Issues in Designing the Next Generation Transport Protocol

Dheeraj Sanghi
Department of Computer Science & Engineering
Indian Institute of Technology Kanpur
dheeraj@iitk.ac.in

Abstract

In this paper, we discuss the need for the next generation transport layer. TCP and UDP were designed over 20 years ago. There have been significant changes in the Internet since then in terms of size, types of applications, quality of links, technology, and heterogeneity of the network. Several new transport protocols have been proposed in the last two decades. But most of them were designed to handle one class of applications in a specific network environment. There haven't been many attempts at designing a general purpose protocol like TCP or UDP. Ever since IETF decided the IP Version 6 standard, some researchers have felt the need for next generation transport protocol. In this paper, we take a fresh look at the issues involved for designing a new transport protocol.

Introduction

Over the last two decades there has been an exponential growth in the number of systems connected to the Internet. The network bandwidth has gone up by orders of magnitude. The computing power of each Internet node has gone up considerably. Applications have become a lot more sophisticated, and have started demanding quality of service. The network is heterogeneous with wireless links playing an increasing role. The Internet Protocol is about to change in the next few years. In wake of all these changes, the existing transport protocols are unable to provide the requisite quality of service.

For example, today's network technologies have very low bit-error rates. As a result it is unnecessary today to do error checking by means of CRC checksums or forward error correction (FEC), etc.

Another problem with today's transport protocols is that they are reaching their limits of operation. We are not the first researchers to come to this conclusion. There have been several others who have pointed out such potential problems, and even discussed the solutions (Carlson & Ficarella 1997). But we believe that most of these solutions are at best a temporary relief, and we must start designing a new protocol.

In the last two decades, there have been a large number of transport protocols proposed by various researchers. These include NETBLT, VMTP, DTP, etc. But most of these protocols were designed for specific environments. An excellent review of transport protocols is given in (Iren *et al.* 1997). NETBLT (Clark, Lambert, & Zhang 1987) was designed to do high speed data transfer over long-delay links. VMTP (Cheriton 1986) was primarily meant to be used over a LAN. DTP (Sanghi & Agrawala 1992) was an experimental protocol to test a new flow control algorithm. XTP (Sanders & Weaver 1990) was designed with the view that it would be implemented in hardware, and it supported multicasting. The design of OSI's TP4 has been very much influenced by TCP, and conceptually, there is very little innovation in it.

Problems with Existing Protocols

As we mentioned above, the existing protocols are reaching their limits of use in today's Internet. For example, TCP has a window size of only 16 bits. This means that one can have at most 64 KB of data in transit at a time, which is too little for today's gigabit networks. Similarly, there is a danger of a 32-bit sequence number wrapping around.

The port numbers are only 16-bits. If there is an active server serving a large number of clients, it is conceivable that it may run short of port numbers.

Another problem with TCP is its congestion control mechanism, particularly in presence of "random losses" such as those present on wireless links. TCP assumes that a loss of packet indicates congestion in the network, and closes its window down to one packet. And then the window is increased slowly after each round-trip time. Fast recovery algorithm can recover from a single such random loss, but a temporary bulk loss will result in poor performance for a significant period of time.

There are other issues like implementation of selective ack (Mathis *et al.* 1996), initial window size in high-delay/ high-bandwidth networks (Allman *et al.* 1998), sequence number wrap-around time in very high speed networks (Jacobson, Braden, & Zhang 1990), unsatisfactory estimation of round-trip time, etc.

For many applications like RPC, UDP is too unreliable, and TCP provides reliability at excessive cost. We need a protocol that can provide right kind of service for such transactions (Braden 1992).

The three-way handshake to establish a connection introduces a time lag of one round-trip time before data can be sent. This creates performance problems in high-delay networks like satellite channels. Moreover the congestion control mechanisms of TCP (Stevens 1997) takes too long a time to recover from a transient lossy behavior.

There are some security issues also with using existing transport protocols (Bellovin 1989). For example, in "SYN attack," one machine can send a series of bogus SYNs. The host will be spending all the CPU time in responding to these SYN packets, and not be able to do anything else.

Although both UDP and TCP are generally independent of the network layer protocol, IPv6 (Deering & Hinden 1995) will require some changes in these protocols. For example, the pseudo-header used in computing the checksum will be different. Also, IPv6 provides for a huge packet size often referred to as *jumbograms*. But the MSS option and Urgent Pointer field in TCP do not permit a size greater than 64 KB. Similarly, packet size in UDP cannot be greater than 64 KB. Some changes need to be made to these protocols to handle large packets.

Though solutions for most of these problems are available, they make the current TCP too patched up. Many of these patches will take a very long time to distribute leaving the network in a highly heterogeneous state.

Apart from improvements in the above shortcomings of TCP (and some of UDP) we would like to add more features to the new version of transport protocol which we will henceforth refer to as TPng. We would like to provide new types of reliability (e.g., losses are acceptable but not out of sequence packets), and support transaction based applications. The new protocol should be completely independent of IPv6 or any other lower layer protocol.

Issues and Requirements

This section covers the requirements of TPng. It may be noted that some of these requirements are already met with the extensions proposed for TCP protocol. But the problem with these extensions is twofold. They make the protocol and its implementation very complicated, and therefore, open to bugs and security holes. Second, TCP can only have a limited number of options. If we were to use several options, then there is not sufficient space in the header for all of them. Therefore, a new protocol becomes very important.

Reliability

Currently, the term *reliability* means only one thing: no losses, no duplicates, and no out-of-sequence packets. The other option is unreliable service, where all these three errors can take place.

Different applications desire a level of reliability which is between the two extremes. Multimedia services usually require that packets be delivered in-sequence and without duplicates, but they can handle a few losses. Some applications can handle controlled losses. For example, if they are using redundancy to do forward-error-correction, then there is no need for retransmissions unless the losses are higher than a certain threshold. An application like NFS can handle duplicates, and out-of-sequence packets, but there should be no losses. A transaction-oriented service does not want any duplicates, but can live with out-of-sequence packets.

We will like TPng to support a variety of reliability options.

Connection Oriented vs. Connectionless

As there are enough applications that require one type of service or the other, both should be supported by TPng. In addition, a transaction-type service on the lines of T/TCP (Braden 1994) is also desirable.

Out-of-Band Data

There are very few applications which require use of *out-of-band* or *expedited* data, and in any case, applications can open another connection to send important data. Considering this, we have decided to not support out-of-band data in TPng.

Acknowledgments

We have decided on positive acknowledgments. Indeed it can be shown that negative acknowledgments can at best be used to speed-up retransmission, but cannot be relied upon to indicate the arrival of the packet at the other end. We have decided that we will use cumulative acknowledgment by default, but selective ack will be supported as optional header. Our decision is similar to what has been decided for TCP, but with a difference. The implementor can decide to send acks for several blocks of data. Also, there will be a bit in the header to permit no acknowledgment also, similar to UDP behavior.

IPv6 Related Issues

We would not like to make TPng dependent upon IPv6. But we would like TPng to exploit all the features supported by IPv6. In particular, TPng must be able to support *jumbograms*, that is packets larger than 64 KB.

Checksum

With improvements in the communication technologies, the bit-error rates have been going down, and many applications like audio and video, are able to withstand small errors. It should be possible for an application to suggest whether checksum should be used or not *on a per-packet basis*. Even within a packet, some data may be more important than the rest. If we are using hierarchical encoding for video, e.g., it is important to detect errors in the base encoding, but not so important to detect errors in higher layers of encoding. We propose that there be a field in the header which suggests how many data bytes are covered by the checksum.

Flow Control

As the sliding window based flow control is fairly well understood, we have decided to stay with it. However, if the lower layer can provide information about the state of the network, that should also be used to control the flow of packets. Also, it should be possible to space the packets based on applications' requirements.

Byte-stream versus Records

TCP currently supports a byte-stream and UDP supports records. TPng will support both possibilities.

Operation over Slower Links

If we want to implement all the features, the header size will become very large. Thus the operation over slower links may be hampered. The principles of header compressions that have been used for TCP/IP operations over slower links will work with TPng as well.

Header Structure

In this section, we will look at what fields should be there in the header, what should be their sizes, and their respective placements. Before we get into the details, some of the principles we shall follow are:

- Any information which is not needed in a majority of packets, will not be kept in the mandatory part of the header. Such information should be sent in the optional header as and when needed.
- Sizes of various fields should be determined keeping in mind future extensibility.
- All the fields should be aligned on their natural boundaries. So, a 64-bit field should be aligned on a 64-bit boundary, and so on.

Fields in the Header

Following fields should be present in the header:

- **Version:** The protocol should have a version number in the header. This will permit future extensions of the protocol easily. We propose a 4-bit field in the very beginning of the header.
- **Packet Size:** We should be able to support large packet sizes. The size will be in terms of bytes. We propose a field of 28-bits, which will support packets of 256 MB. The reason for choosing this size is that we can use the first 32 bits for both version number and packet size fields.
- **Header Size:** An 8-bit field is too small. A 16-bit field will be sufficient for the header size, including the options.
- **Port Numbers:** A 16-bit port number may not be enough in the near future. Therefore, we have decide to increase this to 32 bits. There will be two fields, *source port* and *destination port*.
- **Sequence Number:** TCP has a 32-bit sequence number, which is not safe for operation in gigabit networks. The sequence number may wraparound in time less than the time for which a packet is assumed to "live" in the network. We are proposing a sequence number field of 64 bits.
- **Acknowledgment Sequence Number:** This field will also have to be a 64-bit one.
- **Window Size:** Since a 16-bit window size is too restrictive in long-fat pipes, we are proposing to increase it to 32 bits.
- **Checksum:** For sending the checksum, we are proposing the following fields.
  - **Checksum:** There will be a 16-bit field for sending the computed value of checksum. We will continue to use the CRC-16 algorithm for this.
  - **Checksum Length:** This will suggest how much of the packet is covered by the checksum. It should be possible to cover a packet only partially. This will be a 14-bit field. If an application wants more than 16 KB of a packet to be covered, then the entire packet will be covered.
  - **Checksum Flags:** There will be two flags that will distinguish between the following cases:
    - No checksum is present
    - Checksum covers header only
    - Partial checksum
    - Checksum covers entire packet
- **Service Bits:** There will be 16 service bits or flags which will control the operation of the protocol or meaning of some of the fields. Some of the bits that will be present include:

SYN Same meaning as in TCP.

RST Same meaning as in TCP.

FIN Same meaning as in TCP.

ACK If set, acknowledgment is meaningful. May not be set if the application is not interested in reliability.

SAK If set then the acknowledgment is selective. Please note that by default the acks are cumulative, and it is possible to send a cumulative ack in the header and send a selective ack in the options.

```
    4          16                   32                   48                   64
+++++++++++++++|+++++++++++++++|+++++++++++++++|+++++++++++++++|
|Ver|        Packet Size       |  Header Size  |   Service     |
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|          Source Port         |       Destination Port        |
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|                       Sequence Number                         |
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|                    Acknowledgment Number                      |
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|          Window size         |CO| Cksum Len |   Checksum      |
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|                          Options(????)                        |
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|                       Data                                    |
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Here each `+´ sign represents a bit.
```

Figure 1: Proposed Header Structure

REC If set then the protocol is record based. The peer should give the entire packet to the application as one. Otherwise the protocol is byte-stream oriented.

FLW If set then flow control is to be used. The Window size field assumes significance. If not set, then no flow control is used.

ORD If set, then peer should give packets to applications in sequence only. Otherwise, packets may be delivered in any order.

LOS If set, then TPng should recover from losses. If not set, then application is asking for best-effort service.

TRX If set, then the transaction-based service is desired. The peer should not attempt a 3-way handshake if information about the past connection is cached.

There may be more such flags defined in the course of development.

- **Options:** These are zero or more options each of length in multiples of 32-bits and in the TLV (Type-Length-Value) format. Each option will be assigned a unique 8-bit number. The second byte will indicate the length of the option in multiple of 32-bits. So an option can be upto 1 KB. The remaining part is the data associated with the option.

  The options that we are currently considering include a connection count to be used for transaction-oriented service, a maximum segment size option similar to TCP, selective ack option, and an echo option which can be used to implement better timers.

Header Structure

We present the proposed header structure in Figure 1. As we may note, the minimum size of the header is 40 bytes, which is double that of TCP, but considering that most of the increase comes from the increased size of some important fields, it is quite acceptable.

Conclusion

In this paper, we have discuss the need for the next generation transport layer. We have first discussed the problems with TCP and UDP. Having established the need for a new protocol, we presented the issues that one needs to consider while designing the protocol, including the various services and features that should be supported by this protocol. We presented specific decisions that we have taken while designing this protocol. Finally, we have presented the proposed header structure.

We are planning to work out the details of message types, events, and a state-diagram for the protocol. We will later implement it, and study its performance.

References

Allman, M.; Glover, D.; Griner, J.; Scott, K.; and Touch, J. 1998. Ongoing TCP research related to satellites. draft-ietf-tcpsat-res-issues-02.txt. Work in progress.

Bellovin, S. M. 1989. Security problems in the TCP/IP protocol suite. *ACM Comp. Commun. Review* 19(2).

Braden, R. 1992. Extending TCP for transactions – concepts. RFC 1379.

Braden, R. 1994. T/TCP: TCP extensions for transactions: Functional specification. RFC 1644.

Carlson, R., and Ficarella, D. 1997. Six virtual inches to the left: The problem with IPng. RFC 2140.

Cheriton, D. R. 1986. VMTP: A transport protocol for the next generation of communication systems. In *Proceedings of ACM SIGCOMM '86*, 406–415.

Clark, D. D.; Lambert, M. L.; and Zhang, L. 1987. NETBLT: A high throughput transport protocol. In *Proceedings of ACM SIGCOMM '87*, 353–359.

Connolly, T.; Amer, P.; and Conrad, P. 1994. An extension to TCP: Partial order service. RFC 1693.

Deering, S., and Hinden, R. 1995. Internet Protocol Version 6 specifications. RFC 1883.

Iren, S.; Amer, P. D.; Conrad, P. D.; Golden, E.; and Marasli, R. 1997. The transport layer: Tutorial and survey. Technical Report TR No. 97-16, U. of Delaware, Dept. of Comp. & Info. Sc.

Jacobson, V.; Braden, R.; and Zhang, L. 1990. TCP extensions for high-speed paths. RFC 1185.

Mathis, M.; Mahdavi, J.; Floyd, S.; and Romanow, A. 1996. TCP selective ack options. RFC 2018.

Sanders, R. M., and Weaver, A. C. 1990. The Xpress Transfer Protocol (XTP) — A Tutorial. *Computer Communication Review* 20(5):67–80.

Sanghi, D., and Agrawala, A. K. 1992. DTP: An efficient transport protocol. In *Proc. of Int'l Conf. on Comp. Networks, Arch. and App. (Networks '92)*.

Stevens, W. 1997. TCP slow start, congestion avoidance, fast transmit, and fast recovery. RFC 2001.