Report Number: WUCS-88-30

1988-10-01

Comments on Proposed Transport Protocols

Anil Bhatia, James Sterbenz, and Gurudatta M. Parulkar

Over the last few years, a number of research groups have made considerable progress on the design of high speed networks- on the order of a few hundred Mbps to the few Gbps. The emphasis of this work has been on the design of packet switches and on the design of network access protocols. However, this work has not yet addressed the internetworking and transport level issues in the high speed internet. As part of our effort on the design of VHSI model, we considered the appropriateness of recently proposed transport protocols, NETBLT and VMTP, as candidates for the... Read complete abstract on page 2.

Comments on Proposed Transport Protocols

Anil Bhatia, James Sterbenz, and Gurudatta M. Parulkar

Complete Abstract:

Over the last few years, a number of research groups have made considerable progress on the design of high speed networks- on the order of a few hundred Mbps to the few Gbps. The emphasis of this work has been on the design of packet switches and on the design of network access protocols. However, this work has not yet addressed the internetworking and transport level issues in the high speed internet. As part of our effort on the design of VHSI model, we considered the appropriateness of recently proposed transport protocols, NETBLT and VMTP, as candidates for the transport protocol for our VHSI model. The summary of the results of the study is that NETBLT and VMTP have contributed a number of interesting ideas to the design of transport protocols, and they do improve upon TCP within the current Internet model for the applications they were originally designed for. However, we believe that these protocols are not appropriate solutions for the VHSI model, because the underlying assumptions and trade-offs that these protocols are based on the very different in the VHSI model. For example, the VHSI model assumes a quasi-reliable connection-oriented internet protocol (as opposed to the current unreliable datagram IP), which can make performance guarantees and can ensure that the internet is congestion free (almost all of the time). Also the network speeds in the VHSI are the few order of the magnitude more than what NETBLT and VMTP assume. We argue that the transport protocols in the VHSI model should avoid end-to-end flow control as much as possible, and make the end-to-end error control application specific and independent of the end-to-end latency. In general, the transport protocols should be simpler, designed to be mostly implemented in VLSI, well integrated with the host architecture and operating system, and targeted for a specific class of applications.

COMMENTS ON PROPOSED TRANSPORT PROTOCOLS

Anil Bhatia, James Sterbenz and Guru Parulkar

WUCS-88-30

October 1988

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

James Sterbenz is on leave of absence from IBM Corporation. This work supported by Bell Communications Research, Italtel SIT, NEC, BNR and National Science Foundation grant DCI-8600947.

Abstract

Over the last few years, a number of research groups have made considerable progress on the design of high speed networks - on the order of a few hundred Mbps to a few Gbps. The emphasis of this work has been on the design of packet switches and on the design of network access protocols. However, this work has not yet addressed the internetworking and transport level issues in a high speed internet. As part of our effort on the design of a VHSI model, we considered the appropriateness of recently proposed transport protocols, NETBLT and VMTP, as candidates for the transport protocol for our VHSI model.

The summary of the results of this study is that NETBLT and VMTP have contributed a number of interesting ideas to the design of transport protocols, and they do improve upon TCP within the current Internet model for the applications they were originally designed for. However, we believe that these protocols are not appropriate solutions for the VHSI model, because the underlying assumptions and trade-offs that these protocols are based on are very different in the VHSI model. For example, the VHSI model assumes a *quasi-reliable* connection-oriented internet protocol (as opposed to the current unreliable datagram IP), which can make performance guarantees and can ensure that the internet is congestion free (almost all the time). Also, the network speeds in the VHSI are a few order of magnitude more than what NETBLT and VMTP assume. We argue that the transport protocols in the VHSI model should avoid end-to-end flow control as much as possible, and make the end-to-end error control application specific and independent of the end-to-end latency. In general, the transport protocols should be simpler, designed to be mostly implemented in VLSI, well integrated with the host architecture and operating system, and targeted for a specific class of applications.

James Sterbenz in on leave of absence from IBM Corporation. This work supported by Bell Communications Research, Italtel SIT, NEC, BNR and National Science Foundation grant DCI-8600947.

Comments on Proposed Transport Protocols

Anil Bhatia *James Sterbenz*
Guru Parulkar
Department of Computer Science
Washington University
St. Louis MO 63130

October 7, 1988

1 Introduction

During the last few years, yet another computer communication revolution has been in progress. Researchers in the communication area have pushed the state of the art in data transmission and switching speeds by orders of magnitude. For example, optical fibers that can transmit tens of Gigabits/sec over a few kilometers without a repeater are now available; switching systems that can switch bit-streams with data rates of a few hundred Mbps are being proto typed; and switching systems having data rates of up to a Gbps are being planned. Also, this research effort has provided some added functionality to the communication substrate, which is useful in supporting a variety of applications, including high speed data, still and motion video, computer imaging, and voice. The added functionality includes multipoint communication, variable bandwidth connections, and at a more fundamental level, a new interpretation to a "connection." However, the research on high speed networking has not yet addressed the internetworking of diverse high speed networks (internet issues), and the question of how to deliver the underlying high bandwidth end-to-end to diverse applications within an operating system (transport issues).

The ARPA Internet has provided a good model for internetworking of diverse networks with variety of host computers. As the Internet is rapidly growing in size, a number of problems have shown up with the existing internet model. Also, a number of underlying assumptions of the current Internet are not valid in a very high speed internetworking (VHSI) context. Thus, the internet model, which has contributed a number of fundamental ideas, cannot be used in its current form for the VHSI of the future.

At Washington University we have been involved in a major research effort with its focus on the development of a proto type very high speed network, called the Broadcast Packet Network (BPN) [25,26], and on the internetworking and transport level issues in a VHSI[21]. As part of this effort, we have considered the appropriateness of TCP and two recently proposed transport protocols, NETBLT[8,9] and VMTP[2,3], for the VHSI environment. The purpose of this report is to present the results of this study and present our thoughts on the design of transport protocols in VHSI; we call them application-oriented lightweight transport protocols (ALTPs).

The summary of the results of this study is that NETBLT and VMTP have contributed interesting ideas to the design of the transport protocols, and they do improve upon TCP within the current Internet model for the applications they were originally designed for. However, we believe that these protocols are not appropriate solutions for the VHSI model, because the underlying assumptions and trade-offs that these protocols are based on are very different in the VHSI model. For example, the VHSI model assumes a *quasi-reliable* connection-oriented internet protocol (as opposed to the current unreliable datagram IP), which can make performance guarantees and can ensure that the internet is congestion free (almost all the time). Also, the network speeds in the VHSI are from a few hundred Mbps to a few Gbps which are a few order of magnitude more than what NETBLT and VMTP assume. We believe that the transport protocols in the VHSI model should avoid end-to-end flow control as much as possible, and make the end-to-end error control independent of the end-to-end latency. In general, the transport protocols should be simpler, designed to be mostly implemented in VLSI, well integrated with the host architecture and operating system, and targeted for a specific class of applications.

The rest of this report is organized as follows: Section 2 provides an overview of the ARPA Internet model and the VHSI model. It also gives an overview of TCP and its associated problems. Section 3 provides an overview of NETBLT and VMTP and summarizes how these protocols are improvements over TCP. Section 4 describes the limitations of NETBLT and VMTP within the VHSI model. Section 5 presents our thoughts on the design of ALTPs for VHSI, and finally, Section 6 is the conclusion.

2 Background

2.1 The ARPA Internet Model

ARPANET started out as a single homogeneous packet switching network with the goal of connecting one computer from every major research and academic institution. With the proliferation of local area networks (LANs), it soon evolved into a backbone network essentially connecting campus LANs from these institutions. ARPANET has evolved further into what is called the Arpa Internet as more and more organizations are discovering the benefits of computer networking. The ARPA Internet today comprises a few backbone networks, a number of regional networks, and a large number of campus networks at various participating institutions. Such campus networks typically comprise a few LANs, such as ethernet and token ring networks.

Internet Protocol Hierarchy

The Internet uses a protocol hierarchy which is popularly known as the TCP/IP protocol suite (figure 1)[12,13,22]. At the application level in this hierarchy, the three most commonly used applications are TELNET (remote login protocol), FTP (file transfer protocol), and SMTP (simple mail transfer protocol). A variety of other applications, including voice and multimedia mail, have been developed for the Internet but are not widely used.

At the transport layer, there are both a datagram (UDP: user datagram protocol) and a virtual connection-oriented (TCP: transmission control protocol) interface. Most of the applications use the connection-oriented interface of TCP which ensures reliable delivery of user packets in sequence with no duplicates. At the internet level, there is only one internet packet forwarding protocol, appropriately called the Internet protocol (IP). IP is a datagram oriented protocol and does not make any performance guarantees — it can lose, duplicate and resequence packets. For error and congestion control at

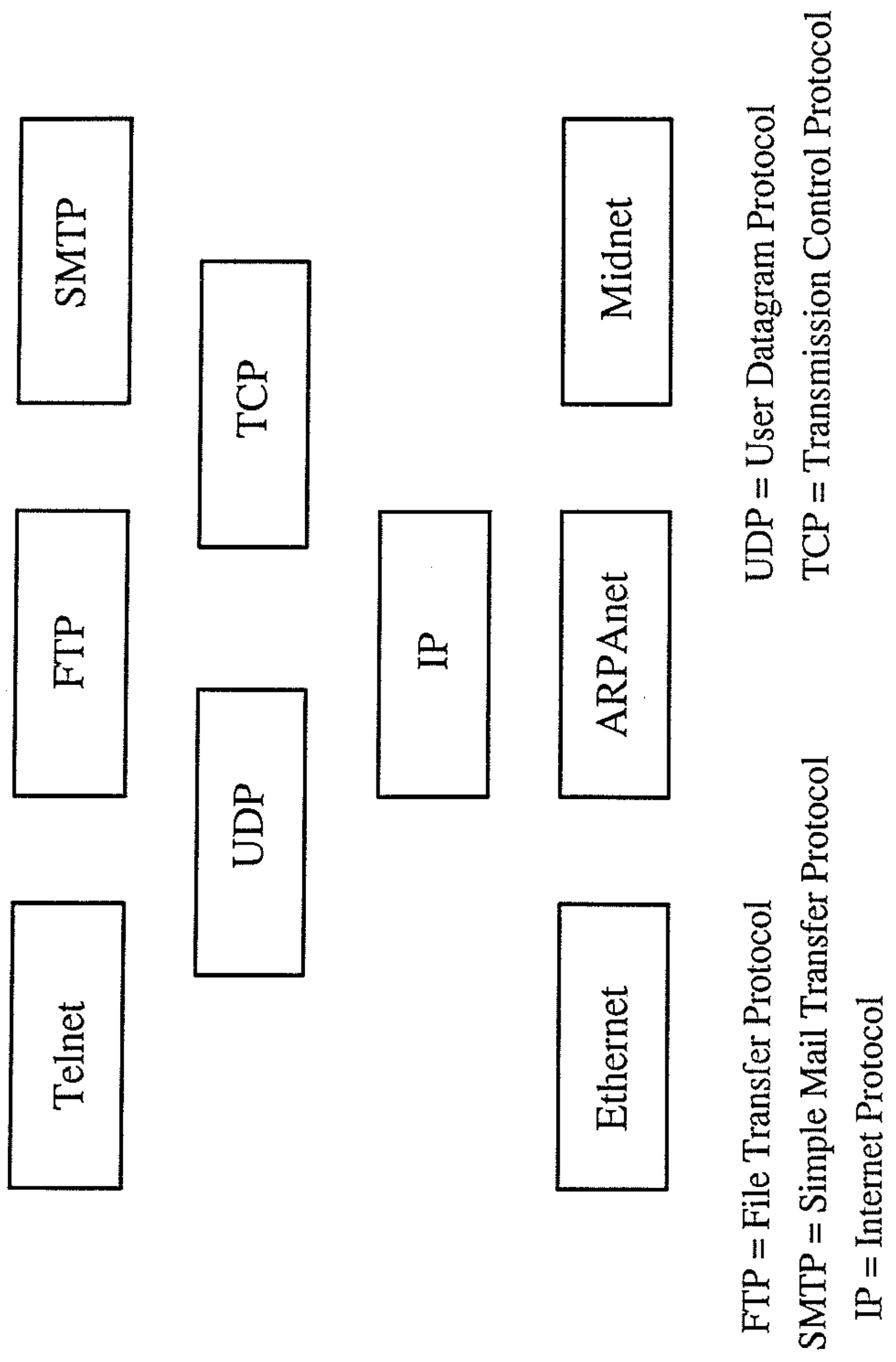
SMTP
FTP
Telnet
TCP
UDP
IP
Midnet
ARPAnet
Ethernet
UDP = User Datagram Protocol
TCP = Transmission Control Protocol
FTP = File Transfer Protocol
SMTP = Simple Mail Transfer Protocol
IP = Internet Protocol

Figure 1: Internet Protocol Hierarchy

the internet level, IP uses a complementary protocol called internet control message protocol (ICMP) which actually uses IP for forwarding its packets IP works on the top of a variety of network protocols, but gives an impression to its upper layers that there exists only one homogeneous network. At the network level, there are all kinds of networks in this Internet as mentioned earlier, and they have their own network access protocols. The expectations of the Internet from its component subnets are modest: the subnet should try its best to forward datagrams toward their final destinations, but it is acceptable for the network to lose, resequence, and duplicate datagrams.

2.2 Revised Internet Model - VHSI Model

A new communication revolution is in progress, and it is time to plan the next generation of internet which will have the recent high speed networks as their component subnets. We pointed out earlier that the high speed networking efforts have not addressed the internetworking and the transport level issues, and that the ARPA internet model is not sufficient *as-is* for the VHSI environment. The purpose of this section is to present a revised internet model (VHSI model) which is more appropriate for the VHSI context.

Figure 2 shows the protocol hierarchy of the VHSI model[1]. Comparison of VHSI protocol hierachy with that of ARPA internet (Figure 1) shows that they are similar in a number of ways[2]. For example, both models have application, transport, internet, and network layers, and these layers have similar functionality in the two models. For example, the purpose of the internet layer in both models is to create a homogeneous network on the top of a diverse set of networks. However, there are significant differences between the two models which we believe will address the limitations of the current Internet and will make this model more appropriate for the VHSI.

Comments about the VHSI model

In the following paragraphs we summarize the salient features of this model.

[1]The current internet protocol suite can co-exist with the VHSI protocols during the transition time, during which the new protocols mature and old ones get slowly phased out.

[2]Similarly, it is not difficult to map the VHSI protocol hierachy to ISO's OSI model.

- The hierarchical VHSI model is such that every layer can make performance guarantees and be at least *quasi-reliable*. Thus, reliability and ability to guarantee performance is built bottom up. For example, the connection-oriented internet protocol (CIP) in VHSI can maintain an end-to-end connection (of certain given attributes) and can ensure a very low probability for a packet being lost, resequenced, or duplicated.

- The main purpose of the internet level in VHSI is to maintain, across the complex physical internet, connections which are application independent and have no inherent semantics associated with them. In other words, an internet connection can be thought of as a simple piece of wire connecting two telephones or it can be an extension of a computer bus allowing two computers to work as a multiprocessor system.

  On the other hand, transport protocols in VHSI are application dependent and are optimized for specific classes of applications. They are responsible for converting the communication facility provided at the internet level to a communication environment best suited for the application. For example, for a distributed IPC (inter-process communication) application, the transport protocol can present the network as an extension of the host itself. Thus, processes on two hosts across the internet will communicate with each other as if they are on the same machine, possibly by a shared memory paradigm similar to what they would use in a tightly coupled multiprocessor system (a functionality that can successfully implement Dave Farber's shared memory paradigm [16]).

- The layered protocol hierarchy in VHSI is for control purposes and not for actual data communication. For example, once a connection has been established, the actual data communication bypasses most of this hierarchy, that is, the data is not necessarily processed by each protocol layer. Thus, the layering of protocols does not have to limit the performance of an application.

- The VHSI model allows the possibility of an application using more than one transport protocol. For example, a multimedia conferencing application can use a transport protocol for voice, and another transport protocol for video. The application will have the responsibility to coordinate the data transfer with more than one transport protocol.

- The purpose of this model is to support a variety of applications, and be able to provide a guaranteed level of performance as needed by an application. As a result, some of the flexibility which existed in the Internet model is taken away. For example, in this model, every component subnetwork is expected to provide guarantees about the performance and is expected not to drop, lose, or resequence packets except with a specified low probability.

2.3 Overview of TCP and Its Limitations

The functionality that is typically associated with the transport protocols include end-to-end flow and error control, some kind of congestion control, and multiplexing. In the current Internet model, there is a notion that one can design general purpose transport protocols which can cater to a variety of applications. In the following paragraphs, we outline the limitations of the transport level in the existing model.

Flow and Error Control

Most of today's transport protocols are designed to work on the top of the current IP, which is unreliable and does not make any performance guarantees. As a result, transport protocols have suitable mechanisms for end-to-end flow and error control to provide reliability to the applications. For example, TCP and most other transport protocols use a variant of the classical sliding window protocol for combined flow and error control. Problems with the sliding window protocol are that it is complex, its combined flow and error control leads to performance bottlenecks [8], and it heavily depends on the end-to-end latency (for its timers), which is difficult to estimate in a complex internet. Thus, the current end-to-end flow and error control mechanisms can be a serious bottleneck in delivering high performance to the applications of the VHSI.

Congestion Control

Transport protocols try to implement congestion control by adjusting the rate of packet transmission based on the estimates of round trip

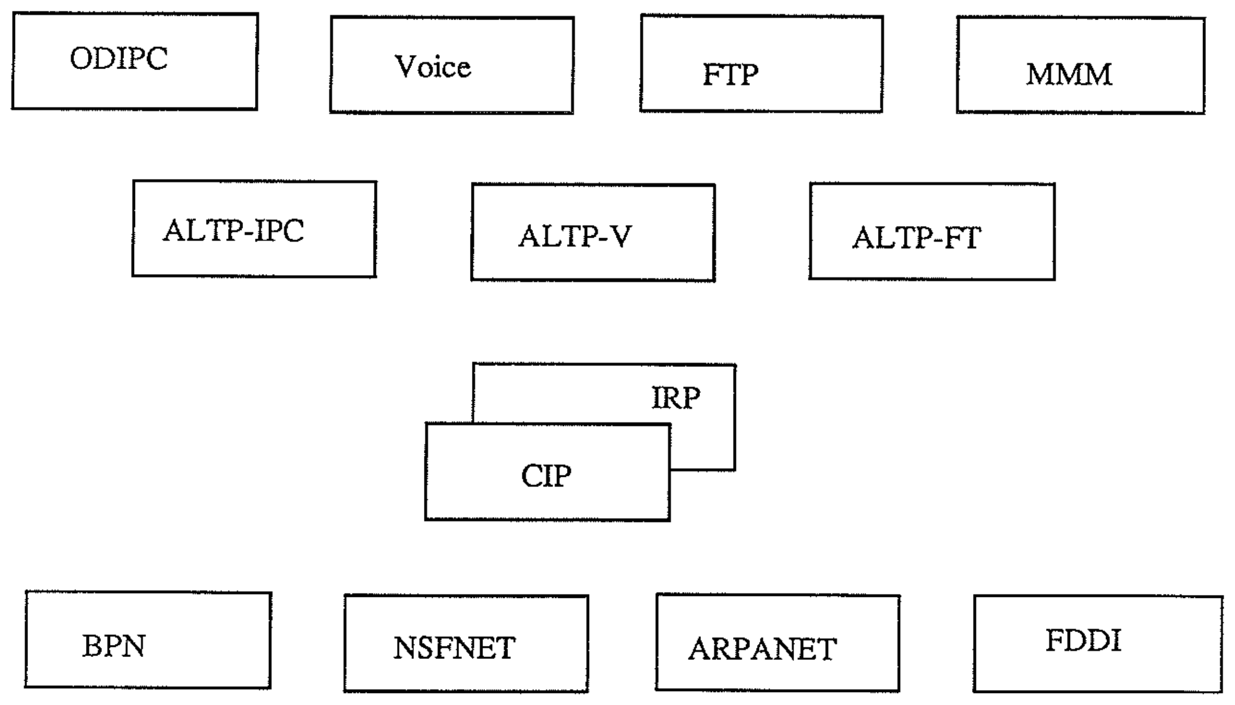

Figure 2: VHSI Model - Protocol Hierarchy

ODIPC	Object-oriented Distributed IPC
MMM	Multi Media Mail
ALTP	Application-oriented Lightweight Transport Protocol
ALTP-V	ALTP for Voice
ALTP-FT	ALTP for File Transfer
CIP	Connection-oriented Internet Protocol
IRP	Internet Routing Protocol
BPN	Broadcast Packet Network

Figure 2: VHSI Protocol Hierarchy

delay. In simple terms, the strategy works as follows: if an acknowledgement for a packet takes longer, the transport protocol assumes that the network is congested, and therefore, the transport protocol reduces the rate of its packet transmission by adjusting its window size. We believe that such a congestion control is not effective because the actual round trip delay is difficult to estimate within the current internet, moreover, it is not an accurate measure of the congestion (at least not a direct measure). Thus, congestion control at the transport level is not necessarily useful and makes the protocols unnecessarily complex.

Recently, Van Jacobson and others have done interesting work in getting the conges tion control strategy of TCP to work better in a congested internet [30]. The emphasis of this work is on the adjustments of the TCP transmit window to avoid congestion based on end-to-end delay. The strategy argues for a very small window at the beginning which is increased slowly with time until the end-to-end ACK indicates saturation of the transmission path. During the life of the TCP connection, any lost packets are considered an indication of congestion and result into a drasitc reduction of the window size. These modifications of TCP have definitely helped achieve better results in today's congested internet. However, in high speed networks, it is not clear if applications can tolerate slow start, and if they can tolerate window adjustments as the state of congestion in the network is changing.

General Purpose Transport Protocol

The current internet model assumes that a general purpose transport protocol can be designed to be used by a variety of applications. The experience with the TCP has shown that, as a general purpose reliable transport protocol, it is not optimized for any application, and thus cannot deliver the high performance to, or work well with, a number of applications [2].

In short, it can be said that the main limitations of TCP within a high speed internet are its window based flow control, dependency of flow control on error control, its complexity for VLSI implementation, and its poor integration with the host operating system and the hardware architecture. Some of these limitations have been identified and discussed in papers on NETBLT and VMTP.

3 Proposed Transport Protocols

There are several alternative transport protocols that have been proposed for use in the Internet. We have examined the specifications for the four transport protocols which have been published as official internet protocol RFC's [23]. These are: reliable data protocol (RDP) [28], internet reliable transaction protocol (IRTP)[20], network bulk transfer protocol (NETBLT), and versatile message transaction protocol (VMTP).

In this report we have considered VMTP and NETBLT in detail because they are more recent, designed for high speed networks, and considered candidates for the next generation transport protocols. However, RDP and IRTP also have some interesting features. For example, RDP is a simpler and application-oriented transport protocol, something we favor in a VHSI model. Similarly, IRTP is a simple transaction-oriented protocol, designed to maintain a multipoint connection among hosts which communicate on a relatively infrequent basis. However, its design emphasizes simplicity over efficiency.

3.1 NETBLT

NETBLT is designed for high throughput, bulk data transmission applications. It provides a virtual circuit between its clients. It works by opening a connection between a client and a server, transferring data in a series of numbered large blocks (each such block has many packets, so we call it a *superpacket*), and then closing the connection. NETBLT is an improvement over the previous transport protocols in the following three ways: it uses rate-based instead of window-based flow control, retransmission timers are based on packet inter-arrival time rather than on the network round-trip-delay (RTD), and the timers are on the receiving end rather than on the transmitting end. As a result of these modifications, flow control becomes independent of error control, and also becomes independent of RTD to a certain extent, and error recovery is made more efficient by having fewer unnecessary retransmissions.

Flow control in NETBLT

NETBLT developers addressed the problems associated with the window based flow control by introducing rate-based flow control. NETBLT uses two strategies for flow control - one external (at the client level), and one internal (rate based).

The external flow control works as follows. Before a buffer (superpacket) can be transmitted, NETBLT confirms that both the client and the server have set up matching buffers, that one is ready to send data, and that the other is ready to receive data. The superpacket size and the transmission rate are negotiable at the beginning of each superpacket transmission. The client or the server can, therefore, control the flow of data by changing the buffer size, or changing the transmission rate, or not providing a new buffer. Negotiating the new buffer and rate information back and forth depends on RTD, which makes this part of the flow control RTD dependent.

The internal flow control is the rate based transmission of packets within a superpacket. This means that the transmitter keeps sending the packets in the superpacket at the negotiated rate; the receiver is expected to receive packets at this rate and have preallocated buffer space for all packets in the given superpacket. This part of the flow control is obviously independent of RTD.

Packets needing retransmissions and new packets not yet transmitted are placed in the same queue, and all packets in this queue are sent at the predetermined rate. Thus, retransmissions do not increase the network load, measured in packets per second, and therefore, do not induce congestion. This also leads to the decoupling of the flow and error control.

The silly window syndrome [7] is avoided here, because the transmitter is not waiting for acknowledgements after transmitting each packet, and its window cannot get blocked. The external flow control strategy pushes part of the flow control to the application level, though only a few existing applications may provide a facility for appropriate buffer allocations.

Error recovery and end state synchronization in NETBLT

The sending and the receiving NETBLTs synchronize their end states[3] either upon successful transmission of a buffer, or upon determination by the receiver that information is missing from a particular buffer. In the first case, a single message ACKs all packets contained in a particular buffer. In the second case, a single message tells the sender exactly which packets to retransmit.

The three important features of error recovery in NETBLT are its use of selective acknowledgement, timers based on inter-arrival time, and timers at the receiving end. The positive implications of these features are that the use of selective acknowledgement avoids the problems associated with cumulative acknowledgement; since the packet inter-arrival time is more deterministic than RTD, the retransmission timers based on the inter-arrival time can be estimated more accurately, resulting in fewer unnecessary retransmissions; and since the timers are on the receiving end, and the receiver knows exactly which packets need retransmission, additional unnecessary retransmissions can be eliminated. Also, as mentioned before, retransmissions in this protocol do not change the rate of packet transmissions, and thus, do not increase congestion.

Rate-based flow control to do congestion control

In the current Internet, most data loss in the network is due to congestion. Rate-based flow control can reduce this congestion in a number of ways. First, since packet retransmissions occur "in-band," they cause no extra load on the network. Second, since unnecessary retransmissions are avoided as explained above, this results in lower congestion in the network. Last, the rate is flexible and can be adjusted to reflect the network's current ability to transport data.

[3]An end state is the state of the connection at each end, that is, the number of the superpacket which is under transmission at the transmitting end, and which packets within the superpacket have been successfully received at the receiving end.

3.2 VMTP

VMTP is designed primarily for use by a group of computers operating as a distributed system. It is basically a reliable transaction stream (as opposed to a data stream) protocol. A VMTP message transaction is initiated by a client sending a request message to a server entity, and terminated by the server sending back a response message. The response acknowledges to the client the receipt of the request message. The server can subsequently receive from the client either another request, which serves as an acknowledgement of the previous response, or an explicit acknowledgement. Data is sent with the request and response messages. VMTP, by itself, does not have a concept of a virtual circuit or a connection. However, it provides facilities for the higher level modules to implement a conversation, which is similar to a connection.

Flow control in VMTP

VMTP uses a packet-group based flow control which works as follows: a transmitter sends a group of packets (call it a superpacket) of at most 16 Kbytes, which constitutes a VMTP message. The receiver accepts and acknowledges the superpacket as a unit before further data is exchanged. To avoid the silly window syndrome, the receiver is programmed such that it does not advance the window until it can be advanced by a significant amount, which is a maximum of 16 Kbytes.

Error control in VMTP

Flow and error control in VMTP are not as decoupled as in NETBLT. In VMTP, the receiver has to acknowledge the proper receipt of a packet group before the next packet group can be exchanged, and the retransmissions can potentially affect the packet rate (or the network load). However, VMTP also uses selective acknowledgement, indicated by a 32-bit mask, and thus, it avoids the problems associated with cumulative acknowledgement.

Congestion Control in VMTP

Designers of VMTP believe that the selective acknowledgement and monitoring of retransmission request patterns can provide a means of dynamically detecting when the transmission rate is too high, and as a result, causing the packet overruns or congestion. They suggest that the transmission rate can be adjusted, in response to the retransmission requests, such as to reduce possible congestion.

VMTP also has the functionality to provide services such as remote procedure call, multicast and real-time datagrams, and to support variants of the basic transaction, such as group message transaction (example: file name query from multiple file servers), and forwarded message transaction (example: file open request, forwarded to an authorization server, then forwarded to a basic file server).

4 Limitations of NETBLT and VMTP in VHSI

As we enter the era of very high speed networking, we need to make appropriate changes in the internet protocols. We are proposing a revised internet model called the VHSI model. The VHSI model argues for a quasi-reliable connection-oriented IP which can make performance guarantees. The VHSI model also argues for a number of application-oriented lightweight transport protocols which are discussed in Section 5. As pointed out earlier, we considered NETBLT and VMTP as possible ALTP candidates for the VHSI model. Though they have addressed and successfully resolved some of the problems with TCP, they are not satisfactory for the next generation of High Speed Internetworking for the following reasons:

- *Problems with flow control:* In both NETBLT and VMTP, flow control still depends on the end-to-end latency, though less than in the case of TCP. For example, in VMTP, the flow control method is essentially a stop and wait protocol (same as window size equal to one) at the superpacket level, because the sender has to wait until a superpacket in a session has been correctly acknowledged before starting the next superpakcet. And in NETBLT, the external flow control clearly depends on the RTD.

  Another limitation of the NETBLT's rate-based flow control is that the intermediate nodes (packet switches and gateways) are not consulted during the rate negotiation. Thus, there may be a situation where the sender and receiver agree on a packet rate which some intermediate systems cannot support, and thus, may lead to serious packet overruns.

The NETBLT retransmission timers are more effective because they are based on the packet inter-arrival time rather than on the RTD. These timers, however, can be still difficult to estimate because the packet inter-arrival time on the receiving end may have an unpredictable distribution.

- *Problems with congestion control:* The approach taken in both these protocols is to correct congestion after it is detected, rather than trying to avoid it in the first place. We believe that in high speed networks, the latter approach of congestion avoidance is more effective. Also, the congestion avoidance/control mechanisms should be provided at lower layers because various congestion parameters are difficult to estimate accurately at the transport layer.

  For example, packet retransmission requests in VMTP are considered a measure of packet overruns somewhere in the network and hence a measure of congestion. Thus, retransmission request patterns are used to adjust the rate of transmission. This method of congestion control is not responsive, because it might take a long time for the transmitter to make a judgement about packet overruns based on the selective retransmission requests. The congestion situation might have changed by this time, and adjusting the transmission rate may be of no help.

  NETBLT implements congestion control by rate-based flow control. This also leads to same problem as in VMTP. The end-to-end latency may be so high that a change in the rate would be too late to be able to relieve congestion. In rate based flow control, negotiation of the transmission rate takes at least one RTD. This means that if a condition of overrun is detected, it will take at least one RTD before the sender knows about it. Even then, it cannot change the rate if it is in the middle of transmitting a superpacket.

- *General purpose protocol:* VMTP is designed to be a general purpose transport protocol meant for all applications. We believe that in high speed networks, it would be more appropriate to have a number of application-oriented lightweight transport protocols (ALTPs), where an ALTP serves a class of similar applications. We believe ALTPs can be simpler, well integrated with the host architecture and its operating system, easily implemented in VLSI, and their flow and error control can be more effective because they understand the data objects exchanged by the application level clients. In the next section we present our thoughts on the design of ALTPs.

- *Hardware implementation:* These protocols were designed to be implemented in software and have high complexity for VLSI implementations. It is clear that transport protocols (or any protocol for that matter) have to be implemented in VLSI for them to deliver high performance. We want to go one step further and suggest that it is not sufficient to take the existing protocols designed for software implementations and implement them in VLSI, because this leads to complex, expensive, and inefficient implementations. We believe that we can design simpler and more efficient transport protocols and their implementation if we know apriori that they would have VLSI implementations.
- *Packet size:* These protocols hint towards a larger packet size. However, high speed networks favor small packet sizes for a variety of reasons, and we believe that increasing the packet size beyond the size of the current internet datagram for higher throughput is not an attractive idea.

5 Thoughts on the Design of Transport Protocols in VHSI

At the transport level, the VHSI model argues for a set of simple, lightweight, custom transport protocols for various classes of applications. The motivation is that these transport protocols can be implemented largely in VLSI hardware, and can be optimized to provide the kind of performance guarantees and functionality the specific applications need. As chip densities increase, even complex protocols that were originally designed for software may be successfully implemented in hardware. It is important to note, however, that for the protocol to be *efficiently* implemented in hardware, the protocol and hardware design should be well integrated.

Thus, we argue that the application-oriented lightweight transport protocols (ALTPs), can serve its clients better than a general purpose transport protocol. For example, a transport protocol designed to support voice communication can be designed such that it guarantees less than 30 ms delay, guarantees no out of sequence packets, and allows only a few dropped packets. A transport protocol designed for distributed IPC (inter-process communication) should be able to guarantee minimal delay, reliable transaction delivery, use a shared memory interface to processes on different machines, *etc.*

ALTPs will avoid end-to-end flow control by relying on the bottom-up flow control, *i.e.* flow control is provided by the underlying sub-networks and gateways, in the form of rate specification enforcements. The error control in a ALTP will be application specific, and be independent of end-to-end latency. ALTPs also do not need to be concerned with congestion control (or avoidance).

Current work in this area is also manifest in the work on the express transport protocol (XTP) and the protocol engine (PE) [4,5,6]. While the goals for XTP are similar to those for ALTPs in VHSI, there are also some significant differences. In simple terms, the XTP approach is to take the existing protocols mechanisms, streamline the packet format for pipeline processing, and implement each step in the pipeline using a semicustom 680x0 processor. Thus, we argue that the XTP has not essentially addressed the basic and essential end-to-end issues, such as end-to-end flow, congestion, and error control for high speed protocols. This hardware implementation will definitely provide more throughput than the current software implementations. However, we are more interested in researching how to make end-to-end mechanisms more efficient for high speed networks and how to divide the labor between hardware and software for higher throughput. Clearly, it is too early to do a more detailed comparison of XTP with ALTPs, as both of these efforts are in their early stages. The following sections outline our thoughts on the design of flow and error control in ALTPs.

5.1 Flow control in ALTPs

When an ALTP or an application opens a connection, it specifies attributes of the connection in terms of average and peak bandwidth, and a factor reflecting the burstiness of the transmission. These parameters can be translated into buffer requirements, based on a rate between the average and peak specifications. This allocation has been explored in [1]. Since the connection set up is end-to-end, all the intermediate systems, including various packet switches and gateways, as well as the endpoint hosts that this connection goes through, can make appropriate buffer and resource reservations. The rate specification will have to be negotiated between the transport and internet/network layers to ensure that the requested rate does not exceed the capacity of internal network nodes (packet switches, gateways, and subnetworks). As a result, as long as both ends transmit subject to the rate specification, the probability of packet loss due to buffer overruns is very low.

Due to the cumulative effects of multiple connections in intermediate nodes of a given connection, the distribution of (bursty) packet arrivals may change from one node to the next in the corresponding connection. That is, the actual packet inter-arrival times are not necessarily related to the external rate specification. Thus, some degree of bandwidth over-allocation (with respect to the connection endpoints) will be necessary.

Check posts, particularly at subnetwork boundaries (gateways) will be used to ensure adherence to the specifications of a connection, and to smooth the variations in the resource usage of the connection. This over allocation is expected to be an insignificant fraction of the total resources, and the check post logic can easily be encorporated in gateways and hosts as required. However, this issue needs to be addressed in detail by developing appropriate analytical and simulation models.

During the life of a connection, the transmitter and receiver may want to change the specifications of the connection. This may require end-to-end negotiations. In the current Internet environment, such a change is typically required because the operating system may not be able to commit buffers to connections for their entire duration, or because of errors causing changes in end-to-end flow control. In ALTPs, error and flow control will be separated, and mechanisms will be provided to ensure that resources and buffers are available for the life of the connection. In the case where an intermediate node does not have the excess bandwidth to allow for a request for transmission rate increase, the connection will have to be rerouted.

End-to-end flow control can not be eliminated for some applications, which by their nature, require frequent changes to the connection characteristics, and thus end-to-end negotiations. In such a situation, the best way to implement flow control may be similar to what is suggested in the current transport protocol proposals.

5.2 Error Control in ALTPs

For end-to-end error control, application specific methods which are independent of end-to-end latency will be used. For example, consider a transport protocol designed for a distributed object-oriented inter-process control (ALTP-IPC), which allows processes to communicate by a shared memory paradigm. Assume that one process sends a memory segment with N pages, and because of errors, n pages are corrupted or lost. In this case, the transport protocol should store the pages received correctly in memory, allow processes to access those pages, and should request retransmission of just the n pages received with error. Thus, the processes can start to execute, unless they need access to pages which are being retransmitted. Note that this is possible because ALTP-IPC has some knowledge of its application, and also understands the application specific data objects.

In the case of a voice transmission (ALTP-V), when a packet is lost, there will not be a retransmission. Similarly, if a packet arrives out of sequence, ALTP-V will drop this packet rather than sending it to the application.

In the case of a file transfer (ALTP-FT), the whole file will be received, and only packets received in error or lost will be retransmitted (which do not include out of sequence or duplicated packets). It should be noted that the errors and corresponding retransmissions do not affect the error-free transmission of other packets. In other words, two ends do not need frequent synchronizations, and the selective retransmission strategy is application dependent.

Note that in these examples, the various ALTPs use different error control mechanisms, and the mechanisms are more effective because they use the knowledge of the application and its data units being communicated.

5.3 Implementation of ALTPs

One of the important issues concerning transport protocols, is how to implement them within a given host operating system on a given host architecture. As network data rates increase, these issues become more important and difficult to address. The source of the difficulty to achieve an efficient implementation is that three entities, the operating system, physical host architecture, and protocols need to be properly integrated together. For example, the physical host architecture and the low level operating system should be such as to allow high bandwidth paths within the host, suitable addressing capability to refer to local and remote objects, objects which are compatible in size and semantics to the objects network protocols understand, caching of remote objects, and allow per packet processing with little operating system overhead.

These issues are considered in more detail in [24], but those relating to the implementation of ALTPs will be outlined here:

- The transport protocols (ALTPs) and internet protocol (CIP) will be simple enough to be largely implemented directly in VLSI hardware, and the architecture will support a direct path from the communications link to the host memory or IOP. This will allow minimum interaction from the host CPU and operating system once a transfer is initiated, and packets will move at network speed directly to memory pages, virtual memory slots, or I/O controller cache blocks.

- The encapsulation/decapsulation and associated copying of data between layers of the standard communications models must be avoided. Any manipulation of packet headers and trailers will be performed on-the-fly, as packets move through the communications processor (CMP). This suggests a pipelined design for the data path of the CMP, but the number of stages in the pipe must be sufficiently low to meet latency targets.

  Bits in the packet header can indicate whether the packet is a control packet that must be processed by the CIP or ALTP protocol modules residing on the CPU, that is, the host software portions of the protocols. As part of connection establishment, the CPU protocol modules will set up the appropriate paths from the CMP to memory, IOP, *etc.* Subsequent packets will have protocol bits in the header cleared, and will proceed directly to the appropriate destination without intervention of the CPU or the operating system. The CMP hardware will match a connection identifier field, determining to which virtual circuit the packet belongs.

- The CPU protocol modules should be implemented such that a single context switch is sufficient for the communications associated with a single transaction, *i.e.*, when a control packet must be processed by multiple layers of the protocols (*e.g.* CIP and ALTP), this will be accomplished by efficient procedure calls, not involving context switches between multiple processes.

One of the advantages of ALTPs is that protocols understand the data structures used by the applications. This allows the ALTP to optimize the data transport based on these objects. For example, ALTP-FT (file transfer), could relate packet or superpacket size to file block size. Similarly, ALTP-IPC (inter-process communication) could relate packet size to page frame and page slot size, and superpacket size to segment size. This mapping could result in significant performance savings, both due to the data mapping (reduced latency due to less fragmentation/reassembly and the associated buffering), and the resulting control mapping (operating system, CPU protocol module, and CMP control correspondence and concurrency).

Note that buffers for flow control are not needed in some ALTPs, since the data is written directly to the target data structure (*e.g.* memory page for ALTP-IPC or file block for ALTP-FT).

6 Conclusion

In this report we have presented a revised internet model appropriate for very high speed internetworking, called the VHSI model. The VHSI model argues for a number of application-oriented lightweight transport protocols (ALTPs) as opposed to a few general purpose application independent transport protocols. We considered the appropriateness of the recently proposed transport protocols, NETBLT and VMTP, and argued that they are not satisfactory within the VHSI model. As the first step towards creating a few prototype ALTPs, we have presented our thoughts on their design. We believe that an ALTP in VHSI should be such that it avoids end-to-end flow and congestion control at the transport level; its error control is application specific and relatively independent of the end-to-end latency; it is designed to be implemented in VLSI; and finally, its implementation is well integrated with the host hardware and its operating system. It should be noted (should be obvious from the report) that our work on the design of ALTPs is at an early stage, and we have not worked out all details to *prove* the superiority of ALTPs or of our approach.

References

[1] Akhtar, S., "Congestion Control in a Fast Packet Switching Network," *MS Thesis, Department of Computer Science, Washington University in St. Louis,* December 1987.

[2] Cheriton, David, "VMTP: A Transport Protocol for the Next Generation of Computer Systems", *SIGCOMM '86 Symposium: Communications Architectures and Protocols (ACM Computer Communication Review)*, Vol. 16 #3, 1986.

[3] Cheriton, David, "VMTP: Versatile Message Transaction Protocol", *DARPA Internet Program Protocol Specification*, Defense Advanced Research Projects Agency – Information Processing Techniques Office, RFC-1045, Arlington Va., Feb. 1988.

[4] Chesson, G., "Protocol Engine," *Proceedings of USENIX Conference,* Spring 1986.

[5] Chesson, G., Brendan E., Vernon S., Andrew C., and Al Whaley, "XTP Protocol Definition", Revision 3.1, Protocol Engines, Inc., PEI 88-13, Santa Barbera, Calif., 1988.

[6] Chesson, G., "XTP/PE Overview", Protocol Engines, Inc., PEI 88-63, Santa Barbera, Calif., 1988.

[7] Clark, D.D., "Window and Acknowledgement Strategy in TCP," *ARPA RFC 813,* July 1982.

[8] Clark, D. D., Lambert, M., and Zhang, L., "NETBLT: A High Throughput Transport Protocol", *SIGCOMM '87 Symposium: Frontiers in Computer Communications Technology (ACM Computer Communication Review),* Vol. 17 #5, 1987.

[9] Clark, D. D., Lambert, M., and Zhang, L., "NETBLT: A Bulk Data Transfer Protocol", *DARPA Internet Program Protocol Specification*, Defense Advanced Research Projects Agency – Information Processing Techniques Office, RFC-998, Arlington Va., Feb. 1988.

[10] Clark, D. D., "Host-Network Interface Architecture," *Laboratory for Computer Science, MIT Cambridge*, 1987.

[11] Coudreuse, J. P. and Servel, M., "Prelude: An Asynchronous Time-Division Switched Network," *International Communications Conference*, 1987.

[12] "Internet Protocol," *DARPA Internet Program Protocol Specification*, Defense Advanced Research Projects Agency – Information Processing Techniques Office, RFC-791, Arlington Va., Sep. 1981.

[13] "Transmission Control Protocol," *DARPA Internet Program Protocol Specification*, Defense Advanced Research Projects Agency – Information Processing Techniques Office, RFC-793, Arlington Va., Sep. 1981.

[14] De Prycker, M., and Bauwens, J., "A Switching Exchange for an Asynchronous Time Division Based Network," *International Communications Conference*, 1987.

[15] Dieudonne, M. and Quinquis, M., "Switching Techniques Review for Asynchronous Time Division Multiplexing," *International Switching Symposium*, 3/87.

[16] Farber, D.J., "Some Thoughts on the Impact of Ultra-High-Speed Networking on Processor Interfaces," *University of Pennsylvania (Private Communication)*.

[17] Haserodt, K. and Turner J.S., "An Architecture for Connection Management in a Broadcast Packet Network," Washington University Computer Science Department, WUCS-87-3.

[18] Huang, A. and Scott K., "Starlite: a Wideband Digital Switch," *Proceedings of Globecom 84*, 12/84, 121–125.

[19] Kulzer, J. J. and Montgomery, W.I., "Statistical Switching Architectures for Future Services," *Proceedings of the International Switching Symposium*, 5/84.

[20] Miller, T., "Internet Reliable Transaction Protocol: Functional and Interface Specification", *DARPA Internet Program Protocol Specification*,

Defense Advanced Research Projects Agency – Information Processing Techniques Office, RFC-938, Arlington Va., Aug. 1985.

[21] Parulkar, G.M., Turner, J.S., "Towards a Framework for High Speed Communication in a Heterogeneous Networking Environment," *Technical Report WUCS-88-7, Department of Computer Science, Washington University in St. Louis,* 1988.

[22] Postel, J. "Internet Control Message Protocol," *DARPA Internet Program Protocol Specification*, Defense Advanced Research Projects Agency – Information Processing Techniques Office, ARPA RFC 792.

[23] Reynolds, J., Postel, J., "Official Internet Protocols," *DARPA Internet Program Protocol Specification*, Defense Advanced Research Projects Agency – Information Processing Techniques Office, ARPA RFC 1011, May 1987.

[24] Sterbenz, J., "High Performance Host and Network Interface Architecture," *Technical Report WUCS-88-?, Department of Computer Science, Washington University in St. Louis, (in preparation),* 1988.

[25] Turner, Jonathan S. "Design of an Integrated Services *Packet* Network," *IEEE Journal on Selected Areas in Communication* 11/86, pages 1373–1380.

[26] Turner, Jonathan S. "Design of a Broadcast Packet Network," *Proceedings of Infocom*, 4/86.

[27] Turner, J.S., "Advanced Communications Systems," Technical Report WUCS-88-15, Department of Computer Science, Washingotn University in St. Louis, 1988.

[28] Velten, David, Robert Hinden, and Jack Sax, "Reliable Data Protocol", *DARPA Internet Program Protocol Specification*, Defense Advanced Research Projects Agency – Information Processing Techniques Office, RFC-908, Arlington Va., Aug. 1984.

[29] Yeh, Y.S., Hluchyi, M.G., Acampora, A.S., "The Knockout Switch: a Simple Modular Architecture for High Performance Packet Switching," *International Switching Symposium,* 3/87.

[30] Jacobson, Van, "Congestion Avoidance and Control", *SIGCOMM '88 Symposium: Communication Architectures Protocols (ACM Computer Communication Review)*, Vol. 18 #4, August 1988.