High Performance Transports: HTPNET, TCP/IP, XTP, TP++ and RTP

By Zahid Hossain hossain@cis.ohio-state.edu

Abstract

The high transmission speed of optical networks has resulted in shifting of communication bottlenecks from the transmission media to the host processing system, especially the transport protocol processing at the end systems. New protocols, such as XTP, HTPNET, RTP, have been proposed by researchers to meet the requirements of high speed networks. TCP/IP, which has been very successful as a transport protocol for the last twenty years, is also adapting itself in high-speed network.

Table of Contents:

---

1 Introduction

Computer networking technology has seen enormous growth over the past thirty years. In the 1960's "networking" was synonymous with the telephone network, and communication speed of hundreds of bits per second was common when using telephone modems; packet radio technology increased the data communications speed to thousands of bits per second. A new concept was introduced in 1970s: the Local Area Network(LAN). In particular, Ethernet brought transmission speed of 10 million bits per second (Mb/s). Again, in the 1980s, MAN (Metropolitan Area Network) and LAN speeds increased to the order of 100 million bits per second, and now we are having WAN (Wide Area Network) with gigabit per second (Gb/s) networks using fiber optics. The high transmission speed of optical networks has resulted in shifting of communication bottlenecks from the transmission media to the host processing system. TCP transport protocol has been very successful for last twenty years, but the rapid advancement of fiber optics technology and the demand for multimedia network services have accelerated the development of next generation of lightwave networks. This lightwave networks are able to operate at a far greater bandwidth in the multi-mega bits range, while offering low error rate performance. Such operating speed, however, has not been matched by a corresponding increased in transport protocol processing at the end systems. As the computer networking is changing, new protocols are being proposed to meet the requirements of these fast networks. HTPNET(High-speed Transpot Protocol for Net Works), XTP(Xpress Transport Protocol), RTP(Rapid Transport Protocol) and TP++ have been proposed to meet the requirements of future networks. TCP/IP has also been modified to

meet the requirements of future networks. In this paper I have discussed High Performance Transport protocols, their performances, and their future.

2 A High Performance Transport Protocol (HTPNET)

HTPNET or High-performance Transport Protocol system is designed to overcome protocol processing bottleneck. It is designed based on a highly parallel architecture and is designed to exploit the evolving characteristics of high speed networks. HTPNET uses an out-of-band signaling system based upon transmitter-paced periodic exchanges of state information between end systems. This mechanism exhibits several attractive properties which have been demonstrated to perform efficiently in a simulated high-speed environment with high bandwidth delay product. A formal validation on the mechanism has been performed using the SPIN [4] validation tool. The analysis has resulted in the uncovering of several properties of the protocol that have been overlooked during the initial design phase. An experimental implementation of HTPNET has been constructed from a network of T800 transputers. The results demonstrated the advantages of exploiting a parallel architecture for protocol processing. The results obtained from the experiment prompted further exploitation of the parallel architecture of HTPNET to provide a suitable platform for the parallel implementation of Presentation processing, which can incur high computation overheads. The protocol processing system is designed to be implemented on the T9000 transputer and C104 router technology. A simulation of the architecture has demonstrated the feasibility of applying functional and packet parallelisms to protocol processing to match the ever increasing processing demand of future high-speed networks. In this section we will discuss different aspects of HTPNET.

HTPNET (High-speed Transport Protocol for Networks) is based on a highly parallel architecture. It also take into consideration the communication network characteristics, namely the connection oriented nature of broadband networks, increased bandwidth-delay-product and lower error rates. HTPNET is currently implemented on a network of a T800 transputers. Experiments have revealed promising packets transfer rate of 12,000 to 13,000 packets/sec with a packet size of 2 Kilobytes. As a result, a data transfer rate of more than 200 Mb/s could be achieved [5].

2.1 HTPNET Overview

The HTPNET aims to decuple as far as possible protocol processing from the main operating system using an additional multiprocessor-based subsystem. In addition, the structure of the HTPNET protocol is based on several finite state machines which are designed to exploit a parallel implementation and achieve high throughput.

2.1.1 Packet Format: HTPNET consists of eight types of packets [5]:

- **Connection Request (CR)**

- **Connection Confirm (CC)**

- **Disconnect Request (DR)**

- **Disconnect Confirm (DC)**

- **Reject (RJ)**

- **Data (DT)**

- **Transmitter State (TXS)**

- **Receiver State (RXS)**

To setup a connection, a three-way handshake protocol is used to establish a connection between two end points similar to that of TCP. This setup allows the endpoints to negotiate communication parameters options, including protocol class, available buffer (flow control) and inter-packet gap (rate control). HTPNET is designed with eight variations of protocol classes to provide different quality of services (QOS). Table 1 defines the functions available different protocol classes. [5]

Class	Error Control	Flow Control	Rate Control
0	No	No	No
1	No	No	Yes
2	No	Yes	No
3	No	Yes	Yes
4	Yes	No	No
5	Yes	No	Yes
6	Yes	Yes	No
7	Yes	Yes	Yes

Table 1: Protocol Class Definition for HTPNET

Class 0 assumes a perfect networking with all control functions disabled. On the other extreme, Class 7 assumes a highly unreliable network and consequently, requires the enabling all control functions. This diversity of available protocol classes provide host with the flexibility to select QOS based on application requirements and the underlying network. Although protocol class is negotiated during setup time, a connection may dynamically re-negotiate the class to allow adaptive protocol processing based on changing network environments. To release a connection, HTPNET uses a symmetric three way handshake protocol used in establishing a connection. To avoid any data loss upon disconnection, a graceful close mechanism is used to ensure that as the connection moves from data transfer to termination, all data intended for the corresponding end points are received.

2.2 Design issues on error control Mechanism for High speed networks (HTPNET)

The changes in the characteristics of high-speed networks have a major impact on the design of a efficient, high performance error control protocol, TRAPS [3], which is designed to exploit the evolving characteristics of future high-speed networks.

2.2.1 State Synchronization and Error Recovery

To support a parallel architecture, HTPNET employs out-of-band signalling, in which peer-transport entities exchange data and state information using separate packets. A state information packet is exchanged frequently so that the state between the transmitter and receiver is synchronized. A unique feature of HTPNET is transmitter paced periodic exchange of state information between the two end systems. SNR [5] protocol has implemented similar idea, except that HTPNET utilizes the transmitter mode to control the periodic exchange of state information and SNR exchanges state information independently between two end systems. [3,4]

2.2.2 Selective retransmission

In HTPNET, when the receiver periodically receives a state packet from the sender, it coveys the full status of the data packets received by supplying a cumulative acknowledgment (CACK), as well as range information (RANGE) on the packets received error-free. The cumulative

acknowledgment with sequence number N indicates that those packets with sequence number up to N have been successfully received. To convey acknowledgment of packets beyond sequence N, range information in a set of pair (bseqn,eseqn, n represent the nth pair) is used. [3,4,5]

2.3 Congestion Avoidance

The congestion control scheme of HTPNET is based on periodic traffic intensity feedback(PTIF) reactive control. The congestion control is seen as a two-level strategy operating on short and medium time-scale[5]. The control state packets are transmitted periodically, with a period of a. If we assume that the congestion node, x packets are queued between two successive control packets c(n) and c(n+1), then the instantaneous arrival rate can be expressed as:

$$I(n+1) = x/a$$

2.4 Architecture of HTPNET

As oppose to the conventional vertical structuring of most protocols, HTPNET is horizontally structured. The rationale for horizontal structure of HTPNET is based on the division of the protocol into separate loosely coupled management functions. These functions are essentially independent of each other, and can be executed in different processors in parallel with minimal inter-communications between the functions. HTPNET is divided into four main processing functions [1,5]:

○ **Transmitter data process**

○ **Receiver data process**

○ **Transmitter control process**

○ **Receiver control process**

The segmentation of HTPNET into distinct horizontal functions allows the data packet processors to dedicate its computing power to handle data packets. Figure 1 shows a HTPNET basic protocol Architecture [1].
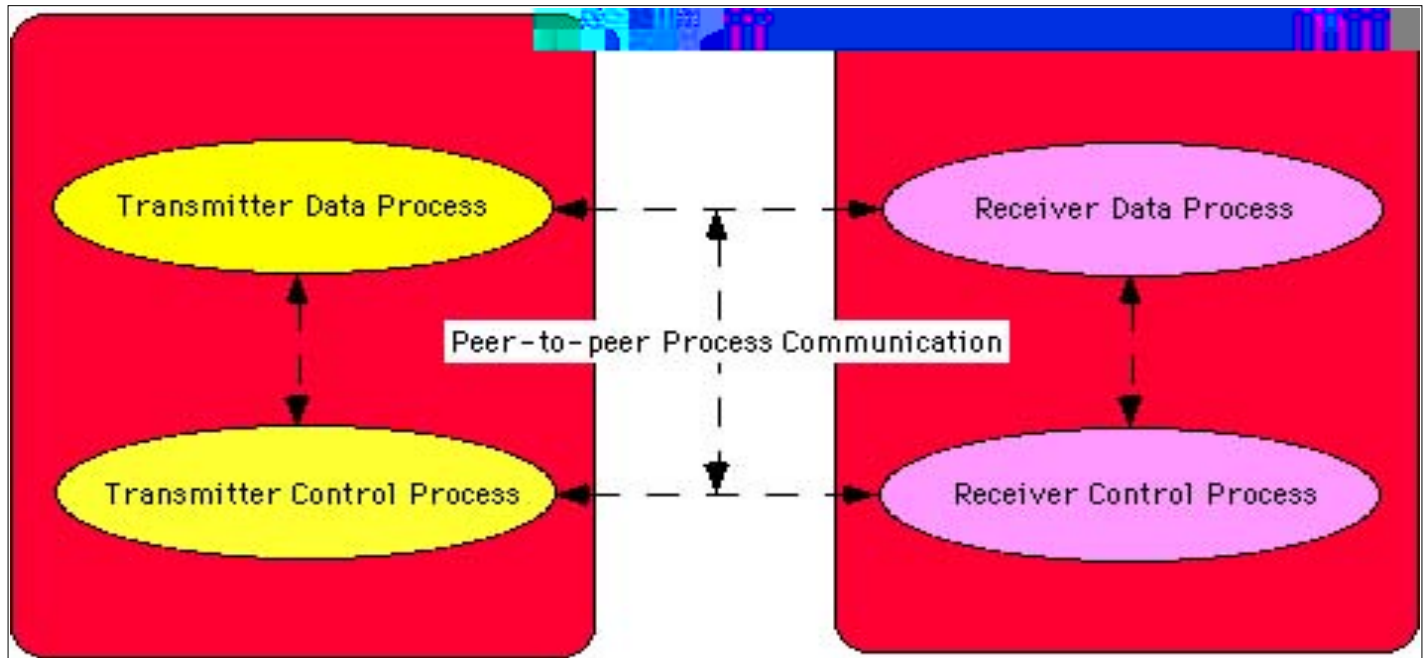
Figure 1 HTPNET Basic protocol Architecture

2.5 Performance of HTPNET

In an experimental implementation model [5] of HTPNET corresponding performance results under different coalifications from 1 to 8 transputer shows close to five-fold increase in throughput for configurations ranging from one transputer to eight transputers. These results clearly demonstrate the advantage of exploiting a parallel architecture in the design of HTPNET. With a configuration of five transputers for protocol processing, a throughput performance of about 12,000 packets/s is attained. Assuming data packets of 2 Kbytes length this translate into an achievable data transfer rate of more than 200 Mb/s.

3 High Performance TCP/IP (On Gigabit Networks)

Some researchers predicted in 1987 that TCP/IP would never run at 100 megabits per second (Mb/s). It was thought that TCP was too complex and used up too much of CPU to be capable of operating efficiently at FDDI speed, not to speak of gigabit per second (Gb/s) speeds. New

"Lightweight protocols" were advocated for high speed networks which is simpler than TCP. These new transport protocols should be simple enough to be implemented in hardware and so would operate more efficiently on high speed links.

But the study by TCP/IP proponents showed that TCP/IP was indeed capable of per forming at high speeds [8]. This work on high-performance TCP/IP examined both implementation techniques and protocol issues. At present, "some TCP implementation can run at gigabit per second rate, proving that speed is not the issue for TCP/IP on gigabit networks", according to Stephen Pink of Swedish Institute of Computer Science. The new challenge for TCP/IP is support for a new generation of real time multimedia applications.

3.1 Performance on Local Area Networks

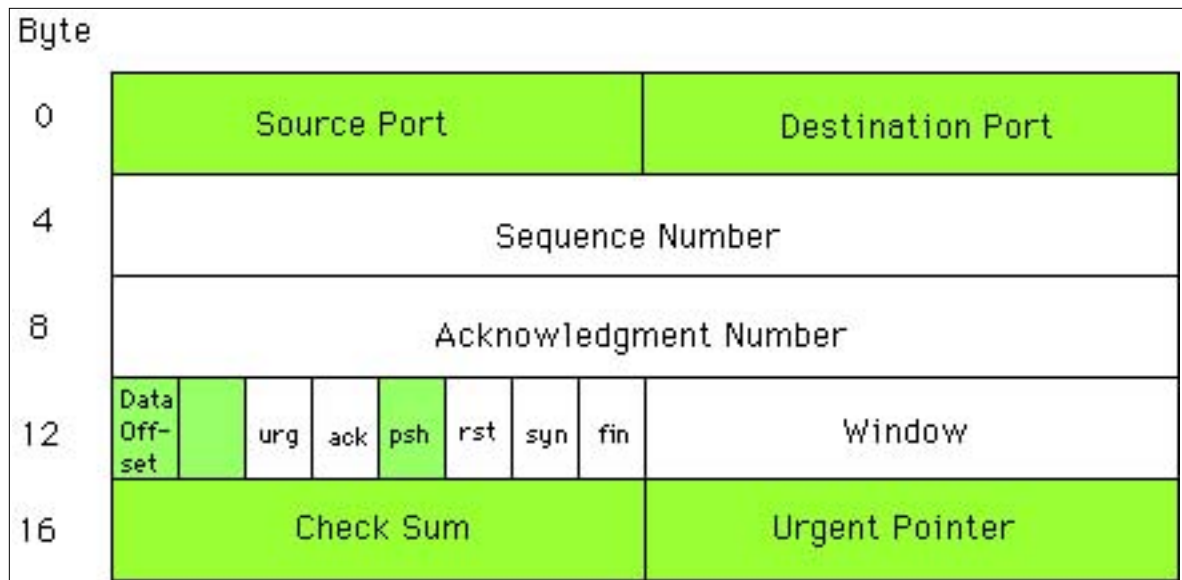## 3.1.1 TCP common path optimizations

TCP is a complex protocol judged from the number of lines of code in a typical implementation. The TCP in a 4.3 BSD Reno has about 3,900 lines of C code. At runtime, however, very little of this code executes frequently. Much of the code handles errors and other special cases. For example, in the case of a file transfer between two peers running TCP, the common path is the code that execute [8]:

❍ **after a connection is established,**

❍ **when the packets carry the data (as oppose to being acknowledgments) and have no control flags set in the header,**

❍ **and when the data is in sequence.**

3.1.2 TCP Header prediction

4.3 BSD Reno and later versions (Berkeley Unix OS) has an important optimization by Van Jacobson called "Header prediction." This is a good example of a successful optimization technique that significantly reduces the cost of per-packet TCP processing. During a TCP coversation, after connection establishment and before connection close, most values in the fields of TCP headers don't change. When the application that uses TCP is a bulk data transfer, even fewer fields change during the transfer. Thus, based on a test on few fields in the header, we can determine if a packet needs special processing, and if it does not then we can forward the data directly to the user.

Figure 2 The TCP Header. Fields in white are checked in the header prediction algorithm. [6]

3.1.3 Other pre-packet optimization

- ## Minimize pcblookup:

  Common path processing occurs after TCP control structures have been locked up. Looking up the protocol control block, called pcblookup, [8] has a significant cost that is incurred pre-packet. One way of reducing the total cost that is incurred pre-packet. One way of reducing the total cost of pcblookup is over a TCP convention is to do it less often than once for each packet. If one assumes that the next packet will be from the same source as the previous packet then the last pcb can be chached. The cache can be checked for a match before preceding with more expensive pcblookup routine. Caching is used in some recent versions of Berkeley TCP and UDP

- ## Reducing number of times a processor has to raise it's priority level during pro tocol processing:

  The idea is to lock the data structures while they are being updated by setting the processor priority level high enough that code processing the new incoming packets does not have access to the data structure.

3.1.4 Pre-byte optimizations

Checksumming and coping costs are incurred per-byte and header processing is a cost that is incurred per packet. Optimizations for per-byte costs are dependent on operating system environments and processor architectures. The experiments [8] that were conducted on Berkeley TCP, showed that with proper coding, the main bottleneck in the common processing path is the memory speed. This means that an efficient implementation will touch the data as few times as possible.

3.2 Performance of TCP on the Internet

The Internet is a large federation of LANs and WANs connected by IP gateways. Here we will discuss one of the most important issues; Congestion control.

3.2.1 Congestion Control

In 1988 Van Jacobson [8] showed that TCP implementations could be easily modified to achieve equilibrium and, once there, avoid losing it. The two most important algorithms presented in his paper "Congestion Avoidance and Control" are called slow start and congestion avoidance [8].

3.4 TCP on Long fast Networks

TCP's sliding window has a maximum size of 64 Kbytes. This limits the performance of TCP on networks with high bandwidth and high delay. If the round trip time of the network is high enough, a sending TCP can fill a receiving TCP's window before an acknowledgment returns to push the window open. Bandwidth on such a network will be wasted. Even at T3 speeds, assuming a 30 ms round-trip time (coast to coast for the US), TCP needs a window size of 164 Kbytes to use the bandwidth effectively. For cross country gigabit networks, TCP needs windows well over 3 megabytes in size [8].

4 XTP: Xpress Transport Protocol

TCP/IP has been terrifically successful through 20 years of Internet growth an change. TCP's success, however, has never stopped researchers from critical examination of how TCP provides transport services, and what services TCP fails to provide. Delta-t (for connection management),

NetBLT (for bulk data transfer), VMTP (for transactions), andothers are the result of fixing some deficiency in TCP. The Xpress Transport Protocol joins this list with four contributions: orthogonal protocol functions for separation of paradigm from policy, separation of rate and flow control, explicit first-class support for multicast, and data delivery service independence.

4.1 XTP Features

4.1.1 Separation of paradigm and policy

At the core of XTP is a set of mechanisms whose functionality is orthogonal[12]. The most notable effect of this is that XTP clearly separates communication paradigm (datagram, virtual circuit, transaction, etc.) from the error control policy employed (fully reliable through uncorrected). Further, flow and rate control as well as error control can be tailored to the communication at hand. If desired, any set of these control procedures can be turned off [16].

4.1.2 Separation of rate and flow control

Flow control operates on end-to-end buffer space. Rate control is a producer/consumer concept that considers processor speed and congestion. TCP does not provide rate control, and combats congestion with slow-start and other heuristics. XTP provides mechanisms for shaping rate control and flow control independently [12,16].

4.1.3 Explicit multicast support

Multicast is not an afterthought in XTP. Rather, each mechanism used for unicast communications is available for multicast use as well. The number of communicants is orthogonal to paradigm and policy [12,16].

4.1.4 Data delivery service independence

XTP is a transport protocol, yet with the advent of switched networks rather than routed internetworks, a traditional network layer service may not be appropriate in every instance. XTP requires only that the underlying data delivery service provides framing and delivery of packets from one XTP-equipped host to another. This could be raw MAC or IP or AAL5. XTP also employs parametric addressing, allowing packets to be addressed with any one of several standard addressing formats[12,16].

4.1.5 Other features of XTP

- **implicit fast connection setup for virtual circuit paradigm**

- **key-based addressing lookups**

- **message priority and scheduling**

- **support for encapsulated and convergence protocols**

- **selective retransmission and acknowledgment**

- **fixed-size 64-bit aligned frame design**

4.2 XTP Protocol Concepts

XTP defines the mechanisms necessary for delivering user data from one end system to one or more other end systems. Each XTP implementation maintains the state of each of its communications. Well-defined packet structures, containing user data or control information, are exchanged in order to effect the user data transfer. The control information is used to provide the requested level of correctness and to assist in making the transfer efficient. Assurance of correctness is done via error control algorithms and maintenance of a connection state machine. Flow and rate control algorithms, certain protocol modes, and traffic shaping information are used to provide the requested quality of service as efficiently as possible.

The collection of information comprising the XTP state at an end system is called a context. This information represents one instance of an active communication between two (or more) XTP endpoints. A context must be created, or instantiated, before sending or receiving XTP packets. There may be many active contexts at an end system, one for each active conversation or message.

Each context manages both an outgoing data stream and an incoming data stream. A data stream is an arbitrary length string of sequenced bytes, where each byte is represented by a sequence number. The aggregate of a pair of active contexts and the data streams between them is called an association.

4.3 The State Machine

A context at an end system is initially in a quiescent state. A user in need of communication services requests that the context be placed into the listening state. The context now listens for an appropriate FIRST packet. The FIRST packet is the initial packet of an association. It contains explicit addressing information. The user must provide all of thenecessary information for XTP to match an incoming FIRST packet with the listening context. At another end system a user requests communication service from XTP. Since this user will initiate the association, the context moves from a quiescent state to an active state directly. The active context constructs a FIRSTpacket, complete with explicit addressing information gotten from the user. The FIRST packet is sent via the underlying data delivery service. When the FIRST packet is received at the first host's XTP implementation, the address is compared against all listening contexts. If a match is found, the listening context moves to the active state. From this point forward an association is established, and communication can be completely symmetric since there are two data streams, one in each direction, in an association. Also, no other packet during the lifetime of the association will carry explicit addressing information. Rather, a unique "key" is carried in each packet that maps the packet to the appropriate context. Once all data from one user has been sent, that data stream from that user's context can be closed. Sentinels in the form of options bits in a packet are exchanged to gracefully close the connection. Other forms of less graceful closings are possible by abbreviating this exchange. When both users are done, and both data streams closed, the contexts move into the inactive state. One of the contexts will send a sentinel that causes the association to dissolve. At this point, both contexts return to the quiescent state.

4.4 Common Headers

All of XTP's packet types use a common header structure. All of the information necessary to steer the packet's payload to the proper point of processing is carried in the header. Much of how an XTP context operates is controlled by a set of bitflags that are concentrated in one field in the packet header. Fifteen flags are defined, including bit flags to control connection shutdown, bitflags to control the acknowledgment policy, and bitflags that are markers in the data stream. The remaining bitflags control the end-to-end operating modes. Examples include enabling or disabling error control or flow control, or enabling multicast mode.

4.5 Control Algorithms

XTP flow control is based on 64-bit sequence numbers and a 64-bit sliding window. XTP also provides rate control whereby an end system or intermediate system can specify the maximum bandwidth and burst size it will accept on a connection. A Traffic Segment provides a means for specifying the shape of the traffic so that both end systems and intermediate systems can manage their resources and facilitate service quality guarantees.
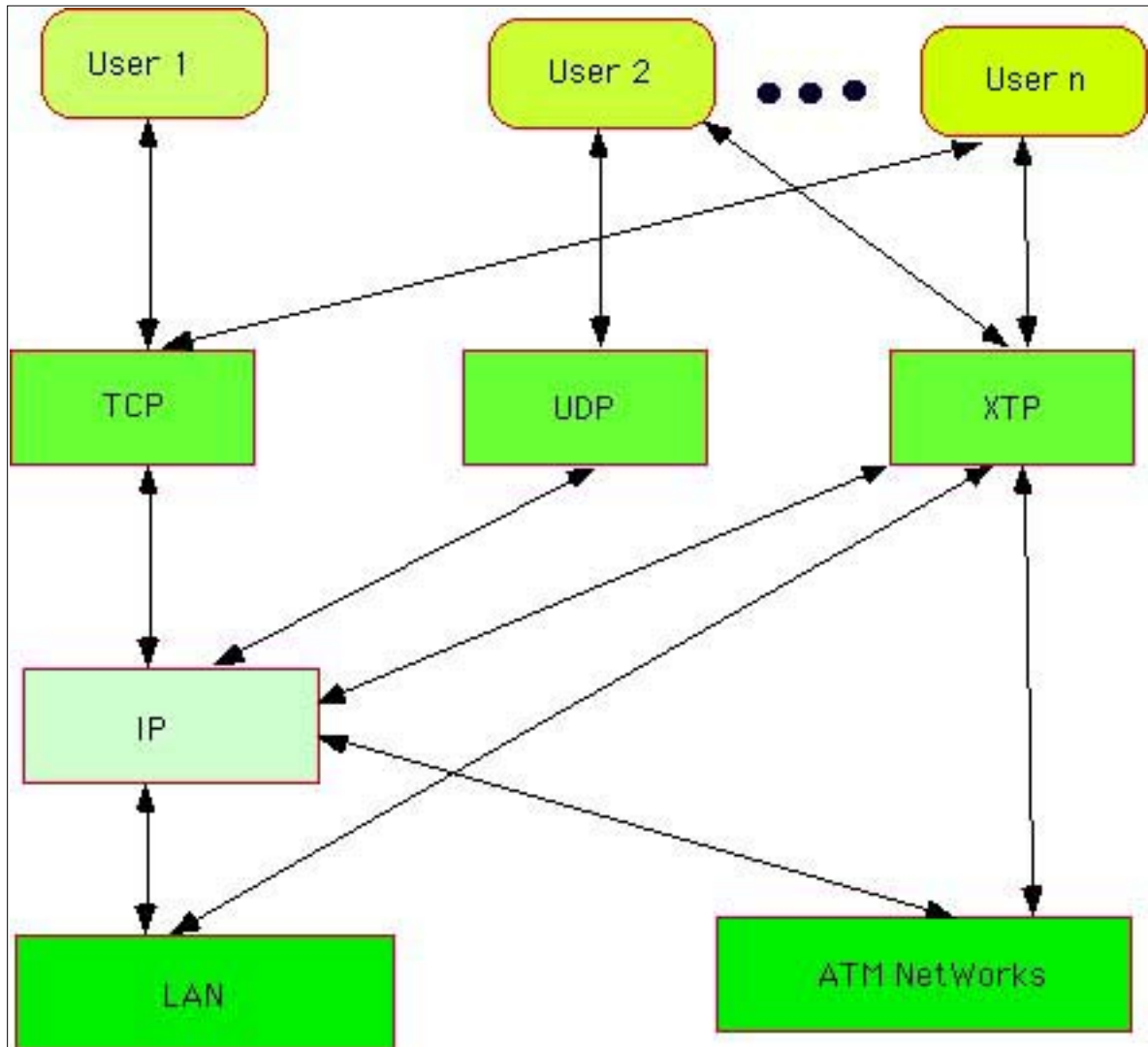
Figure 3 Possible Protocol Interconnections [19]

Error control in XTP incorporates positive and, when appropriate, negative acknowledg ment to effect retransmission of missing or damaged data packets. Retransmission may be either go-back-N or selective. The retransmission algorithms are conservative: only data that is shown to be missing via control messages may be transmitted. This avoids spurious and possible congestion-causing retransmissions. The error control algorithm, while basically conservative, can also be aggressive: a method for a quick-acting error notification is provided.

XTP also specifies techniques for extending error control to a multicast environment. The error control algorithm in multicast is identical to the unicast algorithm, although additional sophistication is required to manage state variables and achieve continuous streaming.

5 TP++: A research project on Design issues of high-performance and multimedia protocols

The TP++ is a research project at Bellcore [8] to explore the transport protocol design issues for high-performance and multimedia protocols. This project explores the transport protocol designed by some researchers as part of the Aurora project [9].

The TP++ transport protocol is designed for heterogeneous internetworks with a large bandwidth-delay product. TP++ messages are easily converted to different packet sizes without a separate fragmentation protocol. We assume that the network provide congestion control, but no error control, as TP++ provide end to end error control. TP++ is designed to efficiently handle data loss caused by congestion and packet misordering caused by multipath routing.

TP++ is designed to carry three major application classes.

■ Constrained latency services are necessary for human interaction (e.g., conventional voice and video), process control, and remote sensing. As the name implies, data for applications using this service are worthless if they do not arrive within a certain time. Constrained latency services generally are willing to accept a small data loss rate.

■ Transactions occur in distributed systems such as distributed operating systems, database and reservation systems. They are bursty in nature and require a small or moderate amount of data to be transmitted.

■ Bulk data transfer is a service that carries a large amount of data between computers. Generally, all data must be received reliably before the receiving application can proceed. Loose latency constraints are acceptable for higher efficiency.

6 Rapid Transport Protocol (RTP)

RTP is a transport protocol being developed at IBM that will permit operations at gigabits/second speeds expected from the network. It is a "lightweight" transport protocol in the sense that it has a very small number of states and timers and has been designed to minimize the amount of buffer copying and interface crossings. In addition to these features, RTP provide some key new functions [8]. It has a fast connection setup capability, wherein data can be sent in the first packet. Thus datagram and connection based services are provided in a single, consistent framework. Error

recovery is optional, and is implemented with a single timer at the receiver. Both Go-Back-N and selective repeat modes are supported. Other notable characteristics include multicast support, reliable full-duplex message delivery and arbitrary length messages. RTP is an optimistic protocol, which improves performance by assuming that the network is "mostly" reliable and that applications are typically ready and willing to communicate. This optimism allows it, for example, to speed-up connection-establishment and minimize handshaking between endpoints.

RTP is able to support all TCP/IP clients. A number of applications have also been implemented [10] that run directly over RTP.

7 Conclusion

In the mid-1980's, TCP/IP was accused of being too slow for high-speed networks, but researchers have found that not TCP/IP but the bad implementations, faulty hardware is the main reason for the poor performance. Though TCP/IP has improved a lot, other transport protocols such as XTP and HTPNET are more efficient for transport protocol processing at the end systems.

REFERENCES

[1] C.S. Chan, T.S. Chan and I. Gorton, A High Performance Transputer-Based Memory Architecture for HTPNET
http://www.vast.unsw.edu.au/~chants/wtc95.ps.Z"

[2] T.S. Chan and I. Gorton, Transputer Support for Protocol Engine Design
"http://www.vast.unsw.edu.au/~chants/pcat.ps.Z"

[3] T.S. Chan and I. Gorton, Design Issues of Error Control Protocol for High-Speed Networks
"http://www.vast.unsw.edu.au/~chants/isita.ps.Z"

[4]T.S. Chan and I. Gorton, Formal Validation of a High Performance Error COntrol Protocol Using SPIN
"http://www.vast.unsw.edu.au/~chants

[5] T.S. Chan and I. Gorton, A High Performance Transport Protocol
http://www.vast.unsw.edu.au/chants/journal.ps.Z

[6] T.S. Chan and I. Gorton, A Transputer-based Implementation of HTPNET: A Transport Protocol for Broadband Networks Transputer Applications and Systems `93 Vol 2, Proceedings of

the 1993 World Transputer Congress
Aachen, Germany, IOS Press, Amsterdam, 1993, pp 899-910.

[7] T.S. Chan and I. Gorton, A Parallel Approach to High-Speed Protocol Processing, Transputer Applications and Systems `94, Proceedings of the 1994 World Transputer Congress
Lake Como, Italy, IOS Press, Amsterdam, 1994, pp 209-222

[8] Stephen Pink, TCP/IP on Gigabit Networks, High Performance Networks, Frontiers and Experience,
Kluwer Academic Publishers, 1994, pp 135 -156

[9] Bruce S. Davice, Jonathan M. Smith, David D. Clark, AURORA: An Experiment in Gigabit Network Technologies, High Performance Networks, Frontiers and Experience
Kluwer Academic Publishers, 1994, pp 13 - 25.

[10] David C. Feldmeir, An Overview of the TP++ transport protocol project, High Performance Networks, Frontiers and Experience
Kluwer Academic Publishers, 1994, pp 157 - 176

[11] Otto Spaniol, Architecture and Protocols for High Speed Networks
Kluwer Academic Publishers, 1994

[12]Greg Chesson, XTP/PE Design Consideration
Silicon Graphics Computer Systems, Mountain View, Ca 94o39-7311, greg@sgi.com

[13]Greg Chesson, The Evaluation of XTP
Silicon Graphics Computer System, Mountain View, Ca 94o39-7311, greg@sgi.com

[14] W. Timothy Strayer, Michael J. Lewis, Raymond E. Cline, Jr., XTP as a Transport Protocol for Distributed Parallel Processing.

[15] Alfred C. Weaver, What is the Xpress Transport Protocol?
Network Xpress Inc., VA

[16] XTP Forum, Xpress Transport Protocol Specification, Revision 4.0, March 1, 1995
XTP Forum, 1394 Greenworth Place, Santa Barbara, CA 93108

[17] Alfred C. Weaver, Xpress Transport Protocol Version 4
Dept. of Computer Science, University of Virginia

[18] W. Timothy Strayer, Simon Gray, Raymond E. Cline, Jr., An Object Oriented Implementation of Xpress Transport Protocol,
Sandia National Laboratories, CA

[Other Reports on Recent Advances in Networking](#)
[Back to Raj Jain's Home Page](#)

Last modified: Aug. 24, 1995