Network Working Group
Request for Comments: 1475

R. Ullmann
Process Software Corporation

June 1993

# TP/IX: The Next Internet

## Status of this Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard. Discussion and suggestions for improvement are requested. Please refer to the current edition of the "IAB Official Protocol Standards" for the standardization state and status of this protocol. Distribution of this memo is unlimited.

## Abstract

The first version of this memo, describing a possible next generation of Internet protocols, was written by the present author in the summer and fall of 1989, and circulated informally, including to the IESG, in December 1989. A further informal note on the addressing, called "Toasternet Part II", was circulated on the IETF mail list during March of 1992.

## Table of Contents

## 1. Introduction

This memo presents the specification for version 7 of the Internet Protocol, as well as version 7 of the TCP and the user datagram protocol. Version 7 has been designed to address several major problems that have arisen as version 4 has evolved and been deployed, and to make a major step forward in the datagram switching and forwarding architecture of the Internet.

The major problems are threefold. First, the address space of version 4 is now seen to be too small. While it was viewed as being almost impossibly large when version 4 was designed, two things have occurred to create a problem. The first is a success crisis: the internet protocols have been more widely used and accepted than their designers anticipated. Also, technology has moved forward, putting microprocessors into devices not anticipated except as future dreams a decade ago.

The second major problem is a perceived routing explosion. The present routing architecture of the internet calls for routing each organization's network independently. It is becoming increasingly clear that this does not scale to a universal internet. While it is possible to route several billion networks in a flat, structureless domain, it is not desireable.

There is also the political administrative issue of assigning network numbers to organizations. The version 4 administrative system calls for organizations to request network assignments from a single

authority. While to some extent this has been alleviated by reserving blocks to delegated assignments, the address space is not large enough to do this in the necessary general case, with large blocks allocated to (e.g.) national authority.

The third problem is the increasing bandwidth of the networks and of the applications possible on the network. The TCP, while having proven useful on an unprecedented range of network speeds, is now the limiting factor at the highest speeds, due to restrictions of window size, sequence-space, and port numbers. These limitations can all be addressed by increasing the sizes of the relevant fields. See [RFC1323].

There is also an opportunity to move the technology forward, and take advantage of a combination of the best features of the hop-by-hop connectionless forwarding of version 4 (and CLNP) as well as the pre-established paths of version 5 (and, e.g., the OSI CONS).

Internet Version 7 includes four major areas of improvement, while at the same time retaining interoperation with version 4 with a small amount of conversion knowledge imposed on version 7 hosts and routers.

- It increases the address fields to 64 bits, with sufficient space for visible future expansion of the internet.

- It adds a numbering layer for administrations, above the organization or network layer, as well as providing more space for subnetting within organizations.

- It increases the range of speeds and network path delays over which the TCP will operate satisfactorily, as well as the number of transactions in bounded time that can be served by a host.

- Finally, it provides a forward route identifier in each datagram, to support extremely fast path, circuit, or flow-based forwarding, or any desired combination, while preserving hop-by-hop connectivity.

The result is not just a movement sideways, deploying a new network layer protocol to patch current problems. It is a significant step forward for network layer technology,

## 1.1 Objectives

The following are some of the objectives of the design.

- Use what has been learned from the IP version 4 protocol, fixing things that are troublesome, and not fixing that which is not broken.

- Retain the essential "look and feel" of the Internet protocol suite. It has been very successful, and one doesn't argue with success.

- Not introduce concepts that the Internet has shown do not belong in the protocol definition. Best example: we do not want to add any kind of routing information into the addressing, other than the administrative hierarchy that has sometimes proved useful. Note that the one feature in version 4 addressing (the class system) designed to aid routing is now the most serious single problem.

- Allow current hosts to interoperate, if not universally, at least within an organization or larger area for the indefinite future. There will be version 4 hosts for 10-15 years into the future, the Internet must remain on good terms with them.

- Likewise, we must not impose the new version, telling sites they must convert to stay connected. People resist imposed solutions. It must not be marketed as something different from IPv4; the differences must be down-played at every opportunity.

- The design must allow individual hosts and routers to be upgraded effectively at random, with no transition plan constraints.

- The design must not require renumbering the Internet. The administrative work already accomplished is immense, if it is to be done again it will be in assigning NSAPs.

- It must allow IPv4 hosts to interoperate without any reduction in function, without any modification to their software or configuration. (Universal connectivity will be lost by IPv4 hosts, but they must be able to continue operating within their organization at least.)

- It must permit network layer state-free translation of datagrams between IPv4 and IPv7; this is important to the previous point, and essential to early testing and transitional deployment.

- It must be a competent alternative to CLNP.

- It must not involve changing the semantics of the network layer service in any way that invalidates the huge amount of work that has gone into understanding how TCP (for example) functions in the net, and the implementation of that understanding.

- It must be defined Real Soon; the window of opportunity is almost closed. It will take vendors 3 years to deploy from the time the standard is rock-solid concrete.

I believe all of these are accomplishable in a consistent, well-engineered solution, and all are essential to the survival of the Internet.

### 1.2 Philosophy

Protocols should become simpler as they evolve.

## 2. Internet numbers

The version 4 numbering system has proven to be very flexible, (mostly) expandable, and simple. In short: it works. There are two problems, neither serious when this specification was first developed in 1988 and 1989, but have as expected become more serious:

- The division into network, and then subnet, is insufficient. Almost all sites need a network assignment large enough to subnet. At the top of the hierarchy, there is a need to assign administrative domains.

- As bit-packing is done to accomplish the desired network structure, the 32 bit limit causes more and more aggravation.

### 2.1 Is 64 Bits Enough?

Consider: (thought experiment) 32 bits presently numbers "all" of the computers in the world, and another 32 bits could be used to number all of the bytes of on-line storage on each computer. (Most have a lot less than 4 gigabytes on-line, the ones that have more could be notionally assigned more than one address.)

So: 64 bits is enough to number every byte of online storage in existence today, in a hierarchical structured numbering plan.

Another way of looking at 64 bits: it is more than 2 billion addresses for each person on the planet. Even if I have microprocessors in my shirt buttons I'm not going to have that many. 32 bits, on the other hand, was never going to be sufficient: there are more than 2^32 people.

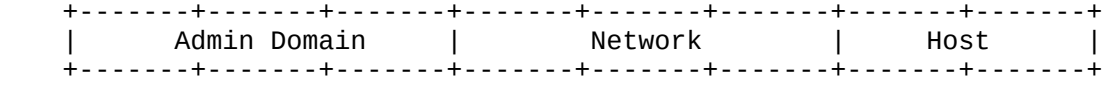
## 2.2 Why version 7?

It was clearly recognized at the start of this project in 1988 that making the address 64 bits implies a new IP header format, which was called either "TP/IX" or "IP version 7"; there wasn't anything magic about the number 7, I made it up. Version 4 is the familiar current version of IP. Version 5 is the experimental ST (Stream) protocol. ST-II, a newer version of ST, uses the same version number, something I was not aware of until recently; I suspected it might have been allocated 6. Besides, I liked 7.

Apparently (as reported by Bob Braden) the IAB followed much the same logic, and may have had the idea planted by the mention of version 7 in the "Toasternet Part II" memo. The IAB in June 1992 floated a proposal that CLNP, or a CLNP-based design, be Internet Version 7. (And promptly got themselves toasted.) However, close inspection of the bits shows that CLNP is clearly version 8.

## 2.3 The version 7 IP address

The Version 7 IP 64 bit address looks like:




Note: the boundary between "network" and "host" is no more fixed than it is today; each (sub)network will have its own mask. Just as the mask today can be anywhere from FF00 0000 (8/24) to FFFF FFFC (30/2), the mask for the 64 bit address can reasonably be FFFF FF00 0000 0000 (24/40) to FFFF FFFF FFFF FFFC (62/2).

The AD (Administrative Domain), identifies an administration which may be a service provider, a national administration, or a large multi-organization (e.g. a government). The idea is that there should not be more than a few hundred of these at first, and eventually thousands or tens of thousands at most. (But note that we do not introduce a hard limit of 2^16 here; this estimate may be off by a few orders of magnitude.) Since only 1/4th of the address space is initially used (first two bits are 01), the remainder can then be allocated in the future with more information available.

Most individual organizations would not be ADs. In the short term, ADs are known to the "core routing"; it pays to keep the number smallish, a few thousand given current routing technology. In the long term, this is not necessary. Big administrations (i.e., with tens of millions of networks) get small blocks where needed, or additional single AD numbers when needed.

While the AD may be used for last resort routing (with a 24/40 mask), it is primarily only an administrative device. Most routing will be done on the entire 48 bit AD+network number, or sub and super-sets of those numbers. (I.e., masks between about 32/32 and 56/8.)

Some ADs (e.g., NSF) may make permanent assignments; others (such as a telephone company defining a network number for each subscriber line) may tie the assignment to such a subscription. But in no case does this require traffic to be routed via the AD.

## 2.4 AD numbers

AD numbers are allocated out of the same numbering space as network numbers. This means that a version 4 address can be distinguished from the first 32 bits of a version 7 address. This is useful to help prevent the inadvertent use of the first half of the longer address by a version 4 host.

There is a non-trivial amount of software that assumes that an "int" is the same size and shape as an IP address, and does things like "ipaddr = *(int *)ptr". This usage has always been incorrect, but does occur with disturbing frequency. As IPv7 8 byte addresses appear in the application layers, this software will find those addresses unreachable; this is preferable to interacting with a random host.

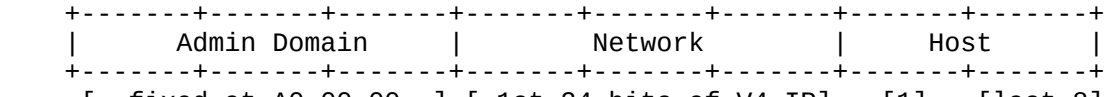
One possible method would be to allocate ADs in the range 96.0.0 to 192.255.255, using the top 1/4 of the version 4 class A space. It is probably best to allocate the first component downwards from 192, so that the boundary between class A and AD can be moved if desired later. This initial allocation provides for 2031616 ADs, many more than there should be even in full deployment.

Eventually, both AD and network will use the full 24 bit space available to them. Knowledge of the AD range should not be coded into software. If it was coded in, that software would break when the entire 24 bit space is used for ADs. (This lesson should have been learned from CIDR.)

## 2.5 Mapping of version 4 numbers

Initially, all existing Internet numbers are defined as belonging to the NSF/Internet AD, number 192.0.0.

The mapping from/to version 4 IP addresses:




So, for example, 192.42.95.15 (V4) becomes 192.0.0.192.42.95.1.15.

And the "standard" loopback interface address becomes 192.0.0.127.0.0.1.1 (I can see explaining that in 2015 to someone born in 1995.)

The present protocol multicast (192.0.0.224.x.y.1.z) and loopback addresses are permanently allocated in the NSF AD.

## 3. IP: Internet datagram protocol

The Internet datagram protocol is revised to expand some fields (most notably the addresses), while removing and relegating to options all fields not universally useful (imperative) in every datagram in every environment.

This results in some simplification, a length less than twice the size of IPv4 even though most fields are doubled in size, and an expanded space for options.

There is also a change in the option philosophy from IPv4: it specified that implementation of options was not optional, what was optional was the existence of options in any given datagram. This is changed in IPv7: no option need be implemented to be fully conformant. However, implementations must understand the option classes; and a future Host Requirements specification for hosts and routers used in the "connected Internet" may require some options in its profile, e.g., Fragment would probably be required.

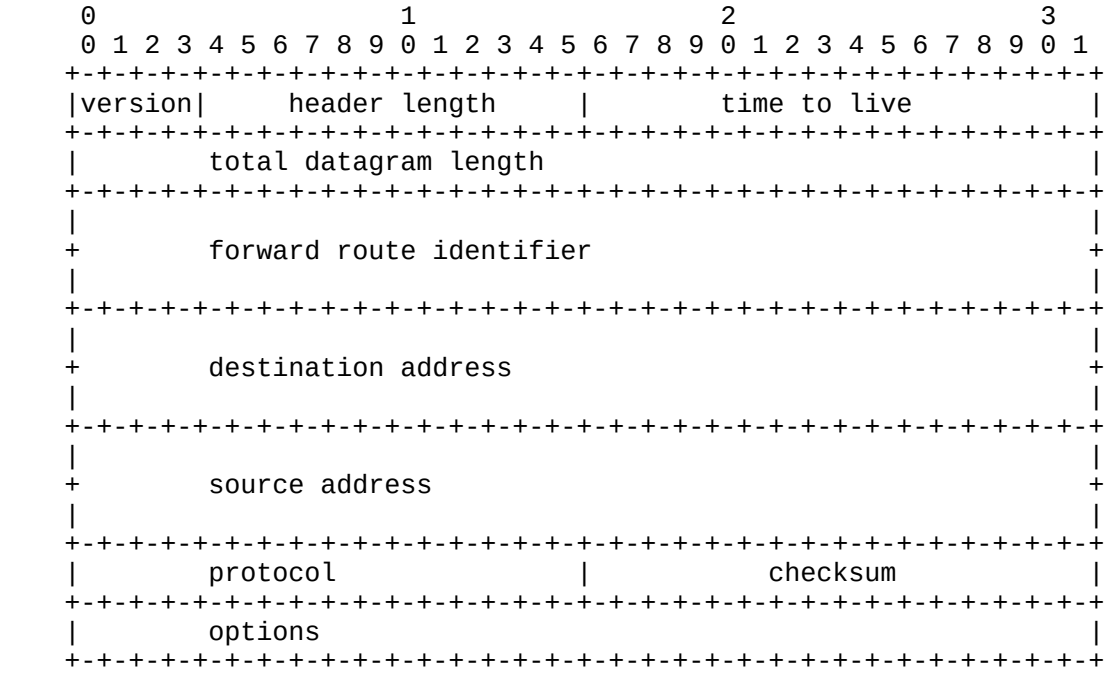
Digression: In IPv4, options are often "considered harmful". It is the opinion of the present author that this is because they are rarely needed, and not designed to be processed rapidly on most architectures. This leads to little or no attempt to improve performance in implementations, while at the same time enormous effort is dedicated to optimization of the no-option case. IPv7 is expected to be different on both counts.

Fields are always aligned on their own size; the 64 bit fields on 64 bit intervals from the start of the datagram.

Options are all 32 bit aligned, and the null option can be used to

push a subsequent option (or the transport layer header) into 64 bit or 64+32 off-phase alignment as desired.

## 3.1 IP datagram header format




A description of each field follows.

### 3.1.1 Version

This document describes version 7 of the protocol.

### 3.1.2 Header length

The header length is a 12 bit count of the number of 32 bit words in the IPv7 header. This allows a header to be (theoretically at least) up to 16380 bytes in length.

### 3.1.3 Time to live

The time to live is a 16 bit count, nominally in 1/16 seconds. Each hop is required to decrement TTL by at least one.

This definition should allow continuation of the useful (even though not entirely valid) interpretation of TTL as a hop count, while we

move to faster networks and routers. (The most familiar use is by "traceroute", which really ought to be directly implemented by one or more ICMP messages.)

The scale factor converts the usual version 4 default TTL into a larger number of hops. This is desireable because the forward route architecture of version 7 enables the construction of simpler, faster switches, and this may cause the network diameter to increase.

#### 3.1.4 Total datagram length

The 32 bit length of the entire datagram in octets. A datagram can therefore be up to 4294967295 bytes in overall length. Particular networks will normally impose lower limits.

#### 3.1.5 Forward route identifier

The identifier from the routing protocol to be used by the next hop router to find its next hop. (A more complete description is given below.)

#### 3.1.6 Destination

The 64 bit IPv7 destination address.

#### 3.1.7 Source

The 64 bit IPv7 source address.

#### 3.1.8 Protocol

The transport layer protocol, e.g., TCP is 6. The present code space for this layer of demultiplexing is about half full. Expanding it to 16 bits, allowing 65535 registered "transport" layers seems prudent.

#### 3.1.9 Checksum

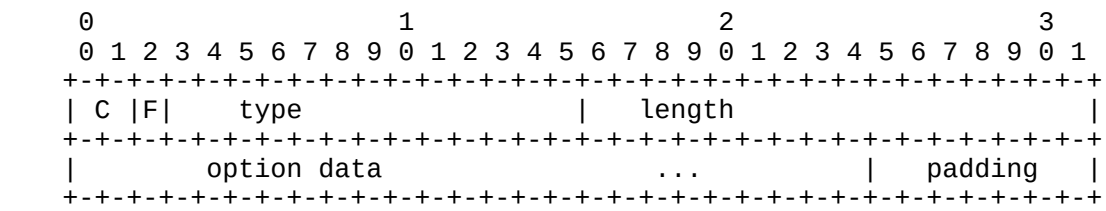
The checksum is a 16 bit checksum of the entire IP header, using the familiar algorithm used in IPv4.

#### 3.1.10 Options

Options may follow. They are variable length, and always 32 bit aligned, as discussed previously.

### 3.2 Option Format

Each option begins with a 32 bit header:




A description of each field:

#### 3.2.1 Class (C)

This field tells implementations what to do with datagrams containing options they do not understand. No implementation is required to implement (i.e., understand) any given option by the TCP/IP specification itself.

Classes:

| | |
|---|---|
| 0 | use or forward and include this option unmodified |
| 1 | use this datagram, but do not forward the datagram |
| 2 | discard, or forward and include this option unmodified |
| 3 | discard this datagram |

A host receiving a datagram addressed to itself will use it if there are no unknown options of class 2 or 3. A router receiving a datagram not addressed to it will forward the datagram if and only if there are no unknown options of class 1 or 3. (The astute reader will note that the bits can also be seen as having individual interpretations, one allowing use even if unknown, one allowing forwarding if unknown.)

Note that classes 0 and 2 are imperative: if the datagram is forwarded, the unknown option must be included.

Class and type are entirely orthogonal, different implementations might use different classes for the same option, except where restricted by the option definition.

Also note that for options that are known (implemented by) the host or router, the class has no meaning; the option definition totally determines the behavior. (Although it should be noted that the option might explicitly define a class dependent behavior.)

### 3.2.2 Copy on fragmentation (F)

If the F bit is set, this option must be copied into all fragments when a datagram is fragmented. If the F bit is reset (zero), the option must only be copied into the first (zero-offset) fragment.

### 3.2.3 Type

The type field identifies the particular option, types being registered as well known values in the internet. A few of the options with their types are described below.

### 3.2.4 Length

Length of the option data, in bytes.

### 3.2.5 Option data

Variable length specified by the length field, plus 0-3 bytes of zeros to pad to a 32 bit boundary. Fields within the option data that are 64 bits long are normally placed on the assumption that the option header is off-phase aligned, the usual case when the option is the only one present, and immediately follows the IP header.

## 3.3 IP options

The following sections describe the options defined to emulate IPv4 features, or necessary in the basic structure of the protocol.

### 3.3.1 Null

The null option, type 0, provides for a space filler in the option area. The data may be of any size, including 0 bytes (perhaps the most useful case.)

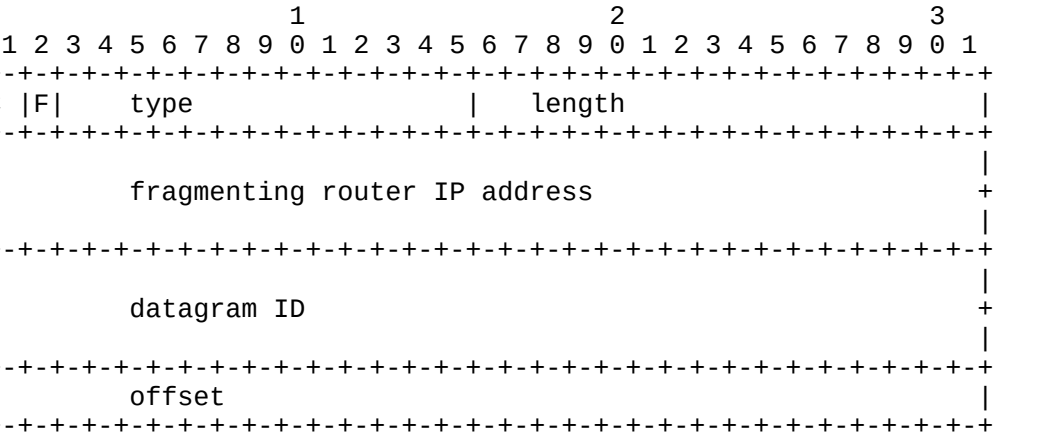
It may be used to change alignment of the following options or to replace an option being deleted, by setting type to 0 and class to 0, leaving the length and content of the data unmodified. (Note that this implies that options must not contain "secret" data, relying on class 3 to prevent the data from leaving the domain of routers that understand the option.)

Null is normally class 0, and need not be implemented to serve its function.

### 3.3.2 Fragment

Fragment (type 1) indicates that the datagram is part of a complete IP datagram. It is always class 2.

The data consists of (one of) the 64 bit IP address(es) of the router doing the fragmentation, a 64 bit datagram ID generated by that router, and a 32 bit fragment offset. The IDs should be generated so as to be very likely unique over a period of time larger than the TCP MSL (maximum segment lifetime). (The TCP ISN (initial sequence number) generator might be used to initialize the ID generator in a router.)




If a datagram must be refragmented, the original 128 bit address+ID is preserved, so that the datagram can be reassembled from any sufficient set of the resulting fragments. The 64 bits fields are positioned so that they are aligned in the usual case of the fragment option following the IP header.

A router implementing Fragment (doing fragmentation) must recognize the Don’t Fragment option.

### 3.3.3 Last Fragment

Last Fragment (type 2) has the same format as Fragment, but implies that this datagram is the last fragment needed to reassemble the original datagram.

Note that an implementation can reasonably add arriving datagrams with Fragment to a cache, and then attempt a reassembly when a datagram with Last Fragment arrives (and the the total length is known); this will work well when datagrams are not reordered in the

network.

### 3.3.4 Don't Fragment

This option (type 3, class 0) indicates that the datagram may not be fragmented. If it can not be forwarded without fragmentation, it is discarded, and the appropriate ICMP message sent. (Unless, of course, the datagram is an ICMP message.) There is no data present.

### 3.3.5 Don't Convert

The Don't Convert option prohibits conversion from IPv7 to IPv4 protocol, requiring instead that the datagram be discarded and an ICMP message sent (conversion failed/don't convert set). It is type 4, usually class 0, and must be implemented by any router implementing conversion. A host is under no such constraint; like any protocol specification, only the "bits on the wire" can be specified, the host receiving the datagram may convert it as part of its procedure. There is no data present in this option.

## 3.4 Forward route identifier

Each IP datagram carries a 64 bit field, called "forward route identifier", that is updated (if the information is available) at each hop. This field's value is derived from the routing protocol (e.g., RAP [RFC1476]). It is used to expedite routing decisions by preserving knowledge where possible between consecutive routers. It can also be used to make datagrams stay within reserved flows and mobile-host tunnels where required.

### 3.4.1 Procedure description

Consider 3 routers, A, B, and C. Traffic is passing through them, between two other hosts (or networks), X and Y, packets are going XABCY and YCBAX. Consider only one direction: routing info flowing from C to A, to provide a route from A to C. The same thing will be happening in the other direction.

An explanation of the notation:

| | |
|---|---|
| R(r,d,i,h) | A route that means: "from router r, to go toward final destination d, replace the forward route identifier in the packet with i, and take next hop h." |
| Ri(r,d) | An opaque (outside of router r) identifier, that can be used by r to find R(r,d,...). |

| | |
|---|---|
| Flowi(r,rt) | An opaque (outside of router r) identifier, that router r can use to find a flow or tunnel with which the datagram is associated, and from that the route rt on which the flow or tunnel is built, as well as the Flowi() for the subsequent hop. |
| Ri(Dgram) | The forward route identifier in a datagram. |

Router C announces a route R(C,Y,0,Y) to router B. It includes in it an identifier Ri(C,Y) internal to C, that will allow C to find the route rapidly. (A table index, or an actual memory address.)

Router B creates a route R(B,Y,Ri(C,Y),C) via router C, it announces it to A, including an identifier Ri(B,Y), internal to B, and used by A as an opaque object.

Router A creates a route R(A,Y,Ri(B,Y),B) via router B. It has no one to announce it to.

Now: X originates a datagram addressed to Y. It has no routing information, and sets Ri(Dgram) to zero. It forwards the datagram to router A (X's default gateway).

A finds no valid Ri(Dgram), and looks up the destination (Y) in its routing tables. It finds R(A,Y,Ri(B,Y),B), sets Ri(Dgram) <- Ri(B,Y), and forwards the datagram to B.

Router B looks at Ri(Dgram) which directly identifies the next hop route R(B,Ri(C,Y),C), sets Ri(Dgram) <- Ri(C,Y) and forwards it to router C.

Router C looks at Ri(Dgram) which directly locates R(C,0,Y), sets Ri(Dgram) <- 0 and forwards to Y.

Y recognizes its own address in Dest(Dgram), ignores Ri(Dgram).

Of course, the routers will validate the Ri's received, particularily if they are memory addresses (e.g., M(a) < Ri < M(b), Ri mod N == 0), and probably check that the route in fact describes the destination of the datagram. If the Ri is invalid, the router must use the ordinary method of finding a route (i.e., what it would have done if Ri was 0), and silently ignore the invalid Ri.

When a route has been aggregated at some router, implicitly or explicitly, it will find that the incoming Ri(Dgram) at most can identify the aggregation, and it must make a decision; the forwarded datagram then contains the Ri for the specific route. (Note this may happen well upstream of the point at which the routes actually

diverge.)

This allows all cooperating routers to make immediate forwarding decisions, without any searching of tables or caches once the datagram has entered the routing domain. If the host participates in the routing, at least to the extent of acquiring the initial Ri required from the first router, then only routers that have done aggregations need make decisions. (If the routing changes with datagrams in flight, some router will be required to make a decision to re-rail each datagram.)

### 3.4.2 Flows

If a "flow" is to be set up, the identifiers are replaced by Flowi(router,route), where each router's structure for the flow contains a pointer to the route on which the flow is built. Datagrams can drop out of the flow at some point, and can be inserted either by the originating host or by a cooperating router near the originator. Since the forward route identifier field is opaque to the sending router, and implicitly meaningful only to the next hop router, use for flows (or similar optimizations) need not be otherwise defined by the protocol. (One presumes that a router issuing both Ri's and Flowi's will take care to make sure that it can distinguish them by some private method.)

If a flow has been set up by a restricted target RAP route announcement, it is no different from a route in the implementation. If this announcement originates from the host itself, the Ri in incoming datagrams can be used to determine whether they followed the flow, or to optimize delivery of the datagrams to the next layer protocol.

### 3.4.3 Mobile hosts

First, a definition: A "mobile host" is a host that can move around, connecting via different networks at different times, while maintaining open TCP connections. It is distinguished from a "portable host", which is simply a host that can appear in various places in the net, without continuity. A portable host can be implemented by assigning a new address for each location (more or less automatically), and arranging to update the domain system. Supporting truly mobile hosts is the more interesting problem.

To implement mobile host support in a general way, either some layer of the protocol suite must provide network-wide routing, or the datagrams must be tunnelled from the "home" network of the host to its present location. In the real network, some combination of these is probable: most of the net will forward datagrams toward the home

network, and then the datagrams will follow a specific host route to the mobile host.

The requirement on the routing system is that it must be able to propagate a host route at least to the home network; any other distribution is useful optimization. When a host route is propagated by RAP as a targeted route, and the routers use the resulting Ri's, the datagram follows an effective tunnel to the mobile host. (Not a real tunnel, in the strict sense; the datagrams are following an actual route at the network protocol layer.)

As explained in RAP [RFC14XX-RAP], a targeted route can be issued when desired; in particular, it can be triggered by the establishment of a TCP connection or by the arrival of datagrams that do not carry an Ri indicating that they have followed a (non-tunnel) route.

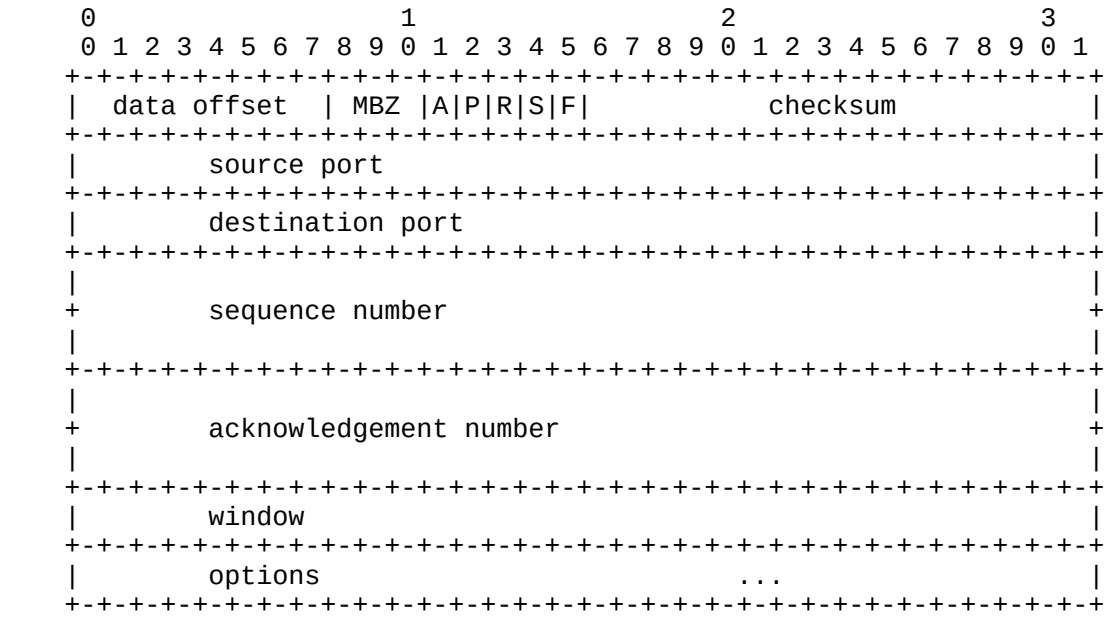
## 4. TCP: Transport protocol

Internet version 7 expands the sizes of the sequence and acknowledgement fields, the window, and the port numbers. This is to remove limitations in version 4 that begin to restrict throughput at (for example) the bandwidth of FDDI and round trip delay of more than 60 milliseconds. At gigabit speeds and delays typical of international links, the version 4 TCP would be a serious limitation. See [RFC1323].

The port numbers are also expanded. This alleviates the problem of going through the entire port number range with a rapid sequence of transactions in less than the lifetime of datagrams in the network.

### 4.1 TCP segment header format

The 64 bit fields (sequence and acknowledgement) in the TCP header are off-phase aligned, in anticipation of the usual case of the TCP header following the 9 32-bit word IP header. If IP options add up to an odd number of 32 bit words, a null option may be added to push the transport header to off-phase alignment.




A description of each field:

### 4.1.1 Data offset

An 8 bit count of the number of 32 bit words in the TCP header, including any options.

### 4.1.2 MBZ

Reserved bits, must be zero, and must be ignored.

### 4.1.3 Flags

These are the protocol state flags, use exactly as in TCPv4, except that there is no urgent data flag.

### 4.1.4 Checksum

This is a 16 bit checksum of the segment. The pseudo-header used in the checksum consists of the destination address, the source address, the protocol field (constant 6 for TCP), and the 32 bit length of the TCP segment.

#### 4.1.5 Source port

The source port number, a 32 bit identifier. See the section on port numbers below.

#### 4.1.6 Destination port.

The 32 bit destination port number.

#### 4.1.7 Sequence

A 64 bit sequence number, the sequence number of the first octet of user data in the segment.

The ISN (Initial Sequence Number) generator used in TCPv4 is used in TCPv7, with the 32 bit value loaded into both the high and low 32 bits of the TCPv7 sequence number. This provides reasonable behavior when the 32 bit rollover option is used (see below) for TCPv4 interoperation. V7 hosts must implement the full 64 bit sequence number rollover.

#### 4.1.8 Acknowledgement

The 64 bit acknowledgement number, acknowledging receipt of octets up to but not including the octet identified. Valid if the A flag is set, if A is reset (0), this field should be zero, and must be ignored.

#### 4.1.9 Window

The 32 bit offered window.

#### 4.1.10 Options

TCP options, some of which are documented below.

### 4.2 Port numbers

Port numbers are divided into several ranges: (all numbers are decimal)

| | |
|---|---|
| 0 | reserved |
| 1-32767 | Internet registered ("well-known") protocols |
| 32768-98303 | reserved, to allow TCPv7-TCPv4 conversion |
| 98304 up | dynamic assignment |

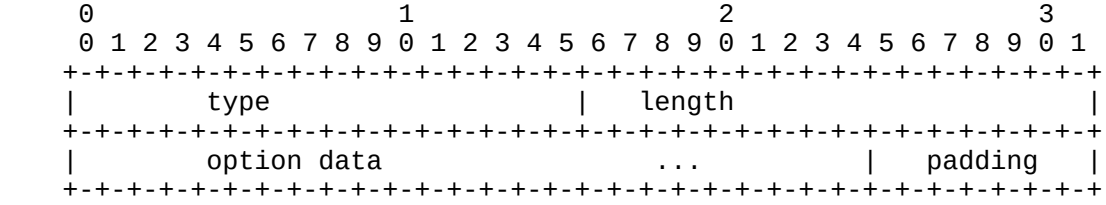
It must also be remembered that hosts are free to dynamically assign for active connections any port not actually in use by that host:

hosts must not reject connections because the "client" port is in the registered range.

## 4.3 TCP options

### 4.3.1 Option Format

Each option begins with a 32 bit header:




### 4.3.2 Null

The null option (type = 0), is to be ignored.

### 4.3.3 Maximum Segment Size

Maximum segment size (type = 1) specifies the largest segment that the other TCP should send, in terms of the number of data octets. When sent on a SYN segment, it is mandatory; if sent on any other segment it is advisory.

Data is one 32 bit word specifying the size in octets.

### 4.3.4 Urgent Pointer

The urgent pointer (type = 2) emulates the urgent field in TCPv4. Its presence is equivalent to the U flag being set. The data is a 64 bit sequence number identifying the last octet of urgent data. (Not an offset, as in v4.)

### 4.3.5 32 Bit rollover

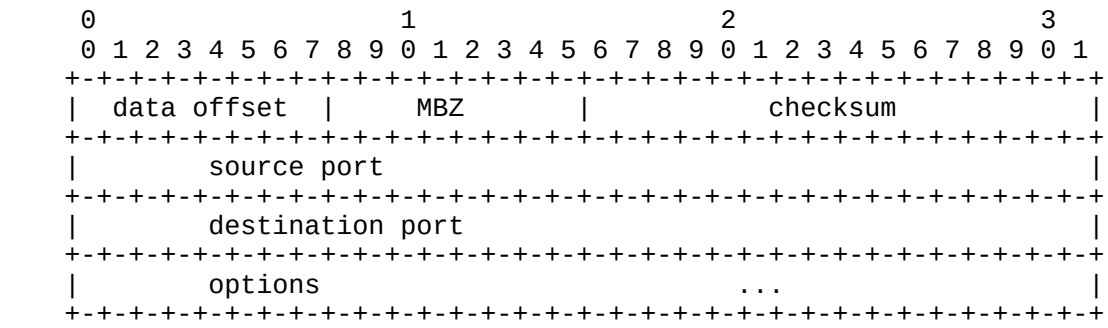
The 32 bit rollover option (type = 3) indicates that only the low order 32 bits of the sequence and acknowledgement packets are significant in the packet.

This is necessary because a converting internet layer gateway has no retained state, and cannot properly set the high order bits. This option must be implemented by version 7 hosts that want to interoperate with version 4 hosts.

## 5. UDP: User Datagram protocol

The user datagram protocol is also expanded to include larger port numbers, for reasons similar to the TCP.

### 5.1 UDP header format




A description of each field:

#### 5.1.1 Data offset

An 8 bit count of the number of 32 bit words in the UDP header, including any options.

#### 5.1.2 MBZ

Reserved bits, must be zero, and must be ignored.

#### 5.1.3 Checksum

This is a 16 bit checksum of the datagram. The pseudo-header used in the checksum consists of the destination address, the source, address, and the protocol field (constant 17 for UDP), and the 32 bit length of the user datagram.

#### 5.1.4 Source port

The source port number, a 32 bit identifier. See the section on TCP port numbers above.

#### 5.1.5 Destination port.

The 32 bit destination port number.

5.1.6 Options

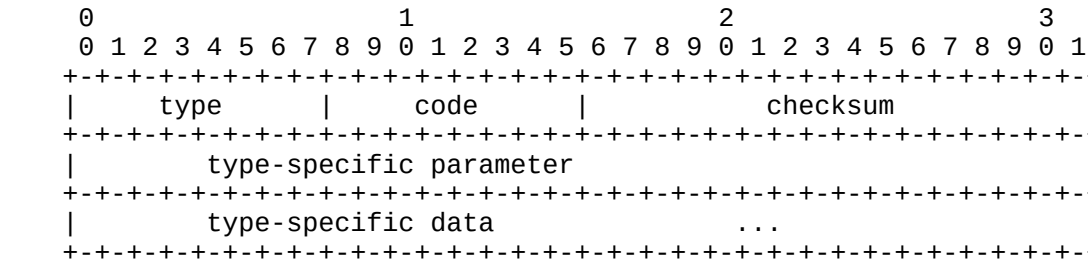
UDP options, none are presently defined.

6. ICMP

The ICMP protocol is very similar to ICMPv4, in some cases not requiring any conversion.

The complication is that IP datagrams are nested within ICMP messages, and must be converted. This is discussed later.

6.1 ICMP header format




Type and code are well-known values, defined in [RFC792]. The codes have meaning only within a particular type, they are not orthogonal.

The next 32 bit word is usually defined for the specific type, sometimes it is unused.

For many types, the data consists of a nested IP datagram, usually truncated, which is a copy of the datagram causing the event being reported. In IPv4, the nested datagram consists of the IP header, and another 64 bits (at least) of the original datagram.

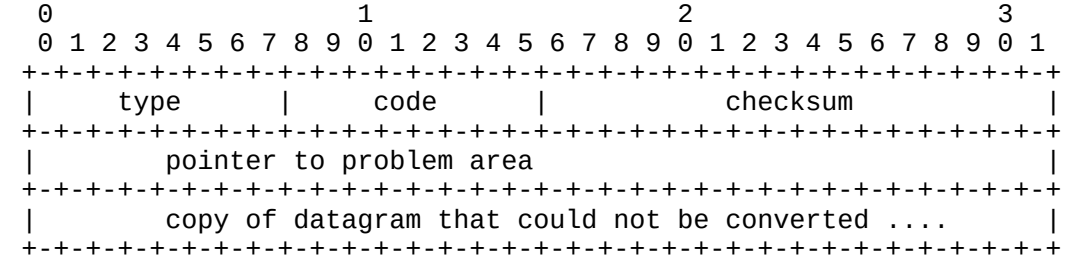
For IPv7, the nested datagram must include the IP header plus 96 bits of the remaining datagram (thus including the port numbers within TCP and UDP), and should include the first 256 bytes of the datagram. I.e., in most cases where the original datagram was not large, it will return the entire datagram.

6.2 Conversion failed ICMP message

The introduction of network layer conversion requires a new message type, to report conversion errors. Note that an invalid datagram should result in the sending of some other ICMP message (e.g., parameter problem) or the silent discarding of the datagram. This message is only sent when a valid datagram cannot be converted.

Note: implementations are not expected to, and should not, check the validity of incoming datagrams just to accomplish this; it simply means that an error detected during conversion that is known to be an actual error in the incoming datagram should be reported as such, not as a conversion failure.

Note that the conversion failed ICMP message may be sent in either the IPv4 or IPv7 domain; it is a valid ICMP message type for IPv4.




The type for Conversion Failed is 31.

The codes are:

| | |
|---|---|
| 0 | Unknown/unspecified error |
| 1 | Don’t Convert option present |
| 2 | Unknown mandatory option present |
| 3 | Known unsupported option present |
| 4 | Unsupported transport protocol |
| 5 | Overall length exceeded |
| 6 | IP header length exceeded |
| 7 | Transport protocol > 255 |
| 8 | Port conversion out of range |
| 9 | Transport header length exceeded |
| 10 | 32 Bit Rollover missing and ACK set |
| 11 | Unknown mandatory transport option present |

The use of code 0 should be avoided, any other condition found by implementors should be assigned a new code requested from IANA. When code 0 is used, it is particularily important that the pointer be set properly.

The pointer is an offset from the start of the original datagram to the beginning of the offending field.

The data is part of the datagram that could not be converted. It must be at least the IP and transport headers, and must include the field pointed to by the previous parameter. For code 4, the transport header is probably not identifiable; the data should

include 256 bytes of the original datagram.

## 7. Notes on the domain system

### 7.1 A records

Address records will be added to the IN (Internet) zone with IPv7 addresses for all hosts as IPv7 is deployed. Eventually the IPv4 addresses will be removed. As mentioned above, the AD (Administrative Domain) space is initially assigned so that the first 4 octets of a v7 address cannot be confused with a v4 address (or, rather, the confusion will be to no effect.)

For example:

```
DELTA.Process.COM.      A       192.42.95.68
                        A       192.0.0.192.42.95.1.68
```

It is important that the A record be used, to avoid the cache consistancy problem that would arise when different records had different remaining TTLs.

Note that if an unmodified version of the more popular public domain nameserver is a secondary for a zone containing IPv7 addresses, it will erroneously issue RRs with only the first four bytes. (I.e., 192.0.0.192 in the example.) This is another reason to ensure that the AD numbers are initially reserved out of the IPv4 network number space. Eventually, zones with IPv7 addresses would be expected to be served only by upgraded servers.

### 7.2 PTR zone

The inverse (PTR) zone is .#, with the IPv7 address (reversed). I.e., just like .IN-ADDR.ARPA, but with .# instead.

This respects the difference in actual authority: the NSF/DDN NIC is the authority for the entire space rooted in .IN-ADDR.ARPA. in the v4 Internet, while in the new Internet it holds the authority only for the AD 0.0.192.#. (Plus, of course, any other ADs assigned to it over time.)

## 8. Conversion between version 4 and version 7

As noted in the description of datagram format, it is possible to provide a mostly-transparent bridge between version 4 and version 7.

This discusses TCP and ICMP at the session/transport layer; UDP is a subset of the TCP conversion. Most protocols at this layer will

probably need no translation; however it will probably be necessary to specify exactly which will have translations done.

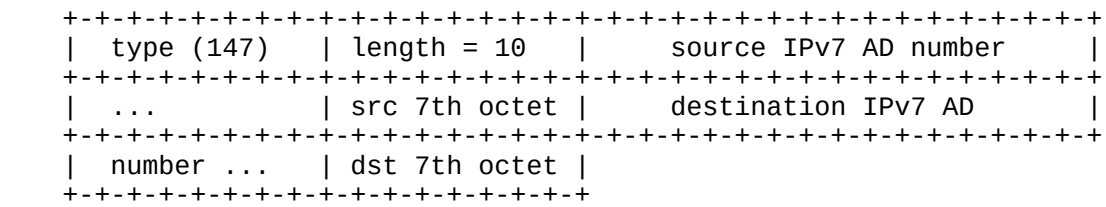
New protocols at the session/transport layer defined over IPv7 should have protocol numbers greater than 255, and will not be translated to IPv4.

Most of the translations should consist of copying various fields, verifying fixed values in the datagram being translated, and setting fixed values in the datagram being produced. In general, the checksum must be verified first, and then a new checksum computed for the generated datagram.

### 8.1 Version 4 IP address extension option

A new option is defined for IP version 4, to carry the extended addresses of IPv7. This will be particularly useful in the initial testing of IPv7, during a time when most of the fabric of the internet is IPv4. An IPv7 host will be able to connect to another IPv7 host anywhere in the internet even though most of the paths and routers are IPv4, and still use the full addressing. This will continue to work until non-unique network numbers are assigned, by which time most of the infrastructure should be IPv7.

#### 8.1.1 Option format




The source and destination are in IPv4 order (source first), for consistancy. The type code is 147.

### 8.2 Fragmented datagrams

Datagrams that have been fragmented must be reassembled by the converting host or router before conversion. Where the conversion is being done by the destination host (i.e., the case of a v7 host receiving v4 datagrams), this is similar to the present fragmentation model.

When it is being done by an intermediate router (acting as an internetwork layer gateway) the router should use all of source, destination, and datagram ID for identification of IPv4 fragments;

note that destination is used implicitly in the usual reassembly at the destination. When reassembling an IPv7 datagram, the 128 bit fragment ID is used as usual.

If the fragments take different paths through the net, and arrive at different conversion points, the datagram is lost.

### 8.3 Where does the conversion happen?

The objective of conversion is to be able to upgrade systems, both hosts and routers, in whatever order desired by their owners. Organizations must be able to upgrade any given system without reconfiguration or modification of any other; and IPv4 hosts must be able to interoperate essentially forever. (IPv4 routers will probably be effectively eliminated at some point, except where they exist in their own remote or isolated corners.)

Each TCP/IP v7 system, whether host or router, must be able to recognize adjacent systems in the topology that are (only) v4, and call the appropriate conversion routine just before sending the datagram.

Digression: I believe v7 hosts will get much better performance by doing everything internally in v7, and using conversion to filter datagrams when necessary. This keeps the usual code path simple, with only a "hook" right after receiving to convert incoming IPv4 datagrams, and just before sending to convert to IPv4. Routers may prefer to keep datagrams in their incoming version, at least until after the routing decision is made, and then doing the conversion only if necessary. In either case, this is an implementation specific decision.

It must be noted that any forwarding system may convert datagrams to IPv7, then back to IPv4, even if that loses information such as unknown options. The reverse is not acceptable: a system that receives an IPv7 datagram should not convert it to IPv4, then back to IPv7 on forwarding.

The preferred method for identifying which hosts require conversion is to ARP first for the IPv7 address, and then again if no response is received, for the IPv4 address. The reservation of ADs out of the v4 network number space is useful again here, protecting hosts that fail to properly use the ARP address length fields.

On networks where ARP is not normally used, the method is to assume that a remote system is v7. If an IPv7 datagram is received from it, the assumption is confirmed. If, after a short time, no IPv7 datagram is received, a v7 ICMP echo is sent. If a reply is received

(in either version) the assumption is confirmed.

If no reply is recieved, the remote system is assumed not to understand IPv7, and datagrams are converted to IPv4 just before transmitting them.

Implementations should also provide for explicit configuration where desired.

## 8.4 Hybrid IPv4 systems

In the course of implementing IPv7, especially in constrained environments such as small terminal servers, it may be useful to implement the IPv4 address extension option directly, thereby regaining universal connectivity.

This may also be a useful interim step for vendors not prepared to do a major rework of an implementation; but it is important not to get stalled in this step.

A hybrid IPv4 + address extension system does not have to implement the conversion, it places this onus on its neighbors. It may itself have an address with the subnet extension (7th byte) not equal to 1.

The implication of hybrid systems is that it is not valid to assume that a host with a IPv7 address is a native IPv7 implementation.

## 8.5 Maximum segment size in TCP

It is probably advisable for IPv4 implementations to reduce the MSS offered by a small amount where possible, to avoid fragmentation when datagrams are converted to IPv7. This arises when IPv4 hosts are communicating through an IPv7 infrastructure, with the same MTU as the local networks of the hosts.

## 8.6 Forwarding and redirects

It may be important for a router to not send ICMP redirects when it finds that it must do a conversion as part of forwarding the datagram. In this case, the hosts involved may not be able to interact directly. The IPv7 host could ignore the redirect, but this results in an unpleasant level of noise as the sequence continually recurs.

## 8.7 Design considerations

The conversion is designed to be fairly efficient in implementation, especially on RISC architectures, assuming they can either do a

conditional move (or store), or do a short forward branch without losing the instruction cache. The other conditional branches in the body of the code are usually not-taken out to the failure/discard case.

Handling options does involve a loop and a dispatch (case) operation. The options in IPv4 are more difficult to handle, not being designed for speed on a 32 bit aligned RISCish architecture, but they do not occur often, except perhaps the address extension option.

For CISC machines, the same considerations will lead to fairly efficient code.

The conversion code must be extremely careful to be robust when presented with invalid input; in particular, it may be presented with truncated transport layer headers when called recursively from the ICMP conversion.

## 8.8 Conversion from IPv4 to IPv7

Individual steps in the conversion; the order is in most cases not significant.

- Verify checksum.
- Verify fragment offset is 0, MF flag is 0.
- Verify version is 4.
- Extend TTL to 16 bits, multiply by 16.
- Set forward route identifier to 0.
- Set first 3 octets of destination to AD (i.e., 192.0.0), copy first three octets from v4 address, set next octet to 1, copy last octet. (This can be done with shift/mask/or operations on most architectures.)
- Do the same translation on source address.
- Copy protocol, set high 8 bits to zero.
- If DF flag set, add Don’t Fragment option.
- If Address Extension option present, copy ADs and subnet extension numbers into destination and source.
- Convert other options where possible. If an unknown option

with copy-on-fragment is found, fail. If copy-on-fragment is not set, ignore the option. I.e., the flag is (ab)used as an indicator of whether the option is mandatory.

- Compute new IP header length.
- Convert session/transport layer (TCP) header and data.
- Compute new overall datagram length.
- Calculate IPv7 checksum.

## 8.9 Conversion from IPv7 to IPv4

The steps to convert IPv7 to IPv4 follow. Note that the converting router or host is partly in the role of destination host; it checks both bits of class in IP options, and (as in the other direction) must reassemble fragmented datagrams.

- Verify checksum.
- Verify version is 7
- Set type-of-service to 0 (there may be an option defined, that will be handled later).
- If length is greater than (about) 65563, fail. (That number is not a typographical error. Note that the IPv7+TCPv7 headers add up to 28 bytes more than the corresponding v4 headers in the usual case.) This check is only to avoid useless work, the precise check is later.
- Generate an ID (using an ISN based sequence generator, possibly also based on destination or source or both).
- Set flags and fragment field to 0.
- Divide TTL by 16, if zero, fail (send ICMP Time Exceeded). If greater that 255, set to 255.
- If next layer protocol is greater than 255, fail. Else copy.
- Copy first 3 octets and 8th octet of destination to destination address.
- Same for source address.
- Generate v4 address extension option. (If enabled; this

probably should be a configuration option, should default to on.)

- Process v7 options. If any unknown options of class not 0 found, fail.
- If Don't Fragment option found, set DF flag.
- If Don't Convert option found, fail.
- Convert other options where possible, or fail.
- Compute new IP header length. This may fail (too large), fail conversion if so.
- Convert session/transport layer (e.g., TCP).
- Compute new overall datagram length. If greater than 65535, fail.
- Compute IPv4 checksum.

### 8.10 Conversion from TCPv4 to TCPv7

- Subtract header words from v4 checksum. (Note that this is actually done with one's complement addition.)
- Copy flags (except for Urgent).
- If source port is less than 32768 (a sign condition test will suffice on most architectures), copy it. If equal or greater, add 65536.
- Same operation on destination port.
- Copy sequence to low 32 bits, set high to 0.
- Copy acknowledgement to low 32 bits, set high to 0.
- Copy window. (The TCPv4 performance extension [RFC1323] window-scale cannot be used, as it would require state; we use the basic window offered.)
- Add 32 bit rollover option.
- Convert maximum segment size option if present.
- Compute data offset and copy data.

- Add header words into saved checksum. It is important not to recompute the checksum over the data; it must remain an end-to-end checksum.

- Return to IP layer conversion.

8.11 Conversion from TCPv7 to TCPv4

- Subtract header from v7 checksum.

- If source port is greater than 65535, subtract 65536. If result is still greater than 65535, fail. (Send ICMP conversion failed/port conversion out of range. The sending host may then reset its port number generator to 98304.)

- Same translation for destination port.

- Copy low 32 bits of sequence number.

- If A bit set, copy low 32 bits of acknowledgement.

- Copy flags.

- If window is greater than 61440, set it to 24576. If less, copy it unchanged. (Rationale for the 24K figure: this has been found to be a good default for IPv4 hosts. If the IPv7 host is offering a very large window, the IPv4 host probably isn't prepared to play at that level.)

- Process options. If 32 Bit Rollover is not present, and A flag is set, fail. (Send ICMP conversion failed/32 bit Rollover missing.)

- If Urgent is present, compute offset. If in segment, set U flag and offset field. If not, ignore.

- Convert Maximum Segment Size option. If greater than 16384, set to 16384.

- Compute new data offset.

- Add header words into v4 checksum.

- Return to IP layer conversion.

### 8.12 ICMP conversion

ICMP messages are converted by copying the type and code into the new packet, and copying the other type-specific fields directly.

If the message contains an encapsulated, and usually truncated, IP datagram, the conversion routine is called recursively to translate it as far as possible. There are some special considerations:

- The encapsulated datagram is less likely to be valid, given that it did generate an error of some kind.
- The conversion should attempt to complete all fields available, even if some would cause failures in the general case. Note, in particular, that in the course of converting a datagram, when a failure occurs, an ICMP message (conversion failed) is sent; this message itself may immediately require conversion. Part of that conversion will involve converting the original datagram.
- Conditions such as overall datagram length too large are not checked.
- The AD and subnet byte assumed in the nested conversion may not be sensible if the IPv4 address extension option is not present and the datagram has strayed from the expected AD. (Not unlikely, given that we know a priori that some error occured.)
- The conversion must be very sure not to make another recursive call if the nested datagram is an ICMP message. (This should not occur, but obviously may.)
- It is probably impossible to generate a correct transport layer checksum in the nested datagram. The conversion may prefer to just zero the checksum field. Likewise, validating the original checksum is pointless.

It may be best in a given implementation to have a separate code path for the nested conversion, that handles these issues out of the optimized usual path.

## 9. Postscript

The present version of TCP/IP has been a success partly by accident, for reasons that weren’t really designed in. Perhaps the most significant is the low level of network integration required to make it work.

We must be careful to retain the successful ingredients, even where we may be unaware of them. Tread lightly, and use all that we have learned, especially about not changing things that work.

This document has described a fairly conservative step forward, with clear extensibility for future developments, but without jumping into the abyss.

## 10. References

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, USC/Information Sciences Institute, August 1980.

[RFC791] Postel, J., "Internet Protocol - DARPA Internet Program Protocol Specification", STD 5, RFC 791, DARPA, September 1981.

[RFC792] Postel, J., "Internet Control Message Protocol - DARPA Internet Program Protocol Specification" STD 5, RFC 792, USC/Information Sciences Institute, September 1981.

[RFC793] Postel, J., "Transmission Control Protocol - DARPA Internet Program Protocol Specification", STD 7, RFC 793, USC/Information Sciences Institute, September 1981.

[RFC801] Postel, J., "NCP/TCP Transition Plan", USC/Information Sciences Institute, November 1981.

[RFC1287] Clark, D., Chapin, L., Cerf, V., Braden, R., and R. Hobby, "Towards the Future Internet Architecture", RFC 1287, MIT, BBN, CNRI, ISI, UCDavis, December 1991.

[RFC1323] Jacobson, V., Braden, R, and D. Borman, "TCP Extensions for High Performance", RFC 1323, LBL, USC/Information Sciences Institute, Cray Research, May 1992.

[RFC1335] Wang, Z., and J. Crowcroft, A Two-Tier Address Structure for the Internet: A Solution to the Problem of Address Space Exhaustion", RFC 1335, University College London, May 1992.

[RFC1338] Fuller, V., Li, T., Yu, J., and K. Varadhan, "Supernetting: an Address Assignment and Aggregation Strategy", RFC 1338, BARRNet, cicso, Merit, OARnet, June 1992.

[RFC1347] Callon, R., "TCP and UDP with Bigger Addresses (TUBA), A Simple Proposal for Internet Addressing and Routing", RFC 1347, DEC, June 1992.

[RFC1476] Ullmann, R., "RAP: Internet Route Access Protocol", RFC 1476, Process Software Corporation, June 1993.

[RFC1379] Braden, R., "Extending TCP for Transactions -- Concepts", RFC 1379, USC/Information Sciences Institute, November 1992.

## 11. Security Considerations

Security issues are not discussed in this memo.

## 12. Author's Address

Robert Ullmann
Process Software Corporation
959 Concord Street
Framingham, MA 01701
USA

Phone: +1 508 879 6994 x226
Email: Ariel@Process.COM