Network Working Group:
Request for Comments: 1705
Category: Informational

R. Carlson
ANL
D. Ficarella
Motorola

October 1994

Six Virtual Inches to the Left: The Problem with IPng

Status of this Memo

This memo provides information for the Internet community. This memo does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Abstract

This document was submitted to the IETF IPng area in response to RFC 1550. Publication of this document does not imply acceptance by the IPng area of any ideas expressed within. Comments should be submitted to the big-internet@munnari.oz.au mailing list.

Overview

This RFC suggests that a new version of TCP (TCPng), and UDP, be developed and deployed. It proposes that a globally unique address (TA) be assigned to Transport layer protocol (TCP/UDP). The purpose of this TA is to uniquely identify an Internet node without specifying any routing information. A new version of TCP, and UDP, will need to be developed describing the new header fields and formats. This new version of TCP would contain support for high bandwidth-delay networks. Support for multiple network layer (Internet Protocol) protocols is also possible with this new TCP. Assigning an address to TCP/UDP would allow for the support of multiple network layer protocols (IPng’s). The TA would identify the location of an Internet node. The IPng layer would provide routing information to the Internet. Separating the location and routing functions will greatly increase the versatility of the Internet.

Table of Contents

1. Introduction

For more than a decade, network engineers have understood the benefits of a multi-layer protocol stack. However, during its development, the Transmission Control Protocol (TCP) was strongly linked to the Internet Protocol (IP) [Postel, 1981a]. When the TCP/IP protocol suite was developed, two important ideas were implemented. The first was that each host would be uniquely identified by a network layer number (i.e., IP number = 192.0.2.1). The second was to identify an application with a transport layer port number (i.e., TCP DNS number = 53). For host-to-host communications, the IP and port numbers would be concatenated to form a socket (i.e., 192.0.2.1.53). While this has lead to a very efficient and streamlined TCP layer, it has tightly coupled the TCP and IP layers. So much so, in fact, that it is nearly impossible to run TCP over any network layer except for

IP.

The motivation for writing this paper resulted from research into the various Internet Protocol Next Generation (IPng) proposals put forth by various IETF working groups. Each of the IPng proposals strives to solve the impending IP address exhaustion problem by increasing the size of the address field. They all allude to modifications to TCP and User Datagram Protocol (UDP) to make them capable of supporting a new network layer IPng protocol. The authors of this paper feel that this points to an inherent TCP/IP design flaw. The flaw is namely that the transport (TCP) and network (IP) layers are not protocol independent. In this paper, we will propose a new TCP and UDP implementation that will make the transport and protocol layers independent and thus allow for any of the IPng protocols to operate on the same internet without any further modification to the higher layer protocols. TCP, and UDP would become extremely powerful Application Programming Interfaces (APIs) that operate effectively over multiple network layer technologies.

2. Historical perspective

2.1 OSI and the 7 layer model

Present day computer and communication systems have become increasingly heterogeneous in both their software and hardware complexity, as well as their intended functionality. Prior to the establishment of computer communications industry standards, proprietary standards followed by particular software and hardware manufacturers prevented communication and information exchange between different manufacturers products and therefore lead to many "closed systems" [Halsal, 1992] incapable of readily sharing information. With the proliferation of these types of systems in the mid 1970s, the potential advantages of "open systems" where recognized by the computer industry and a range of standards started to be introduced [Halsal, 1992].

The first and perhaps most important of these standards was the International Standards Organization (ISO) reference model for Open Systems Interconnection standard (OSI), describing the complete communication subsystem within each computer. The goal of this standard model was to "allow an application process in any computer that supports a particular set of standards to communicate freely with an application process in any other computer that supports the same standards, irrespective of its origin of manufacture" [Halsal, 1992]. The last statement above describes the OSI 7-layer model which has now, in concept, become the fundamental building block of computer networks. Though there are arguably no present day computers and networks completely compliant to all 7 layers of the

OSI protocol stack, most protocol stacks do embrace the fundamental concept of independent layers, thus allowing the flexibility for computers operating with dissimilar protocol stacks to communicate with one another.

Take for example, the datalink layers as supported by TCP/IP. TCP/IP will run equally well in either the local area network (LAN) or wide area network (WAN) environments. Even though the LAN may use Ethernet 802.3 and the WAN may use T1 serial links. This function was designed to present a "standardized set of network functions (i.e., a logical network)", to the upper network layer, "regardless of the exact details of the lower level implementations" [Meyer, Zobrist, 1990].

2.2 Internet Evolution

"The internet architecture, the grand plan behind the TCP/IP protocol suite" was developed and tested in the late 1970s, [Braden, et al, 1991] and but for the addition of subnetting, autonomous systems, and the domain name system in the early 1980s and the more recent IP multicasting implementation, stands today essentially unchanged. Even with the understood benefits of a multi-layer protocol stack, all steps taken to enhance the internet and its services have been very incremental and narrowly focused.

2.3 The Reasons for Change

The reasons for change from IP to IPng can be described in terms of problems for which the current IP will simply become inadequate and unusable in the near future (~2-4 years). These problems are the exhaustion of IP class B address space, the exhaustion of IP address space in general, and the non-hierarchical nature of address allocation leading to a flat routing space [Dixon, 1993].

2.3.1 Class-B Address Exhaustion

One of the fundamental causes of this problem is the lack of a class of network address appropriate for a mid-sized organization. The class-C address, with a maximum of 254 unique host addresses is to small, while class-B, with a maximum of in excess of 65 thousand unique host addresses is to large [Fuller, et al, 1992]. As a result, class-B addresses get assigned even though nowhere near the number of available addresses will ever get used. This fact, combined with a doubling of class-B address allocation on a yearly basis lead the Internet Engineering Steering Group (IESG) to conclude in November, 1992 that the class-B address space would be completely exhausted within 2 years time. At that point, class-C addresses would have to be assigned, sometimes in multiples, to organizations needing more than the 254 possible host addresses from a single

class-C address [Almquist, Gross, 1992].

2.3.2 Routing Table Growth

Based on research conducted by the IESG in November 1992, definite routing table size explosion problems were identified. Namely, it was determined that current router technology at that point could support a maximum of 16,000 routes, which in turn could support the internet for an additional 12 to 18 months (~May, 1994) at the then twofold annual network growth rate. However, vendor router maximum capabilities were in the process of being increased to 64,000 routes, which at the two-fold annual network growth rate, could bring us an additional 2 years of lead time, (at best bringing us to May, 1996, and at worst to November, 1995) assuming the class-B address exhaustion problem mentioned above could be solved in the interim [Almquist, Gross, 1992].

As a short term, incremental solution to this routing table growth problem, and to aid in the class-B address exhaustion problem the IESG endorsed the CIDR supernetting strategy proposal (see RFC-1338 for full details of this proposal). However, this strategy was estimated to have a viability of approximately 3 years, at which point the internet would run out of all classes of IP addresses in general. Hence, it is clear that even CIDR only offers temporary relief. However, if implemented immediately, CIDR can afford the Internet community time to develop and deploy an approach to addressing and routing which allows scaling to orders of magnitude larger than the current architecture (IPng).

3. The Problems with Change

There are many problems, both philosophical and technical, which greatly contribute to the difficulties associated with a large scale change such as the one proposed in the conversion from IP to IPng. These problems range from having to rewrite highly utilized and entrenched user applications, such as NFS, RPC, etc, to potentially having to invest additional capital to purchase hardware that supports the new protocol(s). This proposal solves the urgent internet problems listed above, while simultaneously limiting the amounts of retraining and re- investing that the user community would have to undertake. The TCP layer will once and for all be changed to support a multiprotocol internet. The net affect is that while administrators will necessarily be trained in the operations and details of this new TCP, the much larger operator and end user community will experience no perceptible change in service and network usage.

3.1 TCP/UDP Implementations

Both TCP and UDP are highly dependent on the IPv4 network layer for 2 very low level reasons. 1) a TCP/UDP socket is formed by concatenating a network layer address (IP address) and the transport layer TCP/UDP port number. 2) included in the TCP/UDP checksum calculation are the IP layer source and destination addresses mentioned above which are transferred across the TCP/IP [Postel, 1981b] or UDP/IP [Postel, 1980] interfaces as procedure call arguments. It should be noted at this point that the reason for such strong dependence between the transport and network layers in TCP/IP or UDP/IP is to insure a globally unique TCP/UDP layer address, such that a unique connection could be identified by a pair of sockets. The authors of this paper propose that the IP address requirement with TCP and UDP be replaced with a globally unique transport address (TA) concatenated with a transport layer port address. This solution offers the capability to still maintain a globally unique address and host unique port number with the added benefit of eliminating the transport and network layer dependence on one another.

3.2 User Applications

In addition to TCP and UDP, there are a large number of firmly entrenched higher level applications that use the IP network layer address embedded internally, and would therefore require modification for use with the proposed IPng network layer schemes. These applications include, but are not limited to Network File System (NFS), Remote Procedure Call (RPC), and File Transfer Protocol (FTP). All of these applications should be modified to use the Internet Domain name to identify the remote node, and not an embedded, protocol dependent IP address.

3.3 The Entrenched Masses

Will users voluntarily give up their IPv4 systems to move to IPng? It seems likely that many users will resist the change. They will perceive no benefit and will not install the new software. Making the local Internet contact responsible may not be feasible or practical in all cases. Another issue is backward compatibility issues. If a host needs to run IPng and IPv4 to support old hosts, then 1) where is the address savings IPng promised. 2) Why change if the host you are talking to has IPv4 anyway?

On the other hand, replacing the existing TCP (TCPv6) with this new version (TCPng) will benefit users in several ways. 1) Users will be able to connect to unmodified TCPv6 hosts. 2) As nodes upgrade to TCPng, new features will be enabled allowing TCP to communicate effectively over high bandwidth*delay network links. 3) System

administrators will be able to incrementally upgrade nodes as needed or as local conditions demand. 4) Upgraded nodes may return their IPv4 address and use an IPng address and TCP transporter function, described later, to communicate with IPv4 hosts.

4. Making TCP & UDP Protocol Independent

The OSI 7 layer model specifies that each layer be independent of the adjacent layers. What is specified is the interface between layers. This allows layers to be replaced and/or modified without making changes to the other layers. As was pointed out previously, the TCP and UDP transport layers violate this precept. In the following discussion, when we refer to TCP we mean both the TCP and UDP protocols. The generic term transport layer and TCP will be used interchangeably.

Overcoming TCP's dependence on IP will require changes to the structure of the TCP header. The developers and implementors will also have to change the way they think about TCP and IP. End users will also have to change the way they view the Internet. Gone will be the days when Internet node names and IP addresses can be used interchangeably. The goal of this change is to allow end users to migrate from the current IPv4 network layer to an IPng layer. What this IPng protocol is will be left to the Internet Architecture Board/Internet Engineering Steering Group/Internet Engineering Task Force (IAB/IESG/IETF) to decide. By adopting this proposal, the migration will be greatly enhanced.

One of the stated goals of the IAB is to promote a single Internet protocol suite [Leiner, Rekhter, 1993]. While this is a laudable goal, we should not be blinded by it. The addition of a Transport layer address (TA) does not invalidate the IAB's stated goals. It merely brings the implementation into compliance with standard networking practices. The historical reasons for concatenating TCP port numbers to IP numbers has long since passed. The increasing throughput of transmission lines and the negligible effect of packet overhead (see appendix A) prove this. The details of assigning and using TA's are discussed in the next few sections.

4.1 Transport Addressing

A Transport Address (TA) will be assigned to the TCP transport layer on each Internet node. The purpose of this address is to allow a TCP on one node to communicate with a TCP on a remote node. Some of the goals defined in developing this address are:

1. Fixed size -- A fixed size will make parsing easier for decoding stations.

2. Minimum impact on TCP packet size -- This information will need to be carried each TCP packet.

3. Global Uniqueness -- It is desirable (required) to have a globally unique Transport Address.

4. Automatic Registration -- To reduce implementation problems, an automatic registration of the TA is desirable.

The TA will be used when an Internet node attempts to communicate with another Internet node. Conceptually you can view the TA as replacing the IP number in every instance it now appears in the transport layer (i.e., a socket would change from IP#.Port# to TA#.Port#). A connection setup would thus be:

1. A user starts an application on Node-A and requests service from Node-B. The user identifies Node-B by referencing it's Internet Domain Name.

2. The TCP on Node-A makes a Domain Name Service (DNS) call to determine the TA of Node-B.

3. Node-A constructs a TCP packet using the header Src = TA-A.port and Dest = TA-B.port and passes this packet down to the network layer.

4. The IP on Node-A makes a DNS call to determine the IP address of Node-B. The IP will cache this TA/IP pair for later use.

5. Node-A constructs an IP packet using the header Src = IP-A and Dest = IP-B and passes this packet down to the Media Access layer.

6. Delivery of the packet is identical to the delivery of an existing Internet IP packet.

7. The IP on Node-B examines the IP Dest address and if it matches it's own, strips off the header and passes the data portion up to the TCP. (Note: the packet may have passed through several IP routers between the source and destination hosts.)

8. The TCP on Node-B examines the header to determine if the Dest TA is it's own, if so it passes the data to the application specified by the port address. If not it determines if it should perform the transporter function.

The packet will be forwarded toward the destination or an error message will be returned.

The above steps represent a quick synopsis of how user applications may pass data between different Internet nodes. The exact structure of the network is hidden from the application, allowing the network to be modified and improved as needed. Using the transporter function, several different network layers may be traversed when moving from source to destination (several examples are provided in appendix D).

One of the underlying assumptions is that the user application must refrain from making assumptions about the network structure. As pointed out in section 3, this is not the case for the current Internet network. User applications that are deployed with this new TCP must be capable of making this assumption. This means that the user application should store the Internet Domain Name in it's internal structure instead of the IPng network number. The domain name is globally unique and provides enough information to the system to find the transport and network layer addresses. The user application must pass the following parameters down to TCP:

1. Destination domain name (Text string)
2. Pointer to data buffer
3. Quality of service indicators
4. Options

When the user application writes data to the network, TCP will return a nonzero integer to indicate an error condition, or a zero integer to indicate success. When the user application reads data from the network, TCP will deliver a pointer to a data buffer back to the application.

TCP will receive the users request and it will make a DNS call to determine the destination nodes TA. If DNS returns a TA, TCP will build a Transmission Control Block (TCB) (see the paragraph below) and call the network layer. Otherwise, TCP will make a DNS call looking for the destination nodes IPv4 address. If an address is returned, TCP will takes the steps listed below in building a TCB, and call the proper network layer. If DNS returns a host unknown indication, exit back to the user with a "host unknown" error. TCP should maintain a cache of domain names and addresses in lieu of making repeated DNS calls. This feature is highly recommended, but not required.

The state information needed to keep track of a TCP connection is kept in the Transmission Control Block (TCB). Currently this structure has fields for the TCP parameters, source port, destination

port, window size, sequence number, acknowledgment number, and any TCP options. The network layer source and destination IP numbers are also stored here. Finally, the status of the connection (LISTEN, ESTABLISHED, CLOSING, of the TCP parameters and include the new source and destination Transport addresses. The existing space for the IPv4 addresses will be left in place to allow for backward compatibility. The IPv4 fields will be used if the source is communicating directly with an unmodified TCP/IP host.

The existing status indicators will remain with their meaning unchanged. Connection setup will retain the current 3-way handshake. When performing transporter functions, TCP will NOT build a TCB, unless the destination is an unmodified IPv4 host (see appendix D). The TCP connection remains an end-to-end reliable transport service, regardless of the number of intermediate transporter nodes.

TCP will build an old or new header (defined below) placing the user application data in the data field. If TCP is communicating directly with an unmodified IPv4 host, the existing TCP header (STD 7, RFC 793) will be used for comparability reasons. If the destination host is an unmodified host, and an intermediate transporter node is being used, this new TCP header must be used with the 'C' bit set to 1. The destination TA will be set to the IPv4 address, and the packet will be delivered to the transporter node. If the destination host is modified with this new TCP, the destination address will be set to the TA and the packet will be delivered, possibly through a transporter, to the remote host.

TCP will communicate with it's underlying network layer(s) to deliver packets to remote hosts. The Internet Assigned Number Authority (IANA) will assign unique identifiers to each network layer TCP will support. TCP will maintain a cache of TA's and IANA network layers numbers, to allow support of multiple network layers. When TCP wishes to send data, it will consult this cache to determine which network to send the packet to. If the destination TA is not in this cache, TCP will send a request to each of it's network layer(s) asking if they know how to deliver data to this TA. All of the network layers supported by the sending host will be probed, in an order defined by the system administrator, until one responds 'yes' or they all have said 'no'. The first layer to say 'yes' will be used. If no path exists, an error message will be returned to the user application. Once a network layer is identified, TCP will communicate with it by passing the following parameters:

1) Destination address (TA or IPv4).
2) A pointer to the data buffer.
3) Options.

The network layer will use the destination address as an index into a cache to determine the network address to send to. In the entry is not in the cache, it will make a DNS call to determine the network address and a cache entry will be build (see appendix D). It is mandatory that a cache be maintained. If a host is attached to several different networks (i.e., a transporter) each layer will maintain it's own cache.

When IP receives a data packet from a remote node, it will strip off the IP header and pass a pointer to the data buffer up to TCP. IP will also supply TCP with it's IANA network layer number. TCP may use the source TA and the IANA number to update it's cache.

The structure of a TA is to concatenate a unique manufacture code with a manufacturer defined variable to form a unique 64 bit number. The unique manufacture code will be a 24 bit number, possibly the same code as the IEEE 802.3 MAC address code. The remaining 40 bits will be supplied by the manufacture to uniquely identify the TCP. It is recommended that this field be built by encoding the manufacturer's serial number. An integer serial number will be viewed as an integer number and converted into it's hexadecimal equivalent, left padded with 00 octets if necessary. If a serial number contains Alpha characters, these alpha characters will be converted into octets using the international standard ASCII code. The integer values will then be converted to their hexadecimal equivalent and the 2 values will be concatenated to form the unique identifier. These structure will allow 2^24 (16,777,216) manufactures to build 2^40 (1,099,511,627,776) transport addressable entities. Each of these entities may have 1 or more network interfaces using IPv4, IPng, or any other network layer protocol.

The current growth of the Internet may indicate that this amount of address space is inadequate. A larger fixed space (i.e., 96 or 128 bits) or a variable length field may be required. The disadvantage is that this address must be transmitted in every packet.

4.2 TCPng

The new TCP header is as shown.

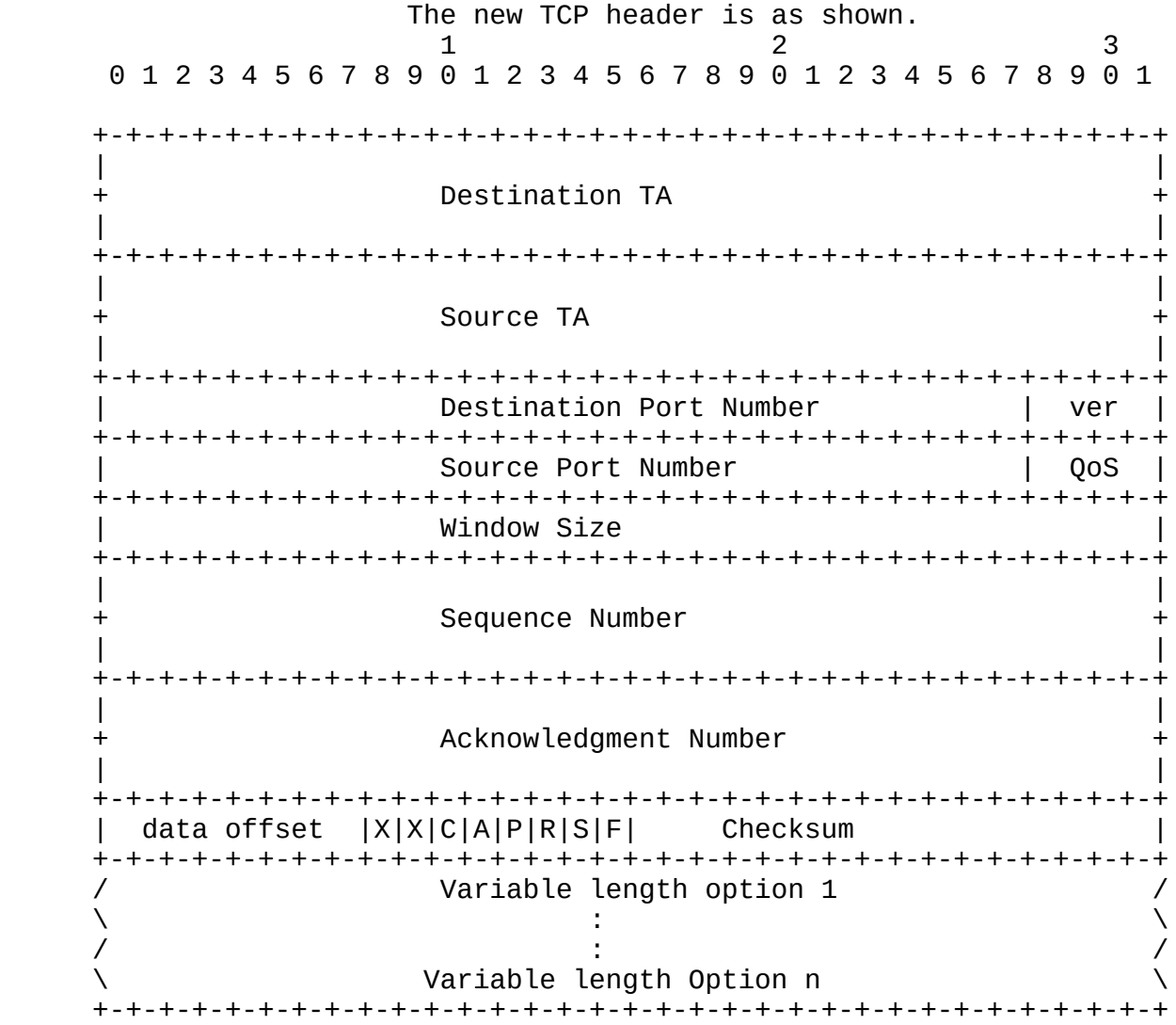


Figure 1

Destination TA: 64 bits.
The Destination Transport Address. The concatenation of the 24 bit IEEE assigned Ethernet address and the 40 bit representation of the machines serial number for the remote node.

Source TA: 64 Bits.
The Source Transport Address. The concatenation of the 24 bit IEEE assigned Ethernet address and the 40 bit representation of the machines serial number for the local node.

Destination Port Number: 28 Bits.
Identifies the specific application on the remote node.

Ver: 4 bits.
Version number. This is TCPng. RFC 793 references 9 earlier editions of ARPA TCP. The current TCP is version 10.

Source Port Number: 28 Bits.
Identifies the specific application on the local node.

QoS: 4 bits.
The Quality of Service parameter may be set by the user application and passed down to a network layer that supports different levels of service.

Window: 32 Bits.
The number of data octets beginning with the one indicated in the acknowledgment field which the sender of this segment is willing to accept.

Sequence Number: 64 Bits.
The sequence number of the first data octet in this segment (accept when the S bit is present). If S bit is on, the sequence number is the initial sequence number (ISN) and the first data octet is ISN+1. (The ISN is computed using the existing algorithm).

Acknowledgment Number: 64 Bits.
If the A bit is set, this field contains the value of the next sequence number the sender of this segment is expecting to receive. Once a connection is established, this is always sent.

Data Offset: 8 Bits.
This is the number of 32 bit words in the TCP header. This indicates where the data begins. The TCP header is an integral number of 32 bit words long. The minimum value is 12 and the maximum is 256. If options are used, they must pad out to a 32 bit boundary.

Flags: 8 Bits.
The A, P, R, S, and F flags carry the same meaning as in the current version of TCP. They are:

1. A = Ack, and acknowledgment field significant
2. P = Push, the push function
3. R = Reset, reset the connection
4. S = Sync, synchronize sequence numbers
5. F = Fin, No more data from sender

The C bit, C = Compatibility, is used to indicate that one end of the connection is an unmodified TCP/IP host. When the C bit is set, all header values must conform to the TCPv6 specifications. The source port, destination port, and window size must be 16 bits and the Sequence and Acknowledgment numbers must be 32 bits. These values are stored in the lower half of the proper area with null octet pads filling out the rest of the field.

The 2 X bits, X = Reserved, are not defined and must be ignored by a receiving TCP.

Checksum: 16 Bits.
The checksum field has the same meaning as in the current version of TCP. The current 96 bit pseudo header is NOT used in calculating the checksum. The checksum covers only the information present in this header. The checksum field itself is set to zero for the calculation.

Variable Length Options:
There are two types of options, mandatory and optional. A TCP must implement all known mandatory options. It must also be capable of ignoring all optional options it does not know about. This will allow new options to be introduced without the fear of damage caused by unknown options. An option field must end on a 32 bit boundary. If not, null octet pad characters will be appended to the right of the option. The structure of an option is shown in figure 2 below:

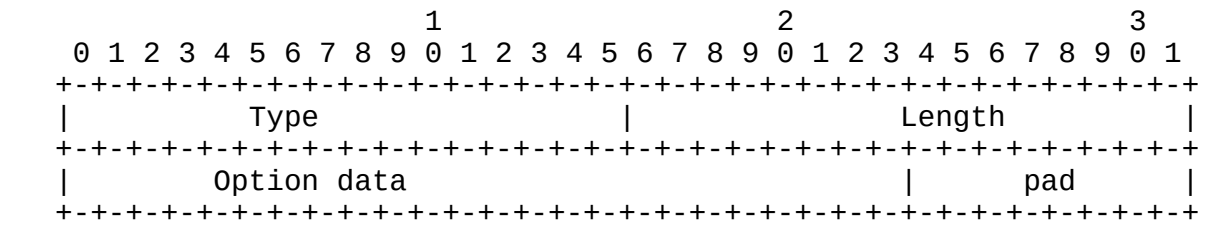


Figure 2

4.3 Mandatory Options

There are three mandatory options defined by this implementation of TCP. Each of these options is implemented using the structure pictured in figure 2 above.

A description of each field follows:

Type: 16 bits
: The type field identifies the particular option.

Length: 16 bits
: The length field represents the size of the option data to follow, in octets.

Option Data: Variable Length
: The option data is of variable length specified by the length field, plus 0-3 bytes of zeros to pad to a 32-bit boundary.

The following are the 3 mandatory options that must be implemented:

Null: 8 bits
: The null option, (type=0) is represented by the bit sequence [00000000], preceded by an additional 8, zero padding bits to fill out the full 16-bit type field. The data may be of any size, including 0 bytes. The option may be used to force an option to be ignored.

Maximum Segment Size: 8 bits
: The maximum segment size option, (type=1) is represented by the bit sequence [00000001] preceded by an additional 8, zero padding bits to fill out the full 16-bit type field. If this option is present, then it communicates the maximum receive segment size at the TCP which sends this segment. This potion is mandatory if sent in the initial connection request (SYN). If it is sent on any other segment it is advisory. The data is a 32-bit word specifying the segment

size in octets [Ullmann, 1993].

Urgent Pointer: 8 bits
The urgent pointer, (type=2) is represented by the bit sequence [00000010] preceded by an additional 8, zero padding bits to fill out the full 16-bit type field. This option emulates the urgent field in TCPv6. The data is a 64-bit sequence number identifying the last octet of urgent data within the segment.

4.4 Optional Options

This version of TCP must be capable of accepting any unknown options. This is to guarantee that when presented with an unrecognized option, TCP will not crash, however it must not reject or ignore any option.

4.5 Compatibility Issues

The Internet community has a large installed base of IP users. The resources required to operate this network, both people and machine, is enormous. These resources will need to be preserved. The last time a change like this took place, moving from NCP to TCP, there were a few 100 nodes to deal with [Postel, 1981c]. A small close knit group of engineers managed the network and mandated a one year migration strategy. Today there are millions of nodes and thousands of administrators. It will be impossible to convert any portion of the Internet to a new protocol without effecting the rest of the community.

In the worst case, users will lose communications with their peers as some systems upgrade and others do not. In the current global environment, this will not be tolerated. Any attempt to simply replace the current IPv4 protocol with a new IPng protocol that does not address compatibility issues is doomed to failure. This reasoning has recently been realized by Ullmann (CATNIP) and he attempts to use translators to convert from one protocol to another (i.e., CATNIP to IPv4). The problem is what to do when incompatible parameters are encountered. Also CATNIP would need to be replaced every time a new network layer protocol was developed.

This proposal attempts to solve these problems by decoupling the transport and network protocols. By allowing TCP to operate over different network layer protocols, we will create a more stable environment. New network layer protocols could be developed and implemented without requiring changes that are visible to the user community. As TCP packets flow from host-to-host they may use several different network layers, allowing users to communicate without having to worry about how the data is moved across the

underlying network.

4.5.1 Backward Compatibility

It may be said that the maturity of a software package can be determined by how much code is required to maintain compatibility with previous versions. With the current growth of the Internet, backward compatibility issues can not be dismissed or added in as an after thought. This version of TCP was designed with backward compatibility in mind. When the TCP communicates with an unmodified IPv4 TCP/IP, it takes steps to insure compatibility. First off it sets a bit in the header indicating that the TCP parameters (ack, seq, port numbers, and window size) use the TCPv6 values. When communicating directly with an unmodified host the existing TCP/IP header is used. Only existing TCP options may be sent as well.

The advantage of this approach is that TCP transporter nodes will not have to make decisions about how to modify packets just passing through. It is up to the source node to build a header that is compatible before sending it. This approach will allow any new TCP to contact and communicate with any unmodified IPv4 host. The source host may have an IPv4 address, or it may send data to a transporter for delivery. The decision will be made based on the source and destination addresses. During connection setup, the location of the destination node is determined and the proper network layer is used to send data.

An existing IPv4 host will be capable of opening a connection to any new TCPng host that is directly connected to the network with an IPv4 protocol stack. If the TCPng host only has an IPng stack, the connection attempt will fail. Some existing batch style services (i.e., Simple Mail Transfer Protocol - SMTP) will continue to work with the help of transporters. Interactive sessions (i.e., Telnet) will fail. Thus, our new TCP is backward compatible, but the existing IPv4 hosts are not forward compatible.

4.6 Level 4 Gateways

The ability to allow hosts with differing network layer protocols to communicate will be accomplished by using a transport layer gateway (called transporter in this paper). The transporter works just like an IP router, receiving TCP packets from one network layer and transporting them over to another. This switching is done by examining the packets source and destination TA's. If a TCP packet arrives with a destination TA that differs from this hosts TA, and the transporter functionality is enabled, the packet should be transported to another network layer. In some cases, the receiving node is a host and not a transporter (i.e., transporter functionality

disabled). In this case the host will discard the packet and return a TCMP (see below) error message.

A transporter is not responsible for reading or formatting the TCP header of packets it receives. The header is simply examined to determine where to deliver the packet. When forwarding, the packet is sent to any of the network layers the transporter supports. The exception is that the packet may not be presented back to the network it was received from. It is the responsibility of the network layer to destroy undeliverable packets. If a transporter is unable to determine which network the packet should be forwarded to, the packet is discarded and a TCMP message is generated and returned to the original source host. Several examples of how transporting works are presented in appendix D.

4.7 Error Conditions

It is recognized that from time to time certain error conditions will occur at some intermediate transporter that will need to be communicated back to the source host. To accomplish this a Transport Control Message Protocol (TCMP) service facility will need to be developed. This protocol will model itself after the Internet Control Message Protocol (ICMP). The operational details are discussed in a separate TCMP document.

5. Advantages and Disadvantages of this approach

This proposal offers the Internet community several advantages. First, TCPng will operate over multiple network layer protocol stacks. Users will be able to select the stack(s) that meets their needs. The problem of IPv4 address exhaustion will be postponed as sites move from IPv4 to IPng protocol stacks. Future IP3g protocol stacks may be designed and deployed without major service disruptions. The increased size of the sequence, acknowledge, and window fields will allow applications to run effectively over high bandwidth-delay network links. Lastly, TCPng will allow applications to specify certain Quality of Service (QoS) parameters which may be used by some network layer protocols (i.e., Asynchronous Transfer Mode - ATM).

This protocol is not without it’s share of design compromises. Among these are a large packet header increased in size from 5 to 12 long words. The addition of a TA means that network administrators must deal with yet another network number that must be globally maintained. Multiple network protocols may add to the complexity of a site’s network. Lastly, is the TA address space large enough so we will not have to rebuild TCP.

6. Conclusions

In this paper, we have reviewed the current status of the Internet society s IPng initiative. We were struck by the enormity of the changes required by those proposals. We felt that a different approach was needed to allow change to occur in a controlled manner. This approach calls for replacing the current TCP protocol with one that does not require a specific IP layer protocol. Once this is in place, various IPng protocols may be developed and deployed as sites require them. Communications between IPv4 and IPng hosts will be maintained throughout the transition period. Modified hosts will be able to remove their IPv4 protocol stacks, while maintaining communications with unmodified hosts by using a TCP transporter.

The title of this paper "Six Virtual Inches to the Left" comes from a talk the author once heard. In this talk an engineer from Control Data Corporation (CDC) told a story of CDC’s attempt to build a cryogenically cooled super computer. The idea being that the power consumption of such a computer would be far lower then that of a conventional super computer. As the story goes, everyone thought this was a great idea until someone pointed out what the power requirements of the cryo system were. The result was that all the assumed power savings were consumed by the cryo system. The implication being that all the power requirements were not saved but simply moved 6 feet from the CPU to the support equipment. The moral being that the entire system should be analyzed instead of just one small piece.

References

[Postel, 1981a] Postel, J., "Transmission Control Protocol - DARPA Internet Program Protocol Specification", STD 7, RFC 793, DARPA, September 1981.

[Halsal, 1992] Data Communications, Computer Networks, and Open Systems.

[Meyer, Zobrist, 1990] TCP/IP versus OSI, The Battle of the Network Standards, IEEE Potentials.

[Braden, et al, 1991] Clark, D., Chapin, L., Cer, V., Braden, R., and R. Hobby, "Towards the Future Internet Architecture", RFC 1287, MIT, BBN, CNRI, ISI, UCDavis, December 1991.

[Dixon, 1993] Dixon, T., "Comparison of Proposals for Next Version of IP", RFC 1454, RARE, May 1993.

[Fuller, et al, 1992] Fuller, V., Li, T., Yu, J., and K. Varadhan, "Supernetting: an Address Assignment and Aggregation Strategy", RFC 1338, BARRNet, cicso, Merit, OARnet, June 1992.

[Almquist, Gross, 1992] Gross, P., and P. Almquist, "IESG Deliberations on Routing and Addressing", RFC 1380, IESG Chair, IESG Internet AD, November 1992.

[Postel, 1981b] Postel, J., "Transmission Control Protocol - DARPA Internet Program Protocol Specification", STD 7, RFC 793, DARPA, September 1981.

[Postel, 1980] Postel, J., "User Datagram Protocol", STD 6, RFC 768, USC/Information Sciences Institute, August 1980.

[Postel, 1981c] Postel, J., "NCP/TCP Transition Plan", RFC 801, USC/Information Sciences Institute, November 1981.

[Leiner, Rekhter, 1993] Leiner, B., and Y. Rekhter, "The Multi-Protocol Internet" RFC 1560, USRA, IBM, December 1993.

[Ullmann, 1993] Ullmann, R., "TP/IX: The Next Internet", RFC 1475, Process Software Corporation, June 1993.

Bibliography

Gilligan, Nordmark, and Hinden, "The SIPP Interoperability and Transition Mechanism", IPAE, 1993.

Jacobson, V., and R. Braden, "TCP Extensions for Long-Delay Paths", RFC 1072, LBL, USC/Information Sciences Institute, October 1988.

Jacobson, V., Braden, R., and D. Borman, "TCP Extensions for High Performance", RFC 1323, LBL, USC/Information Sciences Institute, Cray Research, May 1992.

Jacobson, V., Braden, R., and L. Zhang, "TCP Extension for High-Speed Paths", RFC 1185, LBL, USC/Information Sciences Institute, PARC, October 1990.

Leiner, B., and Y. Rekhter, "The Multiprotocol Internet", RFC 1560, USRA, IBM, December 1993.

O'Malley, S., and L. Peterson, "TCP Extensions Considered Harmful", RFC 1263, University of Arizona, October 1991.

Westine, A., Smallberg, D., and J. Postel, "Summary of Smallberg Surveys", RFC 847, USC/Information Sciences Institute, February 1983.

Appendix A

The minimum size of an ethernet frame is 64 bytes. With the existing TCP/IP protocol, a minimum size frame is 18 bytes (ethernet header & trailer) + 20 bytes (IP header) + 20 bytes (TCP header) for a total of 58 bytes. The transmitting station must add 6 null pad characters to this frame to make it conform to the 64 byte minimum. This new TCP will increase the size of the TCP header to 48 bytes. Subtracting 26 bytes (the old header and pad characters) we are left with 22 bytes or 176 bits. The time it takes to transmit these additional bits is the impact of this new TCP. The transmission time for several types of media currently used is shown in the table below. You will note that the increased times are all under 20 micro-seconds for anything over T1 speeds. User traffic patterns vary of course but it is generally agreed that 80% of the traffic stays at the local site. If this is true then the increased header size has a negligible impact on communications.

Media	Speed (Mbps)	Rate (nsec/bit)	time (usec)
T1	1.544	647.7	144.00
T3	44.736	22.4	3.91
Enet	10.00	100.0	17.60
FDDI	100.00	10.0	1.76
OC-1	51.84	19.3	3.40
OC-3	155.52	6.4	1.13

Appendix B

In order to support the TA, new DNS entries will need to be created. It is hoped that this function will be accomplished automatically. When a station is installed, the local DNS server is defined. On power up, the station will contact this server and send it it’s TA and domain name. A server process will be listening for this type of information, and it will collect the data, run an authorization check, and install the TA into the DNS server. The following entry will be made.

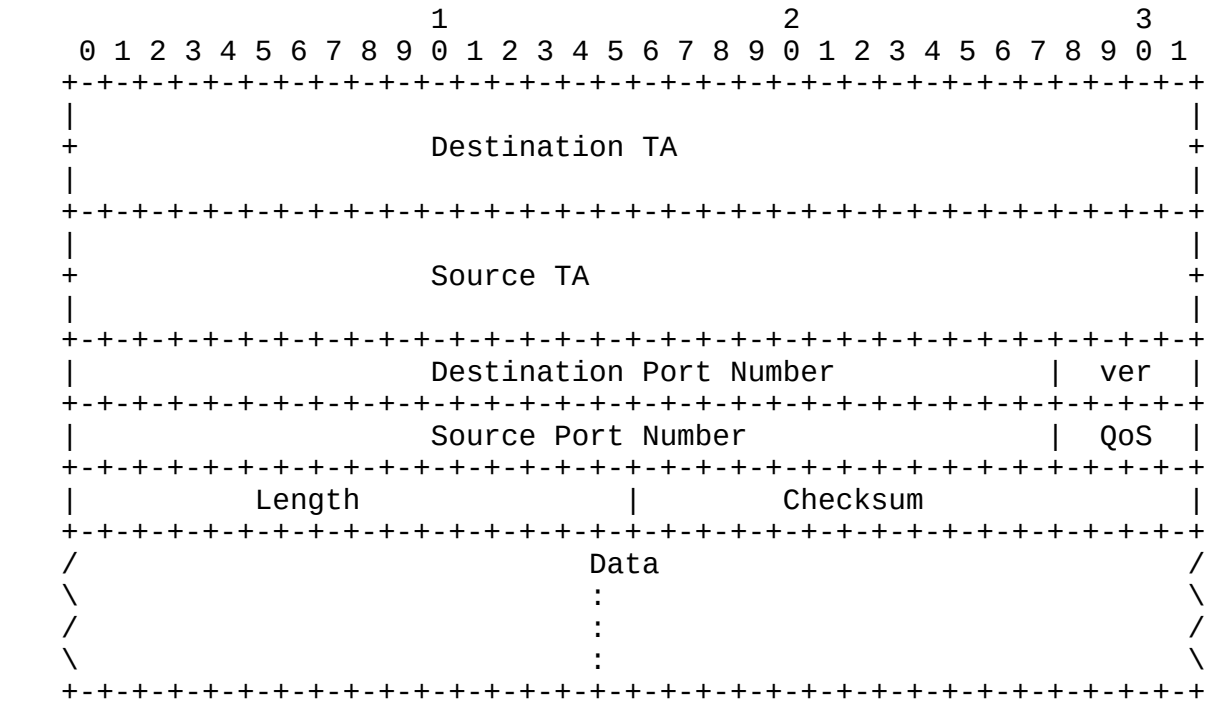
```
node.sub.domain.name    IN     TA   xx.yy.zz.aa.bb.cc.dd.ee

ee.dd.cc.bb.aa.zz.yy.aa.in-addr.tcp IN  PTR node.sub.domain.name.
```

Using these entries, along with the existing DNS A records, a requesting node can determine where the remote node is located. The format xx.yy.zz is the IEEE assigned portion and aa.bb.cc.dd.ee is the encoded machine serial number as described in section 4.1.

Appendix C

Proposed UDP Header



Destination TA: 64 bits.

The Destination Transport Address. The concatenation of the 24 bit IEEE assigned Ethernet address and the 40 bit representation of the machines serial number for the remote node.

Source TA: 64 Bits.

The Source Transport Address. The concatenation of the 24 bit IEEE assigned Ethernet address and the 40 bit representation of the machines serial number for the local node.

Destination Port Number: 28 Bits.

Identifies the specific application on the remote node.

Ver: 4 bits.

This parameter the UDP version number in use within this packet.

Source Port Number: 28 Bits.
Identifies the specific application on the local node.

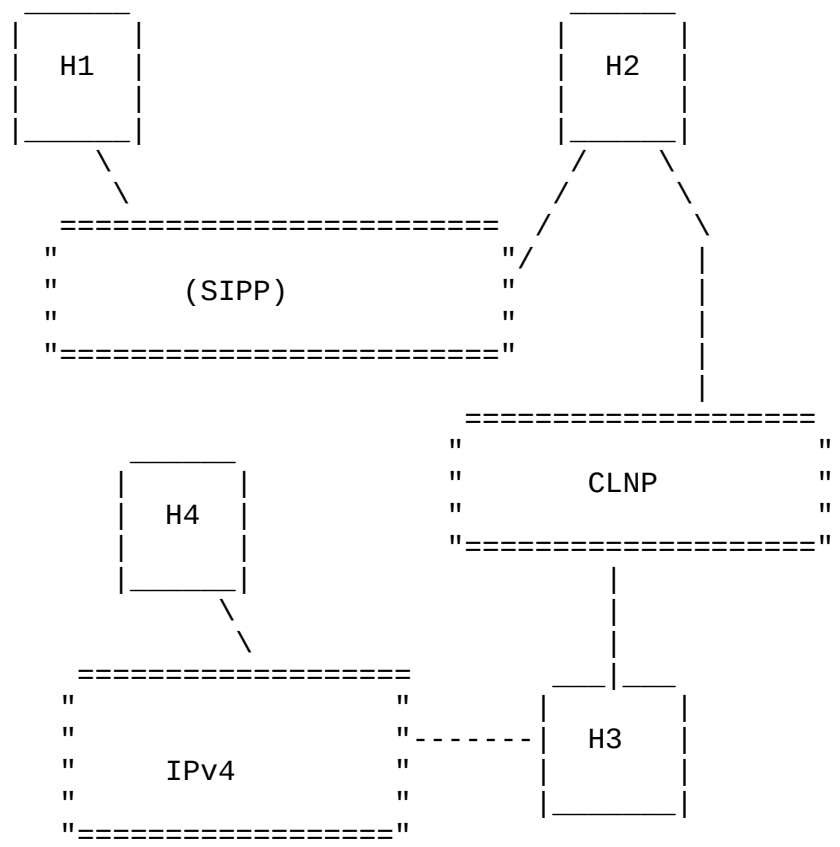
QoS: 4 bits.
The Quality of Service parameter may be set by the user application and passed down to a network layer that supports different levels of service.

Length: 16 bits
The length parameter represents the length of the data area in octets. This value will be set to zero if no data is sent within this packet.

Checksum: 16 bits
The checksum parameter has the same meaning as in the current version of UDP. The current 96 bit pseudo header is NOT used in calculating the checksum. The checksum covers only the information present in this header. The checksum field itself is set to zero for the calculation.

Data: Variable
This is the area in which the data for the datagram will be sent. The length of this data in octets is specified by the length parameter above.

Appendix D



Example 1: H1 Wishes to Establish Communication with H4 (Refer to the figure above.)

1. A user on host H1 attempts to communicate with a user on host H4 by referencing H4 s fully qualified domain name.

2. The TCP on H1 makes a DNS call to determine the TA address of H4.

3. The DNS call returns only the IPv4 address since H4 is determined to be an IPv4 only host.

4. The H1 TCP builds a transmission control block (TCB) setting the C-Bit (compatibility) "ON" since H4 is an IPv4 host. Included in the TCB will also be DA = IP-H4, SA = TA1, DP = 1234, SP = 5000 and any state parameters

describing the connection (port numbers are for example purposes only).

5. The IP on H1 makes a DNS call to determine the network IP address of H4 and correspondingly caches both the TA address from the TCP as well as the network IP address for later use.

6. The packet is now routed using standard SIPP procedures to H2 this is the only path H1 has to H4.

7. H2 receives the packet from H1. The TCP on H2 checks the destination TA of the packet and compares it to its own. In this case it does not match, therefore the packet should be forwarded.

8. H2 s TCP will interrogate the supported network layer(s) and determines the packet must be forwarded to H3.

9. The TCP must now pass the packet the CLNP network layer. The network layer checks its cache to determine if there is a route specified for DA = IP-H4 already in the cache. If so the cache entry is used, if not an entry is created. H2 then routes the packet to H3 via NA3a, which is the network layer address for IP-H4.

10. H3 receives the packet from H2. The TCP on H3 checks the destination TA of the packet and compares it to its own. Once again, it does not match.

11. H3, realizing that the destination address is an IPv4 host, and knowing that it itself is directly connected to the IPv4 network constructs an IPv4 compatible header. H3 also constructs a TCB to manage the IPv4 connection.

12. The packet is sent down to be routed to the IP using standard IP routing procedures.

13. H4 receives the packet at which point the IP on it determines that the destination address is its own and thus proceeds to strip off the IP header and pass the packet up to the TCP layer.

14. The TCP layer than opens the corresponding IPV4_DP port (2311) which forms the first half of the connection to the application.

15. H4 will now reply with a connection accept message, sending the packet back to H3.

16. H3 s TCP receives the packet and based on information in the TCB determines the packet should be delivered to H1. H3 uses the steps outlined above to route the packet back through the network structure.

Example 2: H2 Wishes to Establish Communication with H3 (Refer to the figure above.)

1. A user on host H2 attempts to communicate with a user on host H3 by referencing H3 s fully qualified domain name.

2. The TCP on H2 makes a DNS call to determine the TA address of H3.

3. The DNS call returns the TA address for H3.

4. The H2 TCP builds a transmission control block (TCB) setting the C-Bit (compatibility) "OFF" since H3 is an IPng host. Included in the TCB will also be DA = TA3, SA = TA2, DP = 1111, SP = 2222 and any state parameters describing the connection (port numbers are for example purposes only).

5. The IPng on H2 makes a DNS call to determine the network IPng address of H3 and correspondingly caches both the TA address from the TCP as well as the network IPng address for later use.

6. The packet is now routed to H3 over the IPng supported on that network.

7. H3 receives the packet from H2. The TCP on H3 checks the destination TA of the packet and compares it to its own. In this case it matches.

8. H3 s TCP will construct a TCB and respond with an open accept message.

9. H3 s TCP will interrogate the supported network layer(s) to determine the packet must be delivered to H2 using NA2b which is specified in its cache.

Security Considerations

Security issues are not discussed in this memo.

Authors' Addresses

Richard Carlson
Argonne National Laboratory
Electronics and Computing Technologies
Argonne, IL 60439

Phone: (708) 252-7289
EMail: RACarlson@anl.gov

Domenic Ficarella
Motorola

Phone: (708) 632-4029
EMail: ficarell@cpdmfg.cig.mot.com