                         Pip Header Processing

Status of this Memo

Preamble

   During 1992 and 1993, the Pip internet protocol, developed at
   Bellcore, was one of the candidate replacments for IP. In mid 1993,
   Pip was merged with another candidate, the Simple Internet Protocol
   (SIP), creating SIPP (SIP Plus). While the major aspects of Pip--
   particularly its distinction of identifier from address, and its use
   of the source route mechanism to achieve rich routing capabilities--
   were preserved, many of the ideas in Pip were not. The purpose of
   this RFC and the companion RFC "Pip Near-term Architecture" are to
   record the ideas (good and bad) of Pip.

   The remainder of this document is taken verbatem from the Pip draft
   memo of the same title that existed when the Pip project ended. As
   such, any text that indicates that Pip is an intended replacement for
   IP should be ignored.

Abstract

   Pip is an internet protocol intended as the replacement for IP
   version 4. Pip is a general purpose internet protocol, designed to
   handle all forseeable internet protocol requirements. This
   specification defines the Pip header processing for Routers and
   Hosts.

Acknowledgements

Conventions

    All functions in this specification are mandatory.

1. Introduction

    Pip is an internet protocol intended as the replacement for IP
    version 4. Pip is a general purpose internet protocol, designed to
    handle all forseeable internet protocol requirements. This
    specification defines the Pip header processing for Routers and
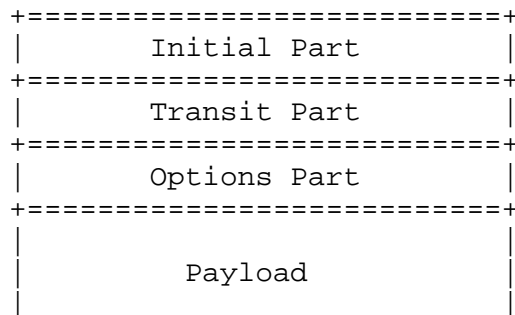   Hosts.

    The design of Pip is fundamentally different from that of previous
    internetwork protocols. Pip is designed to be as general as
    possible, but without significantly compromising performance.
    Because of Pip's generality, it can handle forseeable routing and
    addressing requirements. It is hoped that it will be able to handle
    most if not all future routing and addressing requirements.

    There are many detailed aspects of Pip that provide this generality
    that are not discussed here. It is useful, however, to mention one
    general aspect. That is, Pip strives to remove as much "functional
    semantics" from the base specification as possible. Pip defines a
    packet header and forwarding rules that can include many different
    functional semantics (that is, routing, addressing, and flow
    paradigms). Therefore, the reader may often find him or herself
    asking "But how do you do foo with Pip?" The answer to this sort of
    question belongs in companion documents to the basic Pip spec.

    Pip can be thought of as a mechanism for triggering actions in hosts
    and routers, just as a machine language can be thought of as a
    mechanism for triggering actions in CPUs. The machine language has
    no functional semantics outside of the specific actions it triggers
    (move this register, write that memory location, etc.). But, the
    machine language is a very powerful tool upon which functional
    semantics are built. Likewise, Pip is a powerful tool upon which
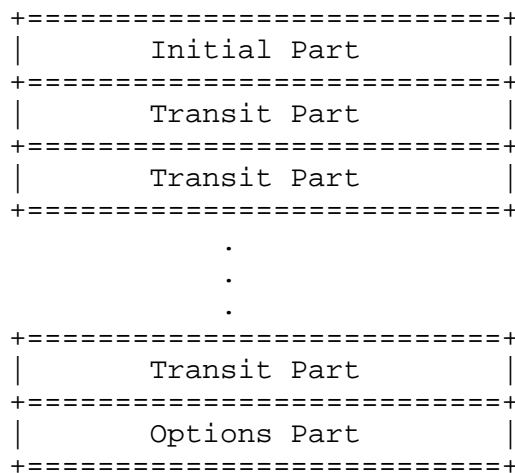   routing, addressing, and flow functions can be built.

2. Pip Specification

   The Pip header is partitioned into three parts, the Initial Part, the
   Transit Part, and the Options Part.

                +===========================+
                | Initial Part |
                +===========================+
                | Transit Part |
                +===========================+
                | Options Part |
                +===========================+
                | |
                | |
                | Payload |
                | |
                | |

   Each part falls on a 32-bit boundary (as indicated by the double
   lines shown), and the Transit Part falls on a 64 bit boundary.

   The concept of tunneling in an integral part of Pip. Pip achieves
   tunneling by encapsulating the Transit Part of the Pip header in
   another Transit Part. Therefore, when tunneling, there is one
   Transit Part for each (nested) tunnel:

                +===========================+
                | Initial Part |
                +===========================+
                | Transit Part |
                +===========================+
                | Transit Part |
                +===========================+
                            .
                            .
                            .
                +===========================+
                | Transit Part |
                +===========================+
                | Options Part |
                +===========================+

   Because each Transit Part has only what is necessary for router
   forwarding and handling, this method of tunneling is reasonably
   efficient in terms of packet size.

2.1 Initial Part

   The Initial Part is formatted as shown in Figure 1.

                                          length, in bits
          +===========================+
          | Version Number = 8 | 4
          +---------------------------+
          | Sub-Version | 4
          +---------------------------+
          | Options Offset | 8
          +---------------------------+
          | Options Contents | 8
          +---------------------------+
          | Options Present | 8
          +===========================+
          | Packet SubID | 16
          +---------------------------+
          | Protocol | 16
          +===========================+
          | Dest ID | 64
          +===========================+
          | Source ID | 64
          +===========================+
          | Payload Length | 32
          +===========================+
          | Host Version | 8
          +---------------------------+
          | Payload Offset | 8
          +---------------------------+
          | Hop Count | 16
          +===========================+

                   Figure 1: Initial Part

   An explanation of each field follows.

   2.1.1 Version Numbers

   The first octet is divided into two 4-bit fields, the Version and the
   Sub-Version. The Version field is set to be 8, and is meant to be
   version 8 of IP. (As of this writing, this is an experimental number
   assigned for development of Pip.) Thus, all encapsulation schemes
   defined for IP can work for Pip as well.

   As long as the Version field is 8, the Initial Part and Options Part
   of the Pip Header is as specified in this standard. (In other words,
   the Sub-Version field refers only to the Transit Part.)

By doing this, we allow the Transit Part of the Pip Header to change
completely without necessarily requiring a host to understand the new
Transit Part. If a host receives a Pip header with a Version number
of 8 and an unknown Sub-version number, the host does not try to
parse the Transit Part at all, rather it processes only the Initial
Part and the Options Part. (By using the Pip Header Protocol to
format Pip Headers, a host can be made to formulate the right Transit
Part, even though the host doesn't understand the semantics of the
Transit Part. This allows radical migration of the Transit Part
while potentially not requiring changes to hosts.)

If a host or a router receives a packet with an unknown Version
number, the packet is silently discarded.

The Sub-Version field is set to the value 0 for the version of Pip
defined in this specification. As long as the Sub-Version number is
0, the Transit Part is as specified in this standard. Any packet
received by a router with a Version number of 8 and an unknown Sub-
Version number is silently discarded.

2.1.2 Options Offset

The Options Offset indicates the position of the Options Part. The
unit of measure of the Options Offset is 32-bit words, counting the
first word of the Pip Header as word 0.

2.1.3 Options Contents

This field indicates how the Options Present field should be
interpreted. Each bit of the Options field indicates if each of up
to eight options is present in the Options Part. The Options
Contents field indirectly indicates which option each bit of the
Options Present field refers to. We say indirectly because the
mapping referred to by the Options Contents field is stored locally.
In other words, without additional information (the mapping), it is
not possible to examine the Options Contents field and know what
option each bit of the Options Present field refers to.

Any of 256 possible Options Contents values can be active at a given
time. (Note that the means by which the meaning of the Options
Contents values are assigned and conveyed to routers and hosts is
outside the scope of this specification.)

2.1.4 Options Present

This field indicates which of the Options indicated by the Options
Contents field are actually present in the Options Part. Each bit of
this field refers to a single option type. The mapping of each bit
to its' option type is determined by the Options Contents field.

For instance, assume that the Options Contents field indicates that
bit 0 of the Options Present field refers to the PDN Address option,
that bit 1 of the Options Present field refers to the foo option, and
that bit 2 of the Options Present field refers to the Fragmentation
option. (As of this writing, there is only one option. Until there
are more than eight options, there is no need to define more than one
Options Contents values.)

In this case, a value of 101 in the Options Present field indicates
that the PDN Address and Fragmentation options are present in the
Options Part, and that the foo option is not present.

Note that an Options Present value of 0 indicates that there are no
options present, regardless of the value of the Options Contents
field. Note also that no more than 8 options, not including the
default first option (the Options Descriptor), can be present in any
Options Part.

The Options Contents/Options Present method of processing options
allows for efficient processing of options. First, a router can
ignore any options that may be present but that do not impact it (for
instance, a router not attached to a PDN need not consider the PDN
Address option). Second, the desired option can be very quickly
retrieved, because the first option, the Options Descriptor option,
contains the offset of each of the up to eight options indicated by
the Options Present field.

2.1.5 Packet SubID

This field is used by Pip hosts to correctly associate received PCMP
messages with local control blocks. This is necessary because the
semantics of the Transit Part can change while a packet is in
transit. Therefore, a router sending a PCMP message cannot
necessarily provide all of the information needed by the Pip host to
correctly identify the context of the received message (that is,
which "packet flow" it is identified with).

A PCMP message uses the Protocol, Source ID, Dest ID, and Packet
SubID to define the PCMP messages context. It is not sufficient to
use just Protocol, Source ID, and Dest ID, because two hosts running
the same protocol between them may have multiple "flows", for

instance, a data flow, a video flow, and an audio flow in the case of
multi-media. Each flow may have a different Transit Part, and take
different paths. Therefore, the Packet SubID field is needed to
further differentiate.

2.1.6 Protocol

Indicates the protocol header found in the payload. The values for
this field are the same as those used for IPv4.

2.1.7 Dest ID

The Dest ID field indicates the Pip ID of the final recipient of the
Pip packet. This field is examined by both hosts and routers.

When a Pip System processes the Routing Directive (RD), it may
determine that it needs to examine the Dest ID for further
processing. This may happen both when a host or router receives a
Pip packet destined for itself, or when a router receives a packet
that should be forwarded based on Dest ID (as indicated by the RD).

When a Pip system determines at forwarding time that a packet is
destined for itself, it checks the Dest ID to verify if that packet
is destined for it. If the complete Dest ID matches one of its own
Pip IDs, then the packet is for it, and is passed to the layer
indicated by the Protocol field (in the Host Part). (The Pip system
may of course wish to check a security option before passing a packet
to an upper layer.)

If the complete Dest ID field does not match one of its own IDs, then
an ID/RD Mismatch PCMP message is sent to the source of the packet,
as indicated by the Source ID and potentially source information in
the RD. The purpose of this message is to flush the ID to RD binding
in the source Pip host.

2.1.8 Source ID

This is the Pip ID of the source of the packet. It is passed to
upper layers for the purposes of identifying the context for the
packet.

2.1.9 Payload Length

The Payload Length gives the length of the Pip packet payload in
units of 8 bits. The Payload Length does not include the length of
the Pip header.

2.1.10 Host Version

The Host Version field indicates what "version" of Pip software the
sending host has implemented. This is to allow a host to inform a
router which ancillary protocols/messages the host is able to accept.
It is envisioned that over time, new host functions will be
developed. Different hosts will install these new functions at
different times. This field allows routers to know what functions
the host can and cannot handle.

2.1.11 Payload Offset

The Payload Offset indicates the position of the Payload Part. The
unit of measure of the Payload Offset is 32-bit words, counting the
first word of the Pip Header as word 0.

If a Pip system encapsulates a Transit Part in another Transit Part,
then the Payload Offset is increased by the length of the new Transit
Part.

2.1.12 Hop Count

The Hop Count is decremented by every router that forwards the Pip
packet. If a system receives a Pip header with a Hop Count equal to
0, and is not the recipient of the packet, then the packet is
discarded and a PCMP Destination Unreachable is routed to the system
indicated by the Routing Directive. (In other words, a host can
legally receive a Transit Part with a Hop Count of 0, and indeed a
host doesn't look at the Hop Count field upon reception.)

2.2 Transit Part

   The Transit Part is formatted as shown in Figure 2.

```
                                          length, in bits
                      +==========================+
                      |         Reserved         |      16
                      +--------------------------+
                      |    Transit Part Offset   |       8
                      +--------------------------+
                      |        HD Contents       |       8
                      +==========================+
                      |   Handling Directive (HD)|      32
         -------------+==========================+
             ^        |        FTIF Offset       |       8
             |        +--------------------------+
             |        |        RC Contents        |       8
             |        +--------------------------+
             |        |   Routing Context (RC)   |      16
          Routing     +==========================+
                      |          FTIF 1          |      16
          Directive   +--------------------------+
             |        |          FTIF 2          |      16
             |        +--------------------------+
             |                     .
             |                     .
             |                     .
             |        +--------------------------+
             |        |          FTIF N          |      16
             |        +--------------------------+
             v        |         Padding          |   Variable
         -------------+==========================+
```
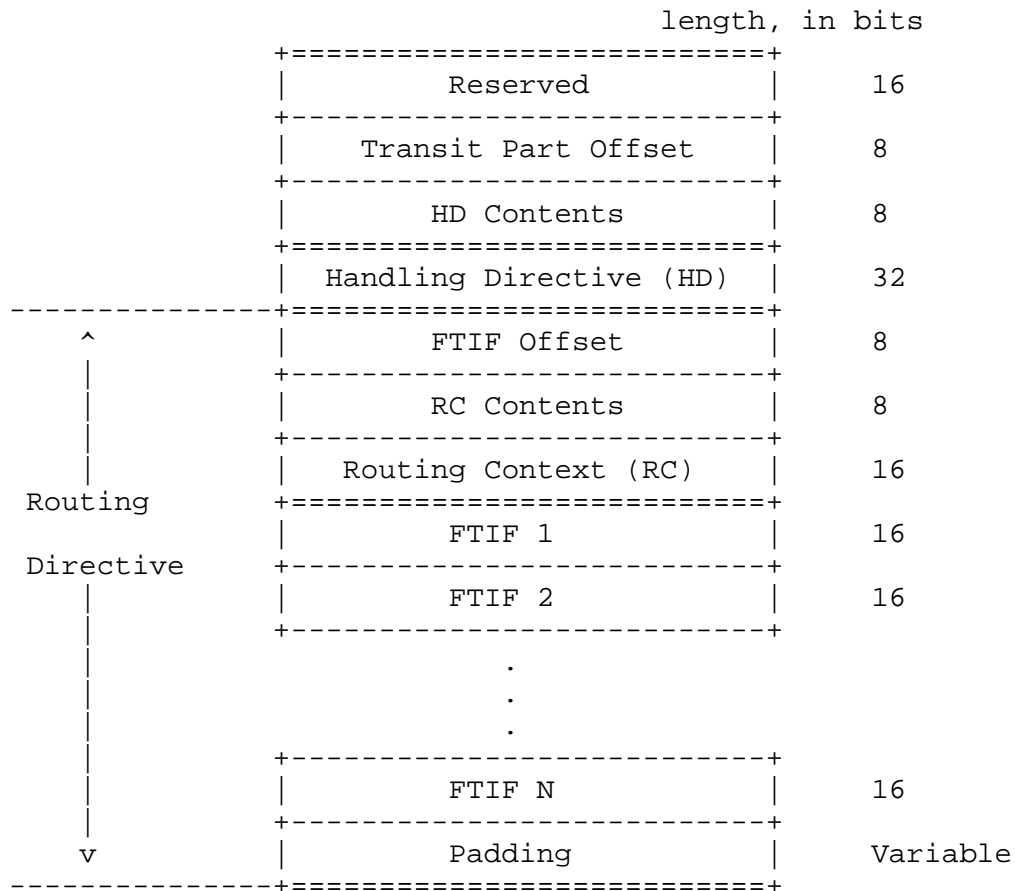
                         Figure 2: Transit Part

   An explanation of each field follows.

2.2.1 Transit Part Offset

   This field gives the position of the first word of the next Transit
   Part. The unit of measure of the Transit Part Offset is 32-bit
   words, counting the first word of the current Transit Part as word 0.
   If there is no next Transit Part, then this field is written as all
   0's.

2.2.2 HD Contents

The HD Contents field indicates how the Handling Directive (HD) field
should be interpreted. The HD field is divided into multiple fields,
each representing a different handling function. Each individual
field in the HD is called an HD Unit (HDU). The Options Contents
field indirectly indicates which HDUs are in the HD field, and where
they are. We say indirectly because the mapping referred to by the
HD Contents field is stored locally. In other words, without
additional information (the mapping), it is not possible to examine
the HD Contents field and know what the HDU locations are.

Any of 256 possible HD Contents values can be active at a given time.
(Note that the means by which the meaning of the HD Contents values
are assigned and conveyed to routers and hosts is outside the scope
of this specification.)

2.2.3 Handling Directive (HD)

The HD is a general purpose field used for the purpose of triggering
special packet handling by a Pip system. The HD field does not
influence a Pip router's next hop choice for a Pip packet, nor does
it influence a Pip host's determination as to whether the Pip packet
is destined for it. Examples of special packet handling would be
"low priority queueing", or "high priority discard", etc. (Note that
the Transit Options also influence "handling", in the sense that
handling is essentially defined here to mean "anything that is not
routing. The HD field, though, is intended for the most common types
of handling--handling that is expected to be in a significant
percentage of packets.)

Both hosts and routers use the HD field. (Hosts may make use of the
HD field for packet handling for both incoming and outgoing packets.)

There is a complete distinction between the syntax and the semantics
of the HD field. (This can be contrasted with, for instance, IP,
which couples the semantics and syntax of the TOS bits. That is, the
IP specification itself determines, to a first degree, how the TOS
bits are interpreted.) Each Pip system can modify the semantic
meaning of the HD, for instance, by increasing or decreasing the
queueing priority of a packet. This is called packet tagging.

From an abstract modeling perspective, the HD is handled as follows:

1. Extract the semantic meaning(s) (the handling instructions
    associated with the HDUs) from the HD field. Transmitting Pip
    hosts determine the semantic meaning by some other means, such as
    the upper layer protocol. If the receiving system decapsulates

     multiple Pip headers, then the HD semantics are extracted from the
     lowest Pip header for which it is not the target (see example on
     tunneling below).

2.   Handle the Pip packet according to those instructions. In some
     cases, it is possible that the Pip system does not understand the
     semantics of one or more HDUs of the HD field. For each HDU whose
     semantics are not understood, however, the pip system at least
     knows whether to 1) pass the HDU on untouched, 2) set it to all
     0s, 3) set it to all 1s, 4) discard the packet silently, or 5)
     discard the packet with a PCMP HDU Not Understood packet.

3.   Modify the semantic meaning if necessary. Note also that if the
     Pip packet is replicated for multicast, each packet has its HD
     semantics modified individually. .LP .in 3 2.2.4 Tunneling .LP
     Consider two Pip systems, X and Y, separated by one or
     intermediate Pip systems. X wishes to tunnel a Transit Part to Y.
     Y is therefore the target system of the tunnel. A Transit Part He
     arrives at X. In order to forward the Transit Part to Y, X
     encapsulates He in another Transit Part, Hy. Y is the target
     system for Transit Part Hy. X sets the HD of He to what it would
     have been if Y was directly connected to X (that is, there were no
     intermediate Pip systems between X and Y). Further, it is
     intended that Y will derive its HD semantics from the HD of
     Transit Part He, not Transit Part Hy. .sp .KS

        ----0-----o-----o-----o-----o-----0----
            X   I   J   K   L   Y

Now consider the operation of Pip system L (the previous hop system
to Y). When L forwards the packet to Y, it may either decapsulate
the packet (in the knowledge that Y is the target for Hy), or not
decapsulate the packet. Either way, L derives its HD semantics from
the HD of Transit Part Hy.

If L does not decapsulate the Transit Part, then it is as though I,
J, K, and L are a "subnetwork" (albeit a Pip subnetwork), and Y is
stripping the "subnetwork" header (Hy) off before processing the true
Transit Part (He). If L does decapsulate the Transit Part, then,
from Y's perspective, it is essentially as though Y were directly
connected to X.

2.2.5 Routing Directive (RD)

The RD consists of the Routing Context (RC), the RC Contents, the
FTIF Offset, and a series of zero or more FTIFs (Forwarding Table
Index Fields). This series of FTIFs is called the FTIF Chain. The
sole purpose of the RD is to determine how to forward the Pip
packet--the RD does not influence handling in any way.

Figure 3 illustrates the decision process for forwarding the Pip
packet.

```
                   +---------+(next level RC)
     (decapsulate)|         |
                  |            v
                  |<--------RC---------------->FIB
                  |      /               |      |    IF Offset)
                  |     |      |
                  |     |      v
                  |<------|---FTIF------------>FIB
                  |     | /    :
                  |     |<-    :(repeatedly...)
                  |     |      :
                  |     |      v
                  |<------|---FTIF------------>FIB
                  |    / |    |
                  |<-    |
                      v      v
                    DestID------------->FIB
```
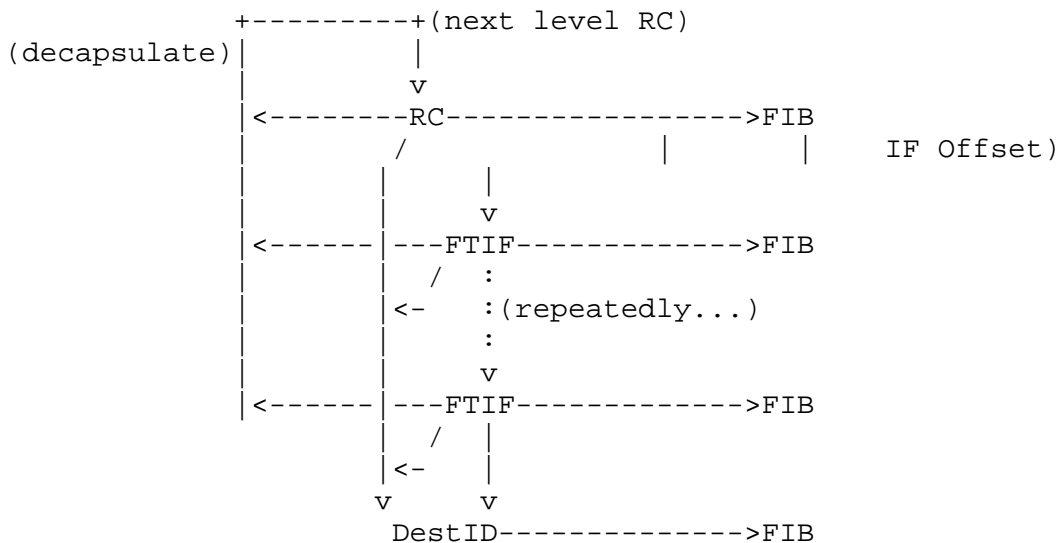
                  Figure 3: Forwarding Process

Figure 3 is interpreted as follows. The FIB is the Forwarding
Information Block. The FIB contains all the information needed to
forward a packet, and may contain multiple next hop (for multicast).
This information includes 1) the outgoing interface, 2) how to
encapsulate the packet, including lower-layer address(es) (the
lower-layer address(es) along with the outgoing interface determine
the next hop Pip system), 3) whether and how to tunnel, 4) how to
modify the semantics of the HD and RC, and how to modify the FTIF
Offset. The goal of the forwarding algorithm is to reach the
appropriate FIB.

The directed lines in Figure 3 start at the RC and, through various
possible paths, reach a FIB. These lines represent the various
information that can influence the forwarding decision (that is, the
FIB chosen). For instance, there is no way to reach a FIB without

first examining the information in the RC. However, it is possible
to identify a FIB by considering only the information in the RC (as
indicated by the directed line leading directly right from the RC).
Based on the information in the RC, it is also possible to determine
that the Transit Part must be decapsulated, and 1) the RC of the next
Transit Part be processed (the line leading directly left), 2) the
FTIF indicated by the FTIF Offset is processed (the line leading down
and right), or 3) the Dest ID is processed (the line leading down and
lest).

Likewise, when considering the value of an FTIF (in addition to all
information already considered), the resulting action may be that 1)
a FIB is identified, 2) the Transit Part is decapsulated, 3) the
subsequent FTIF is processed, or 4) the Dest ID is processed.

The RC is handled similarly to the HD. The RC Contents field
indicates how the RC should be interpreted. While the RC is
constructed similarly to the HD in the sense that it consists of
multiple fields, the RC can be interpreted as a flat field in-so-far
as forwarding a Pip packet is concerned, whereas the HD cannot.

Thus, in a mechanical sense, the RC Contents can be viewed as an
index into a table that returns a pointer to another table (an
rcTable), which is indexed by the RC itself. (Or, the combined RC
Contents/RC can be viewed as a single large index into a single
table, etc.)

The FTIF Offset field indicates which FTIF is active. The active
FTIF is the one that is used to index the forwarding table indicated
by the RC Contents/RC. An FTIF Offset value of 0 means that the
first FTIF is active, an FTIF Offset value of 1 means that the second
FTIF is active, and so on. If there are no FTIFs, then the FTIF
Offset has no meaning, and can be any value. In this case, the RC
field itself will indicate how to forward the packet.

The FTIF Chain is padded out to a 32-bit boundary. Note that there
can be more than 16 bits of padding (for instance, if it is desirable
to pad out to a 64-bit boundary). The padding is ignored upon
receipt, and can be transmitted as any value (that is, it does not
have to be any specific pattern of 0's or 1's).

Note that a single "number" in the FTIF chain may in fact be more
than 16 bits in length. In this case, the number can be encoded as
multiple FTIFs with no loss of generality. It is only required that
in all cases a multiple FTIF number be distinguishable from a single
FTIF number.

2.2.6 Router RD Forwarding Algorithm

This section describes the forwarding algorithm for a Pip router.

1. Using the value of the RC field as an index, retrieve one of the
    following instructions (steps 2 - 5) from the rcTable determined
    by the RC Contents.

2. If the instruction is decapsulate, then decapsulate the Transit
    Part and re-execute step 1 using the next Transit Part.

3. If the instruction is forward, then retrieve the associated
    Forwarding Information Block (FIB), and go to step 12.

4. If the instruction is to examine the Dest ID, then retrieve the
    FIB associated with the Dest ID, and go to step 12.

5. If the instruction is to examine the FTIF Chain, then retrieve the
    forwardingTable indicated by the rcTable entry, and continue on to
    step 6.

6. Using the value of the currently active FTIF (this is the FTIF
    indicated by the FTIF Offset if this is the first FTIF examined)
    as an index, retrieve one or more of the following instructions
    (steps 7 - 10) from the forwardingTable identified in step 5 or
    step 10.

7. If the instruction is decapsulate, then decapsulate the Pip header
    and re-execute step 1 using the new header (this is the same as
    step 2).

8. If the instruction is forward, then (possibly additionally)
    retrieve the associated FIB, and go to step 12 (this is the same
    as step 3).

9. If the instruction is to examine the Dest ID, then retrieve the
    FIB associated with the Dest ID and go to step 12 (this is the
    same as step 4).

10. If the instruction is to examine the next FTIF, then, according
     to the information in the current forwardingTable entry, modify
     the current FTIF and choose a new forwardingTable.

11. Make the next FTIF the current FTIF and go to step 6.

12. The FIB contains a set of potential recipients for the Pip
     packet, including next hop Pip systems (both directly connected
     and at the end of Pip tunnels) and the upper layer of the local

system. Taking into consideration 1) the incoming interface, 2)
the previous hop Pip system if known (as determined by the
lower-layer source address and incoming interface), and 3)
potentially other local information (such as congestion on
outgoing queues), prune the set of potential recipients. (This
may result in no pruning having taken place or in every potential
next hop having been pruned.)

13. For each remaining next hop, format a Pip header by modifying a)
     the RC, b) the current FTIF, c) the FTIF Offset (to point to 1)
     the FTIF pointed to in the received RD, 2) the current FTIF, 3)
     the Nth FTIF counting from the 0th FTIF, or 4) the Nth FTIF
     counting forwards or backwards from the current FTIF) and d) any
     Pip header encapsulations, according to the information in the
     FIB, and transmit the packet to the recipient (either a next hop
     or upper layer).

2.3 Options Part

The Option Part is formatted as shown in Figure 4.

```
        +============================+
        |    Options Descriptor      |        64
        +============================+
        |        Option 2            |       Variable
        +============================+
        |        Option 3            |       Variable
        +============================+
                      .
                      .
                      .
        +============================+
        |        Option N            |       Variable
        +============================+
```
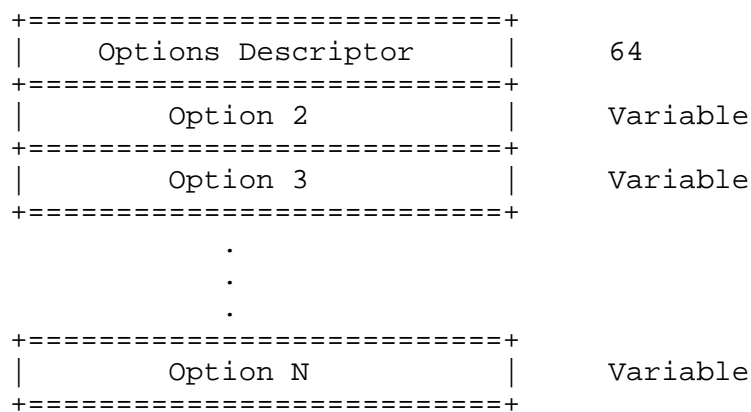
Figure 4: Options Part

Every Option is at least one 32-bit word in length, and ends on a
32-bit word boundary. Because the type of each option is known from
the Options Contents field, there is no need to indicate the option
type in the options field themselves. Thus, there is no common
format among the options--each option has its own format. The
individual options are defined in another specification.

2.3.1 Options Descriptor

The Options Descriptor option gives the offset of each option in the
Options Part. The Options Descriptor consists of eight eight-bit
Option Position fields, each of which gives the position of up to
eight options (there can be no more than 8 Options Part). Each of
the Option Position fields correspond to one of the bits in the
Options Present field. The unit of measure of each Option Position
is 32-bit words, counting the first word of the Options Part as word
0. The high order Option Position field corresponds to the high
order bit in the Options Present field.

Security Considerations

Security issues are not discussed in this memo.

Author's Address

Paul Francis
NTT Software Lab
3-9-11 Midori-cho Musashino-shi
Tokyo 180 Japan

Phone: +81-422-59-3843
Fax +81-422-59-3765
EMail: francis@cactus.ntt.jp