Especially for short connections these messages cause a severe overhead. Parameters are negotiated during the connection setup and updated during data transfer. In order to ensure that both sides close reliably, TCP's closing handshake requires the use of a timer at the endpoint initiating the close procedure. The endpoint retains state information until the final acknowledgment releasing the connection arrives at the remote endpoint [SDEW92].

XTP uses a mixture of handshaking and timer-based techniques. Data can be carried in the connection-opening packet and connection identifiers are managed such that the arrival of the packet opens the association. The use of handshakes to close associations avoids some of the buffer and timer overhead [SDEW92]. In XTP not only parameters are

negotiated and updated during the connection establishment and data transfer, but also the mode of operation is selected. This may improve the performance of the connection, assuming that chosen parameters will remain compatible.

SNR is based on the standard three way handshake similar to that of TCP/IP. It also has the overhead in short connections, caused by the exchanged messages, but the advantage of this protocol is that many parameters are negotiated between the transmitter and the receiver, as well as the selection of the mode of communication. Having the ability to initially negotiate and later modify as many parameters as possible facilitates adapting protocol processing to different environments [DOER90].

Handshake schemes used by TCP/IP and SNR are responsible for an overhead in short-lived connections, but this overhead is negligible for long-lived connections. XTP, due to its implicit connection establishment, has a clear advantage avoiding this overhead in all kinds of connection. On the other hand, XTP and SNR seem to be advantageous because of the big number of negotiated and updated parameters, as well as the initial selection of mode of operation.

## E. SIMPLICITY

Simplicity is a major subject that has to be examined. We have already discussed this factor with respect to packet format, but a high speed protocol should be as simple as possible in its algorithm and in its use. Transport protocols are used primarily by people who know very little about networks.

TCP/IP is a protocol already in use for many years. It can not be considered as an especially complex protocol to use. On the other hand, XTP seems to be more powerful, but also more complex. It allows the user to define a multicast group, to encode a 32-bit

value to indicate a message's relative priority, to choose selective retransmission or "go-back-n," to decide whether and when acknowledgments are desirable, to select the mode of operation and to select out-of-band transmission. It is questionable if all these features could properly be used by any user, otherwise some default values should be selected. Default values will make the protocol simpler, but then many features will be ignored most of the time.

One of the characteristics of SNR is its simplicity. The packet format is very simple, and the only option that a user has to select is the mode of operation.

Transport protocols should be simple. SNR seems simpler than TCP/IP and XTP. This advantage will probably make it easier to implement.

## F. SPECIAL FEATURES

In this section we discuss the special features that each protocol provides. Some of them may not be considered very important, but they should be at least mentioned.

TCP/IP defines IP multicasting addressing, maintains a cache of previously acquired IP-to physical addresses in order to reduce communication costs, and allows the sender to specify data as urgent (out-of-band data). An additional feature is the possibility to indicate low delay, high throughput or high reliability. The disadvantage is that internet does not guarantee the type of transport requested. It is only a hint to the routing algorithm.

XTP provides a powerful mechanism for reliable group communication, with the possibility for the transmitter to request flow and control information from the receiver group. It uses a form of address translation through a hashing function increasing network's efficiency, and provides a small channel for out-of band data. A 32-bit sort field

is included to assign different priorities. Over four billions priority levels may be selected. The problem is if this huge amount of selections will be properly chosen in order to provide the correct message's relative priority. Finally we need to mention rate and burst control, discussed in a previous section, as an extremely important feature of XTP.

A special feature of SNR is the use of counter variables instead of explicit timers. This method does not require a timer to be maintained associated with each data packet, and variations of the round trip delay are automatically reflected onto the retransmission timeout values. Once an average estimate of the RTD is obtained, the protocol naturally adjusts the retransmission frequency in a very simple way [TIPI93]. Parallel processing is also very important. SNR is composed of eight machines each of which perform a specific function. These machines operate almost independently leading the protocol to parallel implementation. This contributes significantly to the improvement of the throughput performance [TIPI93]. Different modes of operation let the user select the most appropriate, depending the various implementations.

All three protocols have many special features. In this section we mentioned some of the most important, emphasizing the advantages or disadvantages, but it is beyond the scope of this thesis to discuss in more details all the different special features of each transport protocol.

## G. ECONOMICAL AND POLITICAL FACTORS

Aside from the technical issues, there are economic and political factors which must be considered. In fact, these may ultimately turn out to be more critical in determining which transport protocol will be used in the future.

TCP/IP is a firmly established protocol, which is in very wide use; in fact, is one of

the most widely used protocols in the world. It is so firmly entrenched that any replacement protocol must be an order of magnitude better. So, while we can argue that XTP and SNR are better protocols than TCP, to make the argument that they are 10 times better is a very different proposition.

As an example, consider the RS232 standard. This is a physical layer standard for communications that has been around since 1960s. Twice new and better standards have been written and accepted; yet, neither of these new physical protocols (though everyone agrees they are better) have been able to uproot RS232, in its various forms.

However, we have a different situation with TCP/IP. Major changes are taking place in world and national telecommunications. A new standard for telecommunications networking, called Asynchronous Transfer Mode (ATM) has been defined and agreed upon by telecommunication carriers the world over. Local, metropolitan and long distance communications networks are going to be much faster in the future. These will be from one to many orders of magnitude faster, and will be more flexible as well. It will be possible to bring bandwidths into the home and office at thousands of times faster than the current rates.

To make a comparison with current standards, it will be possible to have tens of thousands of telephone lines into a home, thousands of TV channels (which can be high definition TV), or much higher speed data lines for computer communications.

In short, major changes are taking place in world telecommunications, and TCP/IP in its present form is simply not adequate; it must and will change and adapt.

But the big question in this thesis is whether TCP/IP will adapt or be replaced by another high speed transport protocol such as XTP or SNR. We can only guess the answer to this question; but whatever the transport protocol of the future, we expect that it will

have some of the features of all three of the protocols we have studied.

## H. EXAMPLES OF FILE TRANSMISSION USING TCP/IP, XTP AND SNR

In order to illustrate the difference between TCP/IP, XTP and SNR, message transmissions using these three transport protocols will be presented. An experimental network, 4,000 Km long, is shown in Figure 28. The purpose is to measure the time needed to send a small file without errors, a large file without errors, and a large file with reasonably low error factor, from San Francisco to Atlanta (4,000 Km).

For these cases some facts have been considered and several assumptions have been made (TABLE 3). The speed for electromagnetic signals through a guided medium, is typically: $2 \times 10^8$ m/sec [STAL88], 2/3 the speed of light in a vacuum. The whole system operates at 500 Mbps. It is assumed that the processing time in the receiver, as well as in all the intermediate gateways and nodes, is negligible, sliding windows in all three protocols are fixed without causing any kind of problem during the transmission of the messages, the retransmission timeout is always twice the time of the round trip delay, and that there is a continuous flow of the messages from the first to the last byte.

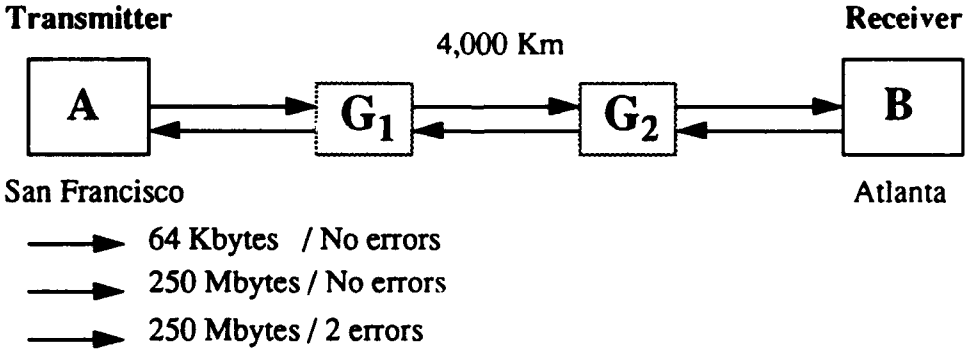



**Figure 28:** Example of Long Distance Network

The maximum data segment size for TCP/IP is 536 bytes (the default size of an IP datagram, 576, minus the standard size of IP and TCP headers) [COME91]. The propagation delay is found by the formula:

$$t = d / \upsilon = 4{,}000{,}000 \text{ m} / 2 \times 10^8 \text{ m/sec} = 0.02 \text{ sec or } 20 \text{ msec}.$$

- The time required for establishing the connection is calculated by adding three times the propagation delay, plus twice the transmission time for the 40 bytes long synchronization messages. Note that the third message of the three way handshake may transfer the first data segment, and hence is not calculated in the connection establishment overhead. Thus, the total time for establishing the connection is:

$$(40 / 65{,}536{,}000 + 20 \text{ msec}) \times 2 + 20 \text{ msec} = 60.00122 \text{ msec}$$

Considering that the packet format in XTP consists of the header (40 bytes), the trailer (4 bytes), and the middle segment (variable length), we assume that the data in each packet is 1024 bytes and the total length is 1068 bytes. The control packet is 100 bytes long and required by the receiver every 1.25 msec (as in SNR). The propagation delay is also 20 msec, and since the connection establishment is implicit, the calculated overhead is only 20 msec.

In SNR a single packet contains 1024 bytes of data. One block contains 8 packets and hence the packet format consists of: *LCI* (16 bits), *type field* (2 bits), *seq #* ($n$=3, $k$=3), data (1024 x 8), and *error check* (16 bits). Therefore, the total length of a packet is 8232 bits or 1029 bytes. The difference in the transmission of the large message of 250 Mbytes is that in this case the seq # field will be 18 bits ($n$=15, $k$=3), and hence the total packet size will be 1030.5 bytes. The propagation delay, and the connection establishment overhead will be the same as TCP/IP. In addition, control packets will be sent periodically for state information. The time interval for these transmissions is calculated from the

formula:

$$T_{in} = RTD / kou = 40 \text{ msec} / 32 = 1.25 \text{ msec}$$

The results of the total transmission time for the three different examples are shown in TABLE 4.

**TABLE 3: ASSUMPTIONS MADE FOR MESSAGE TRANSMISSIONS**

| | TCP/IP | XTP | SNR |
|---|---|---|---|
| Data segment size | 536 bytes | 1024 bytes | 1024 bytes |
| Total segment size | 576 bytes | 1068 bytes | 1029/1030.5 bytes |
| Connection establishment | 60.00122 msec | 20 msec | 60.00122 msec |
| Propagation delay | 20 msec | 20 msec | 20 msec |
| Control packet size | ----- | 100 bytes | 12 bytes |
| Control packets interval | ----- | 1.25 msec | 1.25 msec |

### 1. Small File with No Errors

In TCP/IP: The message will be divided in (64 x 1024) / 536 = 123 segments. The total transfer will be (123 x 576) = 70,848 bytes (padding included) and the total transmission time will be: 70,848 / 65,536,000 = 1.081055 msec. Adding the connection establishment overhead (propagation delay is included) the final total time is:

$$60.00122 \text{ msec} + 1.081055 \text{ msec} = \mathbf{61.082275 \text{ msec}}$$

In XTP: The message will be divided in 65,536 / 1024 = 64 packets. A packet is 1068 bytes, hence the total transfer will be (64 x 1068) = 68,352 bytes. During the transmission time only one control packet is required (100 bytes). Hence the total

transmission time will be (68,352 + 100) / 65,536,000 = 1.044495 msec. Adding the connection establishment overhead the final total time is:

20 msec + 1.044495 msec = **21.044495 msec**

In SNR: The message will be divided in 65,536 / 1024 = 64 packets. A packet is 1029 bytes, hence the total transfer will be (64 x 1029) = 65,868 bytes. During the transmission time only one control packet is required (12 bytes). Hence the total transmission time will be (65,868 + 12) / 65,536,000 = 1.005066 msec. Adding the connection establishment overhead the final total time is:

60.00122 msec + 1.005066 msec = **61.006286 msec**

### 2. Large File with No Errors

In TCP/IP: The message will be divided in (250 x $1024^2$) / 536 = 489,075 segments. The total transfer will be 489,075 x 576 = 281,707,200 bytes (padding included) and the total transmission time will be 281,707,200 / 65,536,000 = 4.298510742 sec. Adding the connection establishment overhead the final total time is:

4.298511 sec + 0.06000122 sec = **4.358511962 sec**

In XTP: The message will be divided in (250 x $1024^2$) / 1024 = 256,000 packets. A packet is 1068 bytes, hence the total transfer will be (256,000 x 1068) = 273,408,000 bytes. During the transmission time 3,338 control packets are required (every 1.25 msec or every 81,920 bytes). Hence the total transmission time will be (273,408,000 + 3,338,000) / 65,536,000 = 4.222808838 sec. Adding the connection establishment overhead the final total time is:

4.222808838 sec + 0.02 sec = **4.242808838 sec**

In SNR: The message will be divided in (250 x $1024^2$) / 1024 = 256,000 packets or 256,000 / 8 = 32,000 blocks. A packet is 1030.5 bytes, hence the total transfer will be

(256,000 x 1030.5) = 263,808,000 bytes. During the transmission time 3221 control packets are required (13.5 bytes each). Hence the total transmission time will be (263,808,000 + 43,483.5) / 65,536,000 = 4.026054131 sec. Adding the connection establishment overhead the final total time is:

4.026054131 sec + 0.06000122 sec = **4.086055351 sec**

### 3. Large File with Two Errors

In TCP/IP: The total length of this message is 281,707,200 bytes. When the first 100,020 segments will be sent an error will occur. Hence only (100,020 x 576) = 57,611,520 bytes will be received correctly. The transmission time for this portion of the message will be 57,611,520 / 65,536,000 = 0.879082031 sec. The transmitter will continue to send the file until a timeout occurs for the undelivered segment and all the segments after the error will be sent again ("go-back-n"). Only 100,019 segments have been received correctly, so after the first timeout (489,075 - 100,019) = 389,056 segments have to be sent. A second error will occur after 205,843 segments. The transmission time will be 118,565,568 / 65,536,000 = 1.809166992 sec. After a second timeout the transmitter will send the rest of the file: 105,531,264 / 65,536,000 = 1.610279297 sec. Finally, the total transmission time of the file is 0.879082031 + 1.809166992 + 1.610279297 + connection establishment + (2 x timeout) =

= 4.29852832 sec + 0.06000122 sec + 0.08 sec = **4.43852954 sec**

In XTP: The total length of the message is 276,746,000 bytes. Since selective repeat is used in XTP, two packets only will be retransmitted. This makes the total length of the file 2,136 bytes longer. The final length will be 276,748,136 and the total transmission time will be: 276,748,136 / 65,536,000 + propagation delay =

= 4.222841431 sec + 0.02 sec = **4.242841431 sec**

In SNR: The total length of the message is 263,851,483.5 bytes. For the two errors only two blocks will be retransmitted. That is, 16 packets or 16,488 bytes. The total length will be 263,851,483.5 + 16,488 = 263,867,971.5 bytes, and the total transmission time will be: 263,867,971.5 / 65,536,000 + connection establishment =

= 4.026305717 sec + 0.06000122 sec = **4.086306937 sec**

**TABLE 4: TRANSMISSION TIME FOR DIFFERENT FILE TRANSFERS**

| Kind of File Transmission | TCP/IP | XTP | SNR |
|---|---|---|---|
| Small File with No Errors | 61.082275 msec | 21.044495 msec | 61.006286 msec |
| Large File with No Errors | 4,358.511962 msec | 4,242.808838 msec | 4,086.055351 msec |
| Large File with Two Errors | 4,438.52954 msec | 4,242.841431 msec | 4,086.306937 msec |

### 4. Comments on the Results

The results of these examples (TABLE 4) are not very accurate, since many assumptions were made. However, from these results, we observe strengths and weaknesses of the three protocols.

We can see that XTP has an advantage during the connection establishment with its implicit handshake, especially for short transfers. This is normal, since the propagation delay is added to the transmission time of the message.

Secondly we note the big advantage of the selective repeat method. Especially at high speed (500 Mbps or more) this is a very strong advantage. Correctly received segments do not have to be retransmitted in case of errors as happens in "go-back-n."

The format of the packets/segments is another important fact we have to take

into consideration. We can see, from the large file example, that SNR has a better performance than XTP, due to the simpler packet format it uses, and the smaller number of bytes it sends for the header, trailer etc.

The assumption that processing time is negligible favors TCP/IP which has not the possibility for parallel processing.

An assumption made for these examples is that sliding windows are fixed, from the very first segment sent. To understand the reason of this assumption, and how the sliding window influences the performance of TCP/IP we will give an additional example.

Assume the round trip delay is 40 msec (4,000 Km network), and the link has an available bandwidth of 0.5 gigabit. If the packet size used is 576 bytes, the correct window size for the channel is (65,536,000 x 0.04) / 576 or about 4,552 packets. If the window size is growing linearly from 1, it will take 4,552 round trips or 182 seconds or 3 minutes to get the correct window size. If the packet size is large (16 Kbytes), the window size is about 160 packets, which takes approximately 6.5 seconds.

In fact TCP/IP uses the slow-start algorithm which is not exactly linear. TCP grows exponentially up to one-half the previous window and then grow linearly toward the gigabit bandwidth of the link. Unfortunately, linear growth takes a long time, and on a long distance link the window size will not reach the size appropriate for gigabits speeds for several seconds or minutes. [PART94]

# VII. CONCLUSION

## A. SUMMARY OF RESEARCH

In this thesis we discussed most of the transport protocols services. We introduced TCP/IP as a widely known transport layer protocol. We also introduced XTP and SNR as two new experimental lightweight transport protocols. In Chapter VI, we compared these protocols attempting to determine if there is a need to change. TABLE 5 summarizes the results of our study.

**TABLE 5: COMPARISON OF PROTOCOL MECHANISMS AND SERVICES**

| | TCP/IP | XTP | SNR |
|---|---|---|---|
| Flow Control mechanism | Window | Window | Window |
| Rate Control mechanism | ——— | Rate and burst control | ——— |
| Disable flow/ error control | ——— | No-flow mode No error mode | No-flow mode No-error mode |
| Error control retransmission policy | PAR Go-back-n | ARQ, Go-back-n, Selective | ARQ Selective |
| Error recovery acknowledgments | Based on data reception | Based on sender's selection | Sender independent |
| Packet format size | Variable | Fixed | Fixed |
| Packet format simplicity | Complex | Complex | Simple |
| Checksum | Before data | After data | After data |

| | TCP/IP | XTP | SNR |
|---|---|---|---|
| Connection Establishment | Handshake | Implicit | Handshake |
| Connection termination | Handshake | Handshake | Handshake |
| Minimal number of packets | 3 | 2 | 3 |
| Simplicity | ----- | Complex | Simple |

TCP/IP provides a reliable interprocess communication using positive acknowledgments and timer based retransmissions. A three way handshake establishes and terminates a connection, and the protocol uses a sliding window to handle efficient transmission and flow control. It is a firmly established protocol, but in its present form, TCP/IP is not capable to follow the evolutions that are taking place in the field of telecommunications.

XTP provides a reliable transport service using selective acknowledgment and selective retransmission. It offers a fast connection setup, flow, error, rate and burst control. XTP provides multiple communications mechanisms, but the user is responsible to select those that are desired. It is a new experimental lightweight transport protocol designed to replace TCP/IP and offer faster and more flexible services.

The last protocol described in this thesis was SNR. It is also a reliable protocol, using selective retransmission. A three way handshake establishes and terminates a connection, and the protocol provides error recovery, sequenced delivery and flow control. Reduction of the processing overhead and utilization of parallel processing provide a high processing speed. SNR is also an experimental high speed transport protocol designed in the AT&T laboratories, in order to provide high throughput consistent with the evolving high speed

physical networks based on fiber optic transmission lines.

As a final comment, we have to notice that many researchers have already proposed extensions for TCP in order to improve its performance as in [JACO88], [JACO90], [JACO92]. It is our conviction, that in the near future, one of the new experimental lightweight transport protocols will replace TCP/IP. There is a need for a considerable amount of work on high performance protocols, but there is also a limited time of life for TCP. Extensions will not always succeed to qualify TCP/IP as a transport layer protocol for high speed networks operating at 20 or more Gb/sec. SNR seems to be in a better position.

## B. FUTURE RECOMMENDATIONS

Software implementation of SNR and XTP in the same operating environment is the continuation of this thesis. Performance results obtained for transport protocol implementation and pragmatic situations should be compared. Extensions of TCP/IP should also be implemented in order to realize the improvements made to the current implemented transport protocol.

Although we emphasized the comparison of the three protocols, we strongly recommend a comparison to other high speed transport protocols, too. VMTP, NETBLT and Delta-t appear as good candidates also.

# LIST OF REFERENCES


[CHER88] D. Cheriton, "VMTP: Versatile Message Transaction Protocol, Protocol Specification, RFC-1045," *Internet Request for Comments*, No. 1045, Stanford University, Feb. 1988.

[CHER89] D. Cheriton, C. Williamson, "VMTP as the Transport Layer for High Performance Distributed Systems," *IEEE Commun. Mag.*, Vol. 27, Jun. 1989.

[CLAZ87] D. D. Clark, M. L. Lambert, L. Zhang, "NETBLT: A Bulk Data Transfer Protocol; RFC-998," *Internet Request for Comments*, No. 998, MIT, Mar. 1987.

[COME91] D. E. Comer, *Internetworking with TCP/IP*, Vol. I and Vol. II, Prentice Hall, 1991.

[DOER90] W. A. Doeringer, D. Dykeman, M. Kaiserswerth, B. W. Meister, H. Rudin, R. Williamson, "A Survey of Light-Weight Transport Protocols for High-Speed Networks," *IEEE Transactions on Communications*, Vol. 38, No. 11, Nov. 1990.

[ISIC81] Information Sciences Institute, "Transmission Control Protocol, Protocol Specification; RFC-793", *Internet Request for Comments*, No. 793, University of Southern California, Sep. 1981.

[JACO88] V. Jacobson, R. Braden, "TCP Extensions for Long-Delay Paths; RFC-1072," *Internet Request for Comments*, No. 1072, Information Sciences Institute, Oct. 1988.

[JACO90] V. Jacobson, R. Braden, "TCP Extension for High-Speed Paths; RFC-1185," *Internet Request for Comments*, No. 1185, Information Sciences Institute, Oct. 1990.

[JACO92] V. Jacobson, R. Braden, "TCP Extensions for High Performance; RFC-1323," *Internet Request for Comments*, No. 1323, Information Sciences Institute, May 1992.

[MCAR92] R. C. McArthur, "Design and Specification of a High Speed Transport Protocol," M.S. Thesis, U.S. Naval Postgraduate School, Mar. 1992.

[MINO91] D. Minoli, *Telecommunications Technology Handbook*, Artech House, 1991.

[LUND92] R.C. McArthur, G.M. Lundy, "Design of High Speed Transport Protocols," U.S. Naval Postgraduate School, 1992.

[LUND93] G.M. Lundy, H.A. Tipici, "Specification and Analysis of a High Speed Transport Protocol," Submitted for publication, U.S. Naval Postgraduate School, 1993.

[PART94] C. Partridge, *Gigabit Networking*, Addison-Wesley Publishing Company Inc., 1994.

[POST80] J. Postel, "User Datagram Protocol; RFC-768," *Internet Request for Comments*, No. 768, Information Sciences Institute, Aug. 1980.

[SABN90] A. N. Netravali, W. D. Roome, K. Sabnani, "Design and Implementation of a High-Speed Transport Protocol," *IEEE Transactions on Communications*, Vol.38, No.11, Nov. 1990.

[SDEW92] W. T. Strayer, B. J. Dempsey, A. C. Weaver, *XTP: The Xpress Transfer Protocol*, Addison-Wesley Publishing Company Inc., 1992.

[STAL88] W. Stallings, *Data and Computer Communications*, Macmillian Publishing Company, 1988.

[TIPI93] H. A. Tipici, "Specification and Analysis of a High Speed Transport Protocol," M.S. Thesis, U.S. Naval Postgraduate School, May 1993.

[WEAV92] A. C. Weaver, "High Speed Communication for Distributed Applications," *IEEE International Workshop on Emerging Technologies and Factory Automation Technology for the Intelligent Factory*, University of Virginia, Aug. 1992.

[ZHAN86] L. Zhang, "Why TCP Timers Don't Work Well," *Proc. ACM SIGCOMM Symposium on Communications, Architectures, and Protocols*, Stowe, VT., 1986.

# INITIAL DISTRIBUTION LIST

| | |
|---|---|
| Defense Technical Information Center<br>Cameron Station<br>Alexandria, VA 22304 - 6145 | 2 |
| Dudley Knox Library<br>Code 52<br>Naval Postgraduate School<br>Monterey, CA 93943 | 2 |
| Dr. G.M. Lundy, Code CS/LN<br>Computer Science Department<br>Naval Postgraduate School<br>Monterey, CA 93943 | 2 |
| Prof. L.D. Stevens, Code CS/ST<br>Computer Science Department<br>Naval Postgraduate School<br>Monterey, CA 93943 | 1 |
| Embassy Of Greece<br>Naval Attache<br>2228 Massachusetts Avenue, NW<br>Washington, DC 20008 | 2 |
| Konstantinos A. Lazaris<br>Apostolopoulou 57<br>Halandri 15231<br>Athens, GREECE | 3 |