Network Working Group
Request for Comments: 1621
Category: Informational

P. Francis
NTT

May 1994

Pip Near-term Architecture

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.

Preamble

During 1992 and 1993, the Pip internet protocol, developed at Belclore, 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 Header Processing" are to record the ideas (good and bad) of Pip.

This document references a number of Pip draft memos that were in various stages of completion. The basic ideas of those memos are presented in this document, though many details are lost. The very interested reader can obtain those internet drafts by requesting them directly from me at <francis@cactus.ntt.jp>.

The remainder of this document is taken verbatim 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 evolve to all forseeable internet protocol requirements. This specification describes the routing and addressing architecture for near-term Pip deployment. We say near-term only because Pip is designed with evolution in mind, so other architectures are expected in the future. This document, however, makes no reference to such future architectures.

Table of Contents

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 describes the routing and addressing architecture for near-term Pip deployment. We say near-term only because Pip is designed with evolution in mind, so other architectures are expected in the future. This document, however, makes no reference to such future architectures (except in that it discusses Pip evolution in general).

This document gives an overall picture of how Pip operates. It is provided primarily as a framework within which to understand the total set of documents that comprise Pip.

1. Pip Architecture Overview

The Pip near-term architecture is an incremental step from IP. Like IP, near-term Pip is datagram. Pip runs under TCP and UDP. DNS is used in the same fashion it is now used to distribute name to Pip Address (and ID) mappings. Routing in the near-term Pip architecture is hop-by-hop, though it is possible for a host to create a domain-level source route (for policy reasons).

Pip Addresses have more hierarchy than IP, thus improving scaling on one hand, but introducing additional addressing complexities, such as multiple addresses, on the other. Pip, however, uses hierarchical addresses to advantage by making them provider-based, and using them to make policy routing (in this case, provider selection) choices. Pip also provides mechanisms for automatically assigning provider prefixes to hosts and routers in domains. This is the main difference between the Pip near-term architecture and the IP architecture. (Note that in the remainder of this paper, unless otherwise stated, the phrase "Pip architecture" refers to the near-term Pip architecture described herein.)

2. Pip Architecture Characteristics

The proposed architecture for near-term Pip has the following characteristics:

1. Provider-rooted hierarchical addresses.

2. Automatic domain-wide address prefix assignment.

3. Automatic host address and ID assignment.

4. Exit provider selection.

5. Multiple defaults routing (default routing, but to multiple exit points).

6. Equivalent of IP Class D style addressing for multicast.

7. CBT style multicast.

8. "Anycast" addressing (route to one of a group, usually the nearest).

9. Providers support forwarding on policy routes (but initially will not provide the support for sources to calculate policy routes).

10. Mobile hosts.

11. Support for routing across large Public Data Networks (PDN).

12. Inter-operation with IP hosts (but, only within an IP-address domain where IP addresses are unique). In particular, an IP address can be explicitly carried in a Pip header.

13. Operation with existing transport and application binaries (though if the application contains IP context, like FTP, it may only work within a domain where IP addresses are unique).

14. Mechanisms for evolving Pip beyond the near-term architecture.

1.2 Components of the Pip Architecture

The Pip Architecture consists of the following five systems:

1. Host (source and sink of Pip packets)

2. Router (forwards Pip packets)

3. DNS

4. Pip/IP Translator

5. Pip Header Server (formats Pip headers)

The first three systems exist in the IP architecture, and require no explanation here. The fourth system, the Pip/IP Translator, is required solely for the purpose of inter-operating with current IP systems. All Pip routers are also Pip/IP translators.

The fifth system, the Pip Header Server, is new. Its function is to format Pip headers on behalf of the source host (though initially hosts will be able to do this themselves). This use of the Pip Header Server will increase as policy routing becomes more sophisticated (moves beyond near-term Pip Architecture capabilities).

To handle future evolution, a Pip Header Server can be used to "spoon-feed" Pip headers to old hosts that have not been updated to understand new uses of Pip. This way, the probability that the internet can evolve without changing all hosts is increased.

2. A Simple Example

A typical Pip "exchange" is as follows: An application initiates an exchange with another host as identified by a domain name. A request for one or more Pip Headers, containing the domain name of the destination host, goes to the Pip Header Server. The Pip Header Server generates a DNS request, and receive back a Pip ID, multiple Pip Addresses, and possibly other information such as a mobile host server or a PDN address. Given this information, plus information about the source host (its Pip Addresses, for instance), plus optionally policy information, plus optionally topology information, the Pip Header Server formats an ordered list of valid Pip headers and give these to the host. (Note that if the Pip Header Server is co-resident with the host, as will be common initially, the host behavior is similar to that of an IP host in that a DNS request comes from the host, and the host forms a Pip header based on the answer from DNS.)

The source host then begins to transmit Pip packets to the destination host. If the destination host is an IP host, then the Pip packet is translated into an IP packet along the way. Assuming that the destination host is a Pip host, however, the destination host uses the destination Pip ID alone to determine if the packet is destined for it. The destination host generates a return Pip header based either on information in the received Pip header, or the destination host uses the Pip ID of the source host to query the Pip Header Server/DNS itself. The latter case involves more overhead, but allows a more informed decision about how to return packets to the originating host.

If either host is mobile, and moves to a new location, thus getting a new Pip Address, it informs the other host of its new address directly. Since host identification is based on the Pip ID and not the Pip Address, this doesn’t cause transport level to fail. If both hosts are mobile and receive new Pip Addresses at the same time (and thus cannot exchange packets at all), then they can query each other’s respective mobile host servers (learned from DNS). Note that

keeping track of host mobility is completely confined to hosts. Routers never get involved in tracking mobile hosts (though naturally they are involved in host discovery and automatic host address assignment).

3. Pip Overview

Here, a brief overview of the Pip protocol is given. The reader is encouraged to read [2] for a complete description.

The Pip header is divided into three parts:

- Initial Part
- Transit Part
- Options Part

The Initial Part contains the following fields:

- Version Number
- Options Offset, OP Contents, Options Present (OP)
- Packet SubID
- Protocol
- Dest ID
- Source ID
- Payload Length
- Host Version
- Payload Offset
- Hop Count

All of the fields in the Initial Part are of fixed length. The Initial Part is 8 32-bit words in length.

The Version Number places Pip as a subsequent version of IP. The Options Offset, OP Contents, and Options Present (OP) fields tell how to process the options. The Options Offset tells where the options are The OP tells which of up to 8 options are in the options part, so that the Pip system can efficiently ignore options that don't pertain to it. The OP Contents is like a version number for the OP field. It allows for different sets of the (up to 8) options.

The Packet SubID is used to relate a received PCMP message to a previously sent Pip packet. This is necessary because, since routers in Pip can tag packets, the packet returned to a host in a PCMP message may not be the same as the packet sent. The Payload Length and Protocol take the place of IP's Total Length and Protocol fields respectively. The Dest ID identifies the destination host, and is not used for routing, except for where the final router on a LAN uses ARP to find the physical address of the host identified by the dest

ID. The Source ID identifies the source of the packet. The Host Version tells what control algorithms the host has implemented, so that routers can respond to hosts appropriately. This is an evolution mechanism. The Hop Count is similar to IP's Time-to-Live.

The Transit Part contains the following fields:

Transit Part Offset
HD Contents
Handling Directive (HD)
Active FTIF
RC Contents
Routing Context (RC)
FTIF Chain (FTIF = Forwarding Table Index Field)

Except for the FTIF Chain, which can have a variable number of 16-bit FTIF fields, the fields in the Transit Part are of fixed length, and are three 32-bit words in length.

The Transit Part Offset gives the length of the Transit Part. This is used to determine the location of the subsequent Transit Part (in the case of Transit Part encapsulation).

The Handling Directive (HD) is a set of subfields, each of which indicates a specific handling action that must be executed on the packet. Handling directives have no influence on routing. The HD Contents field indicates what subfields are in the Handling Directive. This allows the definition of the set of handling directives to evolve over time. Example handling directives are queueing priority, congestion experienced bit, drop priority, and so on.

The remaining fields comprise the Routing Directive. This is where the routing decision gets made. The basic algorithm is that the router uses the Routing Context to choose one of multiple forwarding tables. The Active FTIF indicates which of the FTIFs to retrieve, which is then used as an index into the forwarding table, which either instructs the router to look at the next FTIF, or returns the forwarding information.

Examples of Routing Context uses are; to distinguish address families (multicast vs. unicast), to indicate which level of the hierarchy a packet is being routed at, and to indicate a Type of Service. In the near-term architecture, the FTIF Chain is used to carry source and destination hierarchical unicast addresses, policy route fragments, multicast addresses (all-of-group), and anycast (one-of-group) addresses. Like the OP Contents and HD Contents fields, the RC Contents field indicates what subfields are in the Routing Context.

This allows the definition of the Routing Context to evolve over time.

The Options Part contains the options. The options are preceded by an array of 8 fields that gives the offset of each of up to 8 options. Thus, a particular option can be found without a serial search of the list of options.

4. Pip Addressing

Addressing is the core of any internet architecture. Pip Addresses are carried in the Routing Directive (RD) of the Pip header (except for the Pip ID, which in certain circumstances functions as part of the Pip Address). Pip Addresses are used only for routing packets. They do not identify the source and destination of a Pip packet. The Pip ID does this. Here we describe and justify the Pip Addressing types.

There are four Pip Address types [11]. The hierarchical Pip Address (referred to simply as the Pip Address) is used for scalable unicast and for the unicast part of a CBT-style multicast and anycast. The multicast part of a CBT-style multicast is the second Pip address type. The third Pip address type is class-D style multicast. The fourth type of Pip address is the so-called "anycast" address. This address causes the packet to be forwarded to one of a class of destinations (such as, to the nearest DNS server).

Bits 0 and 1 of the RC defined by RC Contents value of 1 (that is, for the near-term Pip architecture) indicate which of four address families the FTIFs and Dest ID apply to. The values are:

Value	Address Family
00	Hierarchical Unicast Pip Address
01	Class D Style Multicast Address
10	CBT Style Multicast Address
11	Anycast Pip Address

The remaining bits are defined differently for different address families, and are defined in the following sections.

4.1 Hierarchical Pip Addressing

The primary purpose of a hierarchical address is to allow better scaling of routing information, though Pip also uses the "path" information latent in hierarchical addresses for making provider selection (policy routing) decisions.

The Pip Header encodes addresses as a series of separate numbers, one number for each level of hierarchy. This can be contrasted to traditional packet encodings of addresses, which places the entire address into one field. Because of Pip's encoding, it is not necessary to specify a format for a Pip Address as it is with traditional addresses (for instance, the SIP address is formatted such that the first so-many bits are the country/metro code, the next so-many bits are the site/subscriber, and so on). Pip's encoding also eliminates the "cornering in" effect of running out of space in one part of the hierarchy even though there is plenty of room in another. No "field sizing" decisions need be made at all with Pip Addresses. This makes address assignment easier and more flexible than with traditional addresses.

Pip Addresses are carried in DNS as a series of numbers, usually with each number representing a layer of the hierarchy [1], but optionally with the initial number(s) representing a "route fragment" (the tail end of a policy route--a source route whose elements are providers). The route fragments would be used, for instance, when the destination network's directly attached (local access) provider is only giving access to other (long distance) providers, but the important provider-selection policy decision has to do the long distance providers.

The RC for (hierarchical) Pip Addresses is defined as:

bits	meaning
0,1	Pip Address (= 00)
2,3	level
4,5	metalevel
6	exit routing type

The level and metalevel subfields are used to indicate what level of the hierarchy the packet is currently at (see section 8). The exit routing type subfield is used to indicate whether host-driven (hosts decide exit provider) or router-driven (routers decide exit provider) exit routing is in effect (see section 8.1).

Each FTIF in the FTIF Chain is 16 bits in length. The low-order part of each FTIF in a (hierarchical unicast) Pip Address indicates the relationship of the FTIF with the next FTIF. The three relators are Vertical, Horizontal, and Extension. The Vertical and Horizontal relators indicate if the subsequent FTIF is hierarchically above or below (Vertical) or hierarchically unrelated (Horizontal). The Extension relator is used to encode FTIF values longer than 16 bits.

FTIF values 0 - 31 are reserved for special purposes. That is, they cannot be assigned to normal hierarchical elements. FTIF value 1 is defined as a flag to indicate a switch from the unicast phase of packet forwarding to the anycast phase of packet forwarding.

Note that Pip Addresses do not need to be seen by protocol layers above Pip (though layers above Pip can provide a Pip Address if desired). Transport and above use the Pip ID to identify the source and destination of a Pip packet. The Pip layer is able to map the Pip IDs (and other information received from the layer above, such as QOS) into Pip Addresses.

The Pip ID can serve as the lowest level of a Pip Address. While this "bends the principal" of separating Pip Addressing from Pip Identification, it greatly simplifies dynamic host address assignment. The Pip ID also serves as a multicast ID. Unless otherwise stated, the term "Pip Address" refers to just the part in the Routing Directive (that is, excludes the Pip ID).

Pip Addresses are provider-rooted (as opposed to geographical). That is, the top-level of a Pip Address indicates a network service provider (even when the service provided is not Pip). (A justification of using provider-rooted rather than geographical addresses is given in [12].)

Thus, the basic form of a Pip address is:

```
providerPart,subscriberPart
```

where both the providerPart and subscriberPart can have multiple layers of hierarchy internally.

A subscriber may be attached to multiple providers. In this case, a host can end up with multiple Pip Addresses by virtue of having multiple providerParts:

```
providerPart1,subscriberPart
providerPart2,subscriberPart
providerPart3,subscriberPart
```

This applies to the case where the subscriber network spans many different provider areas, for instance, a global corporate network. In this case, some hosts in the global corporate network will have certain providerParts, and other hosts will have others. The subscriberPart should be assigned such that routing can successfully take place without a providerPart in the destination Pip Address of the Pip Routing Directive (see section 8.2).

Note that, while there are three providerParts shown, there is only one subscriberPart. Internal subscriber numbering should be independent of the providerPart. Indeed, with the Pip architecture, it is possible to address internal packets without including any of the providerPart of the address.

Top-level Pip numbers can be assigned to subscriber networks as well as to providers.

```
privatePart,subscriberPart
```

In this case, however, the top-level number (privatePart) would not be advertised globally. The purpose of such an assignment is to give a private network "ownership" of a globally unique Pip Address space. Note that the privatePart is assigned as an extended FTIF (that is, from numbers greater than 2^15). Because the privatePart is not advertised globally, and because internal packets do not need the prefix (above the subscriberPart), the privatePart actually never appears in a Pip packet header.

Pip Addresses can be prepended with a route fragment. That is, one or more Pip numbers that are all at the top of the hierarchy.

```
longDistanceProvider.localAccessProvider.subscriber
    (top-level)          (top-level)     (next level)
```

This is useful, for instance, when the subscriber's directly attached provider is a "local access" provider, and is not advertised globally. In this case, the "long distance" provider is prepended to the address even though the local access provider number is enough to provide global uniqueness.

Note that no coordination is required between the long distance and local access providers to form this address. The subscriber with a prefix assigned to it by the local access provider can autonomously form and use this address. It is only necessary that the long distance provider know how to route to the local access provider.

4.1.1 Assignment of (Hierarchical) Pip Addresses

Administratively, Pip Addresses are assigned as follows [3]. There is a root Pip Address assignment authority. Likely choices for this are IANA or ISOC. The root authority assigns top-level Pip Address numbers. (A "Pip Address number" is the number at a single level of the Pip Address hierarchy. A Pip Address prefix is a series of contiguous Pip Address numbers, starting at the top level but not including the entire Pip Address. Thus, the top-level prefix is the same thing as the top-level number.)

Though by-and-large, and most importantly, top-level assignments are made to providers, each country is given an assignment, each existing address space (such as E.164, X.121, IP, etc.) is given an assignment, and private networks can be given assignments. Thus, existing addresses can be grandfathered in. Even if the top-level Pip address number is an administrative rather than topological assignment, the routing algorithm still advertises providers at the top (provider) level of routing. That is, routing will advertise enough levels of hierarchy that providers know how to route to each other.

There must be some means of validating top-level number requests from providers (basically, those numbers less than 2^15). That is, top-level assignments must be made only to true providers. While designing the best way to do this is outside the scope of this document, it seems off hand that a reasonable approach is to charge for the top-level prefixes. The charge should be enough to discourage non-serious requests for prefixes, but not so much that it becomes an inhibitor to entry in the market. The charge might include a yearly "rent", and top-level prefixes could be reclaimed when they are no longer used by the provider. Any profit made from this activity could be used to support the overall role of number assignment. Since roughly 32,000 top-level assignments can be made before having to increase the FTIF size in the Pip header from 16 bits to 32 bits, it is envisioned that top-level prefixes will not be viewed as a scarce resource.

After a provider obtains a top-level prefix, it becomes an assignment authority with respect to that particular prefix. The provider has complete control over assignments at the next level down (the level below the top-level). The provider may either assign top-level minus one prefixes to subscribers, or preferably use that level to provide hierarchy within the provider's network (for instance, in the case where the provider has so many subscribers that keeping routing information on all of them creates a scaling problem). It is envisioned that the subscriber will have complete control over number assignments made at levels below that of the prefix assigned it by the provider.

Assigning top level prefixes directly to providers leaves the number of top-level assignments open-ended, resulting in the possibility of scaling problems at the top level. While it is expected that the number of providers will remain relatively small (say less than 10000 globally), this can't be guaranteed. If there are more providers than top-level routing can handle, it is likely that many of these providers will be "local access" providers--providers whose role is to give a subscriber access to multiple "long-distance" providers. In this case, the local access providers need not appear at the top

level of routing, thus mitigating the scaling problem at that level.

In the worst case, if there are too many top-level "long-distance" providers for top-level routing to handle, a layer of hierarchy above the top-level can be created. This layer should probably conform to some policy criteria (as opposed to a geographical criteria). For instance, backbones with similar access restrictions or type-of-service can be hierarchically clustered. Clustering according to policy criteria rather than geographical allows the choice of address to remain an effective policy routing mechanism. Of course, adding a layer of hierarchy to the top requires that all systems, over time, obtain a new providerPart prefix. Since Pip has automatic prefix assignment, and since DNS hides addresses from users, this is not a debilitating problem.

4.1.2 Host Addressing

Hosts can have multiple Pip Addresses. Since Pip Addresses are topologically significant, a host has multiple Pip Addresses because it exists in multiple places topologically. For instance, a host can have multiple Pip addresses because it can be reached via multiple providers, or because it has multiple physical interfaces. The address used to reach the host influences the path to the host.

Locally, Pip Addressing is similar to IP Addressing. That is, Pip prefixes are assigned to subnetworks (where the term subnetwork here is meant in the OSI sense. That is, it denotes a network operating at a lower layer than the Pip layer, for instance, a LAN). Thus, it is not necessary to advertise individual hosts in routing updates--routers only need to advertise and store routes to subnetworks.

Unlike IP, however, a single subnetwork can have multiple prefixes. (Strictly speaking, in IP a single subnetwork can have multiple prefixes, but a host may not be able to recognize that it can reach another host on the same subnetwork but with a different prefix without going through a router.)

There are two styles of local Pip Addressing--one where the Pip Address denotes the host, and another where the Pip Address denotes only the destination subnetwork. The latter style is called ID-tailed Pip Addressing. With ID-tailed Pip Addresses, the Pip ID is used by the last router to forward the packet to the host. It is expected that ID-tailed Pip Addressing is the most common, because it greatly eases address administration.

(Note that the Pip Routing Directive can be used to route a Pip packet internal to a host. For instance, the RD can be used to direct a packet to a device in a host, or even a certain memory

location. The use of the RD for this purpose is not part of this near-term Pip architecture. We note, however, that this use of the RD could be locally done without effecting any other Pip systems.)

When a router receives a Pip packet and determines that the packet is destined for a host on one of its' attached subnetworks (by examining the appropriate FTIF), it then examines the destination Pip ID (which is in a fixed position) and forwards based on that. If it does not know the subnetwork address of the host, then it ARPs, using the Pip ID as the "address" in the ARP query.

4.2 CBT Style Multicast Addresses

When bits 1 and 0 of the RC defined by RC Contents = 1 are set to 10, the FTIF and Dest ID indicate CBT (Core Based Tree) style multicast. The remainder of the bits are defined as follows:

bits	meaning
0,1	CBT Multicast (= 10)
2,3	level
4,5	metalevel
6	exit routing type
7	on-tree bit
8,9	scoping

With CBT (Core-based Tree) multicast, there is a single multicast tree connecting the members (recipients) of the multicast group (as opposed to Class-D style multicast, where there is a tree per source). The tree emanates from a single "core" router. To transmit to the group, a packet is routed to the core using unicast routing. Once the packet reaches a router on the tree, it is multicast using a group ID.

Thus, the FTIF Chain for CBT multicast contains the (Unicast) Hierarchical Pip Address of the core router. The Dest ID field contains the group ID.

A Pip CBT packet, then, has two phases of forwarding, a unicast phase and a multicast phase. The "on-tree" bit of the RC indicates which phase the packet is in. While in the unicast phase, the on-tree bit is set to 0, and the packet is forwarded similarly to Pip Addresses. During this phase, the scoping bits are ignored.

Once the packet reaches the multicast tree, it switches to multicast routing by changing the on-tree bit to 1 and using the Dest ID group address for forwarding. During this phase, bits 2-6 are ignored.

4.3 Class D Style Multicast Addresses

When bits 1 and 0 of the RC defined by RC Contents = 1 are set to 01, the FTIF and Dest ID indicate Class D style multicast. The remainder of the RC is defined as:

bits	meaning
0,1	Class D Style Multicast (= 01)
2-5	Scoping

By "class D" style multicast, we mean multicast using the algorithms developed for use with Class D addresses in IP (class D addresses are not used per se). This style of routing uses both source and destination information to route the packet (source host address and destination multicast group).

For Pip, the FTIF Chain holds the source Pip Address, in order of most significant hierarchy level first. The reason for putting the source Pip Address rather than the Source ID in the FTIF Chain is that use of the source Pip Address allows the multicast routing to take advantage of the hierarchical source address, as is being done with IP. The Dest ID field holds the multicast group. The Routing Context indicates Class-D style multicast. All routers must first look at the FTIF Chain and Dest ID field to route the packet on the tree.

Bits 2 through 5 of the RC are the scoping bits.

4.4 Anycast Addressing

When bits 1 and 0 of the RC defined by RC Contents = 1 are set to 11, the FTIF and Dest ID indicate Anycast addressing. The remainder of the RC is defined as:

bits	meaning
0,1	Anycast Address (= 11)
2,3	level
4,5	metalevel
6	exit routing type
7	anycast active
8,9	scoping

With anycast routing, the packet is unicast, but to the nearest of a group of destinations. This type of routing is used by Pip for autoconfiguration. Other applications, such as discovery protocols, may also use anycast routing.

Like CBT, Pip anycast has two phases of operation, in this case the unicast phase and the anycast phase. The unicast phase is for the purpose of getting the packet into a certain vicinity. The anycast phase is to forward the packet to the nearest of a group of destinations in that vicinity.

Thus, the RC has both unicast and anycast information in it. During the unicast phase, the anycast active bit is set to 0, and the packet is forwarded according to the rules of Pip Addressing. The scoping bits are ignored.

The switch from the unicast phase to the anycast phase is triggered by the presence of an FTIF of value 1 in the FTIF Chain. When this FTIF is reached, the anycast active bit is set to 1, the scoping bits take effect, and bits 2 through 6 are ignored. When in the anycast phase, forwarding is based on the Dest ID field.

5. Pip IDs

The Pip ID is 64-bits in length [4].

The basic role of the Pip ID is to identify the source and destination host of a Pip Packet. (The other role of the Pip ID is for allowing a router to find the destination host on the destination subnetwork.)

This having been said, it is possible for the Pip ID to ultimately identify something in addition to the host. For instance, the Pip ID could identify a user or a process. For this to work, however, the Pip ID has to be bound to the host, so that as far as the Pip layer is concerned, the ID is that of the host. Any additional use of the Pip ID is outside the scope of this Pip architecture.

The Pip ID is treated as flat. When a host receives a Pip packet, it compares the destination Pip ID in the Pip header with its' own. If there is a complete match, then the packet has reached the correct destination, and is sent to the higher layer protocol. If there is not a complete match, then the packet is discarded, and a PCMP Invalid Address packet is returned to the originator of the packet [7].

It is something of an open issue as to whether or not Pip IDs should contain significant organizational hierarchy information. Such information could be used for inverse DNS lookups and allowing a Pip packet to be associated with an organization. (Note that the use of the Pip ID alone for this purpose can be easily spoofed. By cross checking the Pip ID with the Pip Address prefix, spoofing is harder--as hard as it is with IP--but still easy. Section 14.2 discusses methods for making spoofing harder still, without requiring encryption.)

However, relying on organizational information in the Pip header generally complicates ID assignment. This complication has several ramifications. It makes host autoconfiguration of hosts harder, because hosts then have to obtain an assignment from some database somewhere (versus creating one locally from an IEEE 802 address, for instance). It means that a host has to get a new assignment if it changes organizations. It is not clear what the ramifications of this might be in the case of a mobile host moving through different organizations.

Because of these difficulties, the use of flat Pip IDs is currently favored.

Blocks of Pip ID numbers have been reserved for existing numbering spaces, such as IP, IEEE 802, and E.164. Pip ID numbers have been assigned for such special purposes such as "any host", "any router", "all hosts on a subnetwork", "all routers on a subnetwork", and so on. Finally, 32-bit blocks of Pip ID numbers have been reserved for each country, according to ISO 3166 country code assignments.

6. Use of DNS

The Pip near-term architecture uses DNS in roughly the same style that it is currently used. In particular, the Pip architecture maintains the two fundamental DNS characteristics of 1) information stored in DNS does not change often, and 2) the information returned by DNS is independent of who requested it.

While the fundamental use of DNS remains roughly the same, Pip's use of DNS differs from IP's use by degrees. First, Pip relies on DNS to

hold more types of information than IP [1]. Second, Pip Addresses in DNS are expected to change more often than IP addresses, due to reassignment of Pip Address prefixes (the providerPart). To still allow aggressive caching of DNS records in the face of more quickly changing addressing, Pip has a mechanism of indicating to hosts when an address is no longer assigned. This triggers an authoritative query, which overrides DNS caches. The mechanism consists of PCMP Packet Not Delivered messages that indicate explicitly that the Pip Address is invalid.

In what follows, we first discuss the information contained in DNS, and then discuss authoritative queries.

6.1 Information Held by DNS

The information contained in DNS for the Pip architecture is:

1. The Pip ID.

2. Multiple Pip Addresses

3. The destination's mobile host address servers.

4. The Public Data Network (PDN) addresses through which the destination can be reached.

5. The Pip/IP Translators through which the destination (if the destination is IP-only) can be reached.

6. Information about the providers represented by the destination's Pip addresses. This information includes provider name, the type of provider network (such as SMDS, ATM, or SIP), and access restrictions on the provider's network.

The Pip ID and Addresses are the basic units of information required for carriage of a Pip packet.

The mobile host address server tells where to send queries for the current address of a mobile Pip host. Note that usually the current address of the mobile host is conveyed by the mobile host itself, thus a mobile host server query is not usually required.

The PDN address is used by the entry router of a PDN to learn the PDN address of the next hop router. The entry router obtains the PDN address via an option in the Pip packet. If there are multiple PDNs associated with a given Pip Address, then there can be multiple PDN addresses carried in the option. Note that the option is not sent on every packet, and that only the PDN entry router need examine the

option.

The Pip/IP translator information is used to know how to translate an IP address into a Pip Address so that the packet can be carried across the Pip infrastructure. If the originating host is IP, then the first IP/Pip translator reached by the IP packet must query DNS for this information.

The information about the destination's providers is used to help the "source" (either the source host or a Pip Header Server near the source host) format an appropriate Pip header with regards to choosing a Pip Address [14]. The choice of one of multiple Pip Addresses is essentially a policy routing choice.

More detailed descriptions of the use of the information carried in DNS is contained in the relevant sections.

6.2 Authoritative Queries in DNS

In general, Pip treats addresses as more dynamic entities than does IP. One example of this is how Pip Address prefixes change when a subscriber network attaches to a new provider. Pip also carries more information in DNS, any of which can change for various reasons. Thus, the information in DNS is more dynamic with Pip than with IP.

Because of the increased reliance on DNS, there is a danger of increasing the load on DNS. This would be particularly true if the means of increasing DNS' dynamicity is by shortening the cache holding time by decreasing the DNS Time-to-Live (TTL). To counteract this trend, Pip provides explicit network layer (Pip layer) feedback on the correctness of address information. This allows Pip hosts to selectively over-ride cached DNS information by making an authoritative query. Through this mechanism, we actually hope to increase the cache holding time of DNS, thus improving DNS' scaling characteristics overall.

The network layer feedback is in the form of a type of PCMP Packet Not Delivered (PDN) message that indicates that the address used is known to be out-of-date. Routers can be configured with this information, or it can be provided through the routing algorithm (when an address is decommissioned, the routing algorithm can indicate that this is the reason that it has become unreachable, as opposed to becoming "temporarily" unreachable through equipment failure).

Pip hosts consider destination addresses to be in one of four states:

1. Unknown, but assumed to be valid.

2. Reachable (and therefore valid).

3. Unreachable and known to be invalid.

4. Unreachable, but weakly assumed to be valid.

The first state exists before a host has attempted communication with another host. In this state, the host queries DNS as normal (that is, does not make an authoritative query).

The second state is reached when a host has successfully communicated with another host. Once a host has reached this state, it can stay in it for an arbitrarily long time, including after the DNS TTL has expired. When in this state, there is no need to query DNS.

A host enters the third state after a failed attempt at communicating with another host where the PCMP PND message indicates explicitly that the address is known to be invalid. In this case, the host makes an authoritative query to DNS whether or not the TTL has expired. It is this case that allows lengthy caching of DNS information while still allowing addresses to be reassigned frequently.

A host enters the fourth state after a failed attempt at communicating with another host, but where the address is not explicitly known to be invalid. In this state, the host weakly assumes that the address of the destination is still valid, and so can requery DNS with a normal (non-authoritative) query.

7. Type-of-Service (TOS) (or lack thereof)

One year ago it probably would have been adequate to define a handful (4 or 5) of priority levels to drive a simple priority FIFO queue. With the advent of real-time services over the Internet, however, this is no longer sufficient. Real-time traffic cannot be handled on the same footing as non-real-time. In particular, real-time traffic must be subject to access control so that excess real-time traffic does not swamp a link (to the detriment of other real-time and non-real-time traffic alike).

Given that a consensus solution to real- and non-real-time traffic management in the internet does not exist, this version of the Pip near-term architecture does not specify any classes of service (and related queueing mechanisms). It is expected that Pip will define classes of service (primarily for use in the Handling Directive) as solutions become available.

8. Routing on (Hierarchical) Pip Addresses

Pip forwarding in a single router is done based on one or a small number of FTIFs. What this means with respect to hierarchical Pip Addresses is that a Pip router is able to forward a packet based on examining only part of the Pip Address--often a single level.

One advantage to encoding each level of the Pip Address separately is that it makes handling of addresses, for instance address assignment or managing multiple addresses, easier. Another advantage is address lookup speed--the entire address does not have to be examined to forward a packet (as is necessary, for instance, with traditional hierarchical address encoding). The cost of this, however, is additional complexity in keeping track of the active hierarchical level in the Pip header.

Since Pip Addresses allow reuse of numbers at each level of the hierarchy, it is necessary for a Pip router to know which level of the hierarchy it is acting at when it retrieves an FTIF. This is done in part with a hierarchy level indicator in the Routing Context (RC) field. RC level is numbered from the top of the hierarchy down. Therefore, the top of the hierarchy is RC level = 0, the next level down is RC level = 1, and so on.

The RC level alone, however, is not adequate to keep track of the appropriate level in all cases. This is because different parts of the hierarchy may have different numbers of levels, and elements of the hierarchy (such as a domain or an area) may exist in multiple parts of the hierarchy. Thus, a hierarchy element can be, say, level 3 under one of its parents and level 2 under another.

To resolve this ambiguity, the topological hierarchy is superimposed with another set of levels--metalevels [11]. A metalevel boundary exists wherever a hierarchy element has multiple parents with different numbers of levels, or may with reasonable probability have multiple parents with different numbers of levels in the future.

Thus, a metalevel boundary exists between a subscriber network and its provider. (Note that in general the metalevel represents a significant administrative boundary between two levels of the topological hierarchy. It is because of this administrative boundary that the child is likely to have multiple parents.) Lower metalevels may exist, but usually will not.

The RC, then, contains a level and a metalevel indicator. The level indicates the number of levels from the top of the next higher metalevel. The top of the global hierarchy is metalevel 0, level 0. The next level down (for instance, the level that provides a level of

hierarchy within a provider) is metalevel 0, level 1. The first level of hierarchy under a provider is metalevel 1, level 0, and so on.

To determine the RC level and RC metalevel in a transmitted Pip packet, a host (or Pip Header Server) must know where the metalevels are in its own Pip Addresses.

The host compares its source Pip Address with the destination Pip Address. The highest Pip Address component that is different between the two addresses determines the level and metalevel. (No levels higher than this level need be encoded in the Routing Directive.)

Neighbor routers are configured to know if there is a level or metalevel boundary between them, so that they can modify the RC level and RC metalevel in a transmitted packet appropriately.

8.1 Exiting a Private Domain

The near-term Pip Architecture provides two methods of exit routing, that is, routing inter-domain Pip packets from a source host to a network service provider of a private domain [12,15]. In the first method, called transit-driven exit routing, the source host leaves the choice of provider to the routers. In the second method, called host-driven exit routing, the source host explicitly chooses the provider. In either method, it is possible to prevent internal routers from having to carry external routing information. The exit routing bit of the RC indicates which type of exit routing is in effect.

With host-driven exit routing, it is possible for the host to choose a provider through which the destination cannot be reached. In this case, the host receives the appropriate PCMP Packet Not Delivered message, and may either fallback on transit-driven exit routing or choose a different provider.

When using transit-driven exit routing, there are two modes of operation. The first, called destination-oriented, is used when the routers internal to a domain have external routing information, and the host has only one provider prefix. The second, called provider-oriented, is used when the routers internal to a domain do not have any external routing information or when the host has multiple provider prefixes. (With IP, this case is called default routing. In the case of IP, however, default routing does not allow an intelligent choice of multiple exit points.)

With provider-oriented exit routing, the host arbitrarily chooses a source Pip Address (and therefore, a provider). (Note that if the

Pip Header Server is tracking inter-domain routing, then it chooses the appropriate provider.) If the host chooses the wrong provider, then the border router will redirect the host to the correct provider with a PCMP Provider Redirect message.

8.2 Intra-domain Networking

With intra-domain networking (where both source and destination are in the private network), there are two scenarios of concern. In the first, the destination address shares a providerPart with the source address, and so the destination is known to be intra-domain even before a packet is sent. In the second, the destination address does not share a providerPart with the source address, and so the source host doesn't know that the destination is reachable intra-domain. Note that the first case is the most common, because the private top-level number assignment acts as the common prefix even though it isn't advertised globally (see section 4.1).

In the first case, the Pip Addresses in the Routing Directive need not contain the providerPart. Rather, it contains only the address part below the metalevel boundary. (A Pip Address in an FTIF Chain always starts at a metalevel boundary).

For instance, if the source Pip Address is 1.2.3,4.5.6 and the destination Pip Address is 1.2.3,4.7.8, then only 4.7.8 need be included for the destination address in the Routing Directive. (The comma "," in the address indicates the metalevel boundary between providerPart and subscriberPart.) The metalevel and level are set accordingly.

In the second case, it is desirable to use the Pip Header Server to determine if the destination is intra-domain or inter-domain. The Pip Header Server can do this by monitoring intra-domain routing. (This is done by having the Pip Header Server run the intra-domain routing algorithm, but not advertise any destinations.) Thus, the Pip Header Server can determine if the providerPart can be eliminated from the address, as described in the last paragraph, or cannot and must conform to the rules of exit routing as described in the previous section.

If the Pip Header Server does not monitor intra-domain routing, however, then the following actions occur. In the case of host-driven exit routing, the packet will be routed to the stated provider, and an external path will be used to reach an internal destination. (The moral here is to not use host-driven exit routing unless the Pip Header Server is privy to routing information, both internal and external.)

In the case of transit-driven exit routing, the packet sent by the host will eventually reach a router that knows that the destination is intra-domain. This router will forward the packet towards the destination, and at the same time send a PCMP Reformat Transit Part message to the host. This message tells the host how much of the Pip Address is needed to route the packet.

9. Pip Header Server

Two new components of the Pip Architecture are the Pip/IP Translator and the Pip Header Server. The Pip/IP Translator is only used for transition from IP to Pip, and otherwise would not be necessary. The Pip Header Server, however, is a new architectural component.

The purpose of the Pip Header Server is to form a Pip Header. It is useful to form the Pip header in a separate box because 1) in the future (as policy routing matures, for instance), significant amounts of information may be needed to form the Pip header--too much information to distribute to all hosts, and 2) it won't be possible to evolve all hosts at the same time, so the existence of a separate box that can spoon-feed Pip headers to old hosts is necessary. (It is impossible to guarantee that no modification of Pip hosts is necessary for any potential evolution, but being able to form the header in a server, and hand it to an outdated host, is a large step in the right direction.)

(Note that policy routing architectures commonly if not universally require the use of some kind of "route server" for calculating policy routes. The Pip Header Server is, among other things, just this server. Thus, the Pip Header Server does not so much result from the fact that Pip itself is more complex than IP or other "IPv7" proposals. Rather, the Pip Header Server reflects the fact that the Pip Architecture has more functionality than ROAD architectures supported by the simpler proposals.)

We note that for the near-term architecture hosts themselves will by-and-large have the capability of forming Pip headers. The exception to this will be the case where the Pip Header Server wishes to monitor inter-domain routing to enhance provider selection. Thus, the Pip Header Server role will be largely limited to evolution (see section 16).

9.1 Forming Pip Headers

Forming a Pip header is more complex than forming an IP header because there are many more choices to make. At a minimum, one of multiple Pip Addresses (both source and destination) must be chosen [14]. In the near future, it will also be necessary to choose a TOS.

After DNS information about the destination has been received, the the following information is available to the Pip header formation function.

1. From DNS: The destination's providers (either directly connected or nearby enough to justify making a policy decision about), and the names, types, and access restrictions of those providers.

2. From the source host: The application type (and thus, the desired service), and the user access restriction classes.

3. From local configuration: The source's providers, and the names, types, and access restrictions of those providers.

4. Optionally from inter-domain routing: The routes chosen by inter-domain to all top level providers. (Note that inter-domain routing in the Pip near-term architecture is path-vector. Because of this, the Pip Header Server does not obtain enough information from inter-domain routing to form a policy route. When the technology to do this matures, it can be installed into Pip Header Servers.)

   The inter-domain routing information is optional. If it is used, then probably a Pip Header Server is necessary, to limit this information to a small number of systems.

There may also be arbitrary policy information available to the Pip header formation function. This architecture does not specify any such information.

The Pip header formation function then goes through a process of forming an ordered list of source/destination Pip Addresses to use. The ordering is based on knowledge of the application service requirements, the service provided by the source providers, guesses or learned information about the service provided by the destination providers or by source/destination provider pairs, and the cost of using source providers to reach destination providers. It is assumed that the sophistication of forming the ordered list will grow as experienced is gained with internet commercialization and real-time services.

The Pip Header formation function then returns the ordered pairs of source and destination addresses to the source host in the PHP response message, along with an indication of what kind of exit routing to use with each pair. Any additional information, such as PDN Address, is also returned. With this information, the source host can now establish communications and properly respond to PCMP messages. Based on information received from PCMP messages,

particularly PCMP Packet Not Delivered messages but also Mobile Host messages, the host is able to choose appropriately from the ordered list.

Note that if Pip evolves to the point where the Transit Part of the Pip header is no longer compatible with the current Transit Part, and the querying host has not been updated to understand the new Transit Part, then the PHP response message contains a bit map of the Transit Part. The host puts this bit map into the Transit Part of the transmitted Pip header even though it does not understand the semantics of the Transit Part. The Host Version field indicates to the Pip Header Server what kinds of transit parts the host can understand.

9.2 Pip Header Protocol (PHP)

The Pip Header Protocol (PHP) is a simple query/response protocol used to exchange information between the Pip host and the Pip Header Server [6]. In the query, the Pip host includes (among other things) the domain name of the destination it wishes to send Pip packets to. (Thus, the PHP query serves as a substitute for the DNS query.)

The PHP query also contains 1) User Access Restriction Classes, 2) Application Types, and 3) host version. The host version tells the Pip Header Server what features are installed in the host. Thus, the Pip Header Server is able to determine if the host can format its own Pip header based on DNS information, or whether the Pip Header Server needs to do it on behalf of the host. In the future, the PHP query will also contain desired TOS (possibly in lieu of Application Type). (Note that this information could come from the application. Thus, the application interface to PHP (the equivalent of gethostbyname()) must pass this information.)

9.3 Application Interface

In order for a Pip host to generate the information required in the PHP query, there must be a way for the application to convey the information to the PHP software. The host architecture for doing this is as follows.

A local "Pip Header Client" (the source host analog to the Pip Header Server) is called by the application (instead of the current gethostbyname()). The application provides the Pip Header Client with either the destination host domain name or the destination host Pip ID, and other pertinent information such as user access restriction class and TOS. The Pip Header Client, if it doesn’t have the information cached locally, queries the Pip Header Server and receives an answer. (Remember that the Pip Header Server can be co-

resident with the host.)

Once the Pip Header Client has determined what the Pip header(s) are, it assigns a local handle to the headers, returns the handle to the application, and configures the Pip packet processing engine with the handle and related Pip Headers. The application then issues packets to the Pip layer (via intervening layers such as transport) using the local handle.

10. Routing Algorithms in Pip

This section discusses the routing algorithm for use with (hierarchical) Pip Addresses.

The architecture for operating routing algorithms in Pip reflects the clean partitioning of routing contexts in the Pip header. Thus, routing in the Pip architecture is nicely modularized.

Within the Hierarchical Pip Address, there are multiple hierarchical levels. Wherever two routers connect, or two levels interface (either in a single router or between routers), two decisions must be made: 1) what information should be exchanged (that is, what of one side's routing table should be propagated to the other side), and 2) what routing algorithm should be used to exchange the information? The first decision is discussed in section 10.1 below (Routing Information Filtering). The latter decision is discussed here.

Conceptually, and to a large extent in practice, the routing algorithms at each level are cleanly partitioned. This partition is much like the partition between "egp" and "igp" level routing in IP, but with Pip it exists at each level of the hierarchy.

At the top-level of the Pip Address hierarchy, a path-vector routing algorithm is used. Path-vector is more appropriate at the top level than link-state because path-vector does not require agreement between top-level entities (providers) on metrics in order to be loop-free. (Agreement between the providers is likely to result in better paths, but the Pip Architecture does not assume such agreement.)

The top-level path-vector routing algorithm is based on IDRP, but enhanced to handle Pip addresses and Pip idiosyncrasies such as the Routing Context. At any level below the top level, it is a local decision as to what routing algorithm technology to run. However, the path-vector routing algorithm is generalized so that it can run at multiple levels of the Pip Address hierarchy. Thus, a lower level domain can choose to take advantage of the path-vector algorithm, or run another, such as a link-state algorithm. The modified version of

IDRP is called MLPV [10], for Multi-Level Path-Vector (pronounced "milpiv").

Normally, information is exchanged between two separate routing algorithms by virtue of the two algorithms co-existing in the same router. For instance, a border router is likely to participate in an exchange of routing information with provider routers, and still run the routing algorithm of the internal routers. If the two algorithms are different routing technologies (for instance, link-state versus distance-vector) then internal conversion translates information from one routing algorithm to the form of the other.

In some cases, however, two routing algorithms that exchange information will exist in different routers, and will have to exchange information over a link. If these two algorithms are different technologies, then they need a common means of exchanging routing information. While strictly speaking this is a local matter, MLPV can also serve as the interface between two disparate routing algorithms. Thus, all routers should be able to run MLPV, if for no other reason than to exchange information with other, perhaps proprietary, routing protocols.

MLPV is designed to be extendible with regards to the type of routes that it calculates. It uses the Pip Object parameter identification number space to identify what type of route is being advertised and calculated [9]. Thus, to add new types of routes (for instance, new types of service), it is only necessary to configure the routers to accept the new route type, define metrics for that type, and criteria for preferring one route of that type over another.

10.1 Routing Information Filtering

Of course, the main point behind having hierarchical routing is so that information from one part of the hierarchy can be reduced when passed to another. In general, reduction (in the form of aggregation) takes place when passing information from the bottom of the hierarchy up. However, Pip uses tunneling and exit routing to, if desired, allow information from the top to be reduced when it goes down.

When two routers become neighbors, they can determine what hierarchical levels they have in common by comparing Pip Addresses. For instance, if two neighbor routers have Pip Addresses 1.2.3,4 and 1.2.8,9.14 respectively, then they share levels 0 and 1, and are different at levels below that. (0 is the highest level, 1 is the next highest, and so on.) As a general rule, these two routers exchange level 0, level 1, and level 2 routing information, but not level 3 or lower routing information. In other words, both routers

know how to route to all things at the top level (level 0), how to route to all level 1 things with "1" as the level 0 prefix, and how to route to all level 2 things with "1.2" as the level 1 prefix.

In the absence of other instructions, two routers will by default exchange information about all levels from the top down to the first level at which they have differing Pip Addresses. In practice, however, this default exchange is as likely to be followed as not. For instance, assume that 1.2.3,4 is a provider router, and 1.2.8,9.14 is a subscriber router. (Note that 1.2.8 is the prefix given the subscriber by the provider, thus the metalevel boundary indicated by the comma.) Assume also that the subscriber network is using destination-oriented transit-driven exit routing (see section 8.1). Finally, assume that router 1.2.8,9.14 is the subscriber's only entry point into provider 1 (other routers provide entry points to other providers).

In this case, 1.2.8,9.14 does not need to know about level 2 or level 1 areas in the provider (that is, it does not need to know about 1.2.4..., 1.2.5..., or 1.3..., 1.4..., and so on). Thus, 1.2.8,9.14 should be configured to inform 1.2.3,4 that it does not need level 1 or 2 information.

As another example, assume still that 1.2.3,4 is a provider and 1.2.8,9.14 is a subscriber. However, assume now that the subscriber network is using host-driven exit routing. In this case, the subscriber does not even need to know about level 0 information, because all exit routing is directed to the provider of choice, and having level 0 information therefore does not influence that choice.

11. Transition

The transition scheme for Pip has two major components, 1) translation, and 2) encapsulation. Translation is required to map the Pip Address into the IP address and vice versa. Encapsulation is used for one Pip router (or host) to exchange packets with another Pip router (or host) by tunneling through intermediate IP routers.

The Pip transition scheme is basically a set of techniques that allows existing IP "stuff" and Pip to coexist, but within the limitations of IP address depletion (though not within the limitations of IP scaling problems). By this I mean that an IP-only host can only exchange packets with other hosts in a domain where IP numbers are unique. Initially this domain includes all IP hosts, but eventually will include only hosts within a private domain. The IP "stuff" that can exist includes 1) whole IP domains, 2) individual IP hosts, 3) IP-oriented TCPs, and 4) IP-oriented applications.

11.1 Justification for Pip Transition Scheme

Note that all transition to a bigger address require translation. It cannot be avoided. The major choices one must make when deciding on a translation scheme are:

1. Will we require a contiguous infrastructure consisting of the new protocol, or will we allow tunneling through whatever remains of the existing IP infrastructure at any point in time?

2. Will we allow global connectivity between IP machines after IP addresses are no longer globally unique, or not? (In other words, will we use a NAT scheme or not? [15])

Concerning question number 1; while it is desirable to move as quickly as possible to a contiguous Pip (or SIP or whatever) infrastructure, especially for purposes of improved scaling, it is fantasy to think that the whole infrastructure will cut over to Pip quickly. Furthermore, during the testing stages of Pip, it is highly desirable to be able to install Pip in any box anywhere, and by tunneling through IP, create a virtual Pip internet. Thus, it seems that the only reasonable answer to question number 1 is to allow tunneling.

Concerning question number 2; it is highly desirable to avoid using a NAT scheme. A NAT (Network Address Translation) scheme is one whereby any two IP hosts can communicate, even though IP addresses are not globally unique. This is done by dynamically mapping non-unique IP addresses into unique ones in order to cross the infrastructure. NAT schemes have the problems of increased complexity to maintain the mappings, and of translating IP addresses that reside within application data structures (such as the PORT command in FTP).

This having been said, it is conceivable that the new protocol will not be far enough along when IP addresses are no longer unique, and therefore a NAT scheme becomes necessary. It is possible to employ a NAT scheme at any time in the future without making it part of the intended transition scheme now. Thus, we can plan for a NATless transition now without preventing the potential use of NAT if it becomes necessary.

11.2 Architecture for Pip Transition Scheme

The architecture for Pip Transition is that of a Pip infrastructure surrounded by IP-only "systems". The IP-only "systems" surrounding Pip can be whole IP domains, individual IP hosts, an old TCP in an otherwise Pip host, or an old application running on top of a Pip-

smart TCP.

The Pip infrastructure will initially get its internal connectivity by tunneling through IP. Thus, any Pip box can be installed anywhere, and become part of the Pip infrastructure by configuration over a "virtual" IP. Of course, it is desirable that Pip boxes be directly connected to other Pip boxes, but very early on this is the exception rather than the rule.

Two neighbor Pip systems tunneling through IP simply view IP as a "link layer" protocol, and encapsulate Pip over IP just as they would encapsulate Pip over any other link layer protocol. In particular, the hop-count field of Pip is not copied into the Time-to-Live field of IP. There is no automatic configuration of neighbor Pip systems over IP. Manual configuration (and careful "virtual topology" engineering) is required. Note that ICMP messages from a IP router in a tunnel is not translated into a PCMP message and sent on. ICMP messages are sinked at the translating router at the head of the tunnel. The information learned from such ICMP messages, however, may be used to determine unreachability of the other end of the tunnel, and may there result in PCMP message on later packets.

In the remainder of this section, we do not distinguish between a virtual Pip infrastructure on IP, and a pure Pip infrastructure.

Given the model of a Pip infrastructure surrounded by IP, there are 5 possible packet paths:

1. IP - IP

2. IP - Pip - IP

3. IP - Pip

4. Pip - IP

5. Pip - Pip

The first three paths involve packets that originate at IP-only hosts. In order for an IP host to talk to any other host (IP or not), the other host must be addressable within the context of the IP host's 32-bit IP address. Initially, this "IP-unique" domain will include all IP hosts. When IP addresses become no longer unique, the IP-unique domain will include a subset of all hosts. At a minimum, this subset will include those hosts in the IP-host's private domain. However, it makes sense also to arrange for the set of all "public" hosts, basically anonymous ftp servers and mail gateways, to be in this subset. In other words, a portion of IP address space should be

set aside to remain globally unique, even though other parts of the address space are being reused.

11.3 Translation between Pip and IP packets

Paths 2 and 4 involve translation from Pip to IP. This translation is straightforward, as all the information needed to create the IP addresses is in the Pip header. In particular, Pip IDs have an encoding that allows them to contain an IP address (again, one that is unique within an IP-unique domain). Whenever a packet path involves an IP host on either end, both hosts must have IP addresses. Thus, translating from Pip to IP is just a matter of extracting the IP addresses from the Pip IDs and forming an IP header.

Translating from an IP header to a Pip header is more difficult, because the 32-bit IP address must be "translated" into a 64-bit Pip ID and a Pip Address. There is no algorithm for making this translation. A table mapping IP addresses (or, rather, network numbers) to Pip IDs and Pip Addresses is required. Since such a table must potentially map every IP address, we choose to use dynamic discovery and caching to maintain the table. We choose also to use DNS as the means of discovering the mappings.

Thus, DNS contains records that map IP address to Pip ID and Pip Address. In the case where the host represented by the DNS record is a Pip host (packet path 3), the Pip ID and Pip Address are those of the host. In the case where the host represented by the DNS record is an IP-only host (packet paths 2 and 4), the Pip Address is that of the Pip/IP translating gateway that is used to reach the IP host. Thus, an IP-only domain must at least be able to return Pip information in its DNS records (or, the parent DNS domain must be able to do it on behalf of the child).

With paths 2 and 3 (IP-Pip-IP and IP-Pip), the initial translating gateway (IP to Pip) makes the DNS query. It stores the IP packet while waiting for the answer. The query is an inverse address (in-addr) using the destination IP address. The translating gateway can cache the record for an arbitrarily long period, because if the mapping ever becomes invalid, a PCMP Invalid Address message flushes the cache entry.

In the case of path 4 (Pip-IP), however, the Pip Address of the translating gateway is returned directly to the source host--piggybacked on the DNS record that is normally returned. Thus this scheme incurs only a small incremental cost over the normal DNS query.

11.4 Translating between PCMP and ICMP

The only ICMP/PCMP messages that are translated are the Destination Unreachable, Echo, and PTMU Exceeded messages. The portion of the offending IP/Pip header that is attached to the ICMP/PCMP message is not translated.

11.5 Translating between IP and Pip Routing Information

It is not necessary to pass IP routing information into the Pip infrastructure. The mapping of IP address to Pip Address in DNS allows Pip to find the appropriate gateway to IP in the context of Pip addresses only.

It is impossible to pass Pip routing information into IP routers, since IP routers cannot understand Pip addresses. IP domains must therefore use default routing to reach IP/Pip translators.

11.6 Old TCP and Application Binaries in Pip Hosts

A Pip host can be expected to have an old TCP above it for a long time to come, and a new (Pip-smart) TCP can be expected to have old application binaries running over it for a long time to come. Thus, we must have some way of insuring that the TCP checksum is correctly calculated in the event that one or both ends is running Pip, and one or both ends has an old TCP binary. In addition, we must arrange to allow applications to interface with TCP using a 32-bit "address" only, even though those 32 bits get locally translated into Pip Addresses and IDs.

As stated above, in all cases where a Pip host is talking to an IP-only host, the Pip host has a 32-bit IP address. This address is embedded in the Pip ID such that it can be identified as an IP address from inspection of the Pip ID alone.

The TCP pseudo-header is calculated using the Payload Length and Protocol fields, and some or all of the Source and Dest Pip IDs. In the case where both Source and Dest Pip IDs are IP-based, only the 32-bit IP address is included in the pseudo-header checksum calculation. Otherwise, the full 64 bits are used. (Note that using the full Payload Length and Protocol fields does not fail when old TCP binaries are being used, because the values for those fields must be within the 16-bit and 8-bit limits for TCP to correctly operate.)

The reason for only using 32 bits of the Pip ID in the case of both ends using an IP address is that an old TCP will use only 32 bits of some number to form the pseudo-header. If the entire 64 bits of the Pip ID were used, then there would be cases where no 32-bit number

could be used to insure that the correct checksum is calculated in all cases.

Note that in the case of an old TCP on top of Pip, "Pip" (actually, a Pip daemon) must create a 32-bit number that can both be used to 1) allow the Pip layer to correctly associate a packet from the TCP layer with the right Pip header, and 2) cause the TCP layer to calculate the right checksum. (This number is created when the application initiates a DNS query. A Pip daemon intervenes in this request, calculates a 32 bit number that the application/TCP can use, and informs the Pip layer of the mapping between this 32 bit number and the full Pip header.)

When the destination host is an IP only host, then this 32-bit number is nothing more than the IP address. When the destination host is a Pip host, then this 32-bit number is some number generated by Pip to "fool" the old TCP into generating the right checksum. This 32-bit number can normally be the same as the lower 32 bits of the Pip ID. However, it is possible that two or more active TCP connections is established to different hosts whose Pip IDs have the same lower 32 bits. In this case, the Pip layer must generate a different 32-bit number for each connection, but in such a way that the sum of the two 16-bit components of the 32-bit numbers are the same as the sum of the two 16-bit components of the lower 32 bits of the Pip IDs.

In the case where an old Application wants to open a socket using an IP address handed to it (by the user or hard-coded), and not using a domain name, then it must be assumed that this IP address is valid within the IP-unique domain. To form a Pip ID out of this 32-bit number, the host appends the high-order 24 bits of its own Pip ID, plus the IP-address-identifier-byte value, to the 32-bit IP address.

11.7 Translating between Pip Capable and non-Pip Capable DNS Servers

In addition to transitioning "Pip-layer" packets, it is necessary to transition DNS from non-Pip capable to Pip capable. During transition there will be name servers in DNS that only understand IP queries and those that understand both Pip and IP queries. This means there must be a mechanism for Pip resolvers to detect whether a name server is Pip capable, and vice versa. Also, a name server, if it provides recursive service, must be able to translate Pip requests to IP requests. (Pip-capable means a name server understands Pip and existing IP queries. It does not necessarily mean the name server uses the Pip protocol to communicate.)

New resource records have been defined to hold Pip identifiers and addresses, and other information [1]. These resource records must be queried using a new opcode in the DNS query packet header. Existing

resource records can be queried using both the old and new header formats. Name servers that are not Pip-capable will respond with a format error to queries with the new opcode. Thus, a resolver can determine dynamically whether a name server is Pip-capable, by sending it a Pip query and noting the response. This only need be done once, when querying a server for the "first" time, and the outcome can be cached along with the name server's address.

Using a new opcode for making Pip queries also helps name servers determine whether a resolver is Pip-capable (it is not always not obvious from the type of query made since many queries are common to to IP and Pip). Determining whether a resolver is Pip-capable is necessary when responding with address information that is not explicitly requested by the query. An important example of this is when a name server makes a referral to another name server in a response: if the request comes from a Pip resolver, name server addresses will be returned as Pip identifier/address resource records, otherwise the addresses will be returned as IP A resource records.

Those servers that are Pip-capable and provide recursive service must translate Pip requests to IP requests when querying an IP name server. For most queries, this will just mean modifying the opcode value in the query header to reflect an IP query, rather than a Pip query. (Most queries are identical in IP and Pip.) Other queries, notably the query for Pip identifier/address information, must be translated into its IP counterpart, namely, an IP A query. On receipt of an answer from an IP name server, a Pip name server must translate the query header and question section back to its original, and format the answer appropriately. Again, for most queries, this will be a trivial operation, but responses containing IP addresses, either as a result of an explicit query or as additional information, must be formatted to appear as a valid Pip response.

Pip-capable name servers that provide recursive name service should also translate IP address requests into Pip identifier/address requests when querying a Pip-capable name server. (A host's IP address can be deduced from the host's Pip identifier.) This enables a Pip-capable name server to cache all relevant addressing information about a Pip host in the first address query concerning the host. Caching partial information is undesirable since the name server, using the current DNS caching strategy, would return only the cached information on a future Pip request, and IP, rather than Pip, would be used to communicate with the destination host.

12. Pip Address and ID Auto-configuration

One goal of Pip is to make networks as easy to administer as possible, especially with regards to hosts. Certain aspects of the Pip architecture make administration easier. For instance, the ID field provides a network layer "anchor" around which address changes can be administered.

This section discusses three aspects of autoconfiguration; 1) domain-wide Pip Address prefix assignment, 2) host Pip Address assignment, and 3) host Pip ID assignment.

12.1 Pip Address Prefix Administration

A central premise behind the use of provider-rooted hierarchical addresses is that domain-wide address prefix assignment and re-assignment is straight-forward. This section describes that process.

Pip Address prefix administration limits required manual prefix configuration to DNS and border routers. This is the minimum required manual configuration possible, because both border routers and DNS must be configured with prefix information for other reasons. DNS must be configured with prefix information so that it can reply to address queries. DNS files are structured so that the prefix is administered in only one place (that is, every host record does not have to be changed to create a new prefix). Border routers must be configured with prefix information in order to advertise exit routes internally.

Note in particular that no internal (non-border) routers or hosts need ever be manually configured with any externally derived addressing information. All internal routers that are expected to fall under a common provider-prefix must, however, be configured with a "group ID" taken from the Pip ID space. (This group ID is not a multicast ID per se. Rather, it is an identifier that allows prefix updates to be targetted to a specific set of routers.)

Each border router is configured with the following information.

1. The type of exit routing for the domain. This tells the border router whether or not it needs to advertise external routes internally.

2. The address prefix of the providers that the border is directly connected to. This prefix information includes any metalevel boundaries above the subscriber/provider metalevel boundary (called simply the subscriber metalevel).

3. Other information about the provider (provider name, type, user access restriction classes).

4. A list of common-provider-prefix group IDs that should receive the auto-configuration information. (The default is that only systems that share a group ID with the border router will receive the information.)

This information is injected into the intra-domain routing algorithm. It is automatically spread to all routers indicated by the group ID list. This way, the default behavior is for the information to be automatically constrained to the border router's "area".

When a non-border router receives this information, it 1) records the route to the providers in its forwarding table, and 2) advertises the information to hosts in the router discovery protocol [8]. Thus hosts learn not only their complete address, but also information on how to do exit routing and on how to choose source addresses.

12.2 Host Autoconfiguration

There are three phases of host autoconfiguration:

1. The host locally creates a flat unique Pip ID (probably globally unique but at least unique on the attached subnet).

2. The host learns its Pip Addresses.

3. The host optionally obtains a hierarchical, organizationally meaningful Pip ID and a domain name from a Pip ID/domain name assignment service. This service updates DNS.

Item three is optional. If Pip ID and domain name assignment services are not installed, then the host must obtain its domain name and, if necessary, Pip ID, from static configuration. Each of the three phases are described below.

12.2.1 Host Initial Pip ID Creation

When a host boots, it can form an ID based only on local information. If the host has an IEEE 802 number, either from an IEEE 802 interface or from an internal identifier, then it can create a globally unique Pip ID from the IEEE 802 Pip ID type [4]. Otherwise, the host can create an ID from the IEEE 802 space using its subnet (link layer) address. This latter ID is only guaranteed to be locally unique.

12.2.2 Host Pip Address Assignment

Unless a host does not wish to use ID-tailed Pip Addresses (see section 4.1.2), host Pip Address assignment is trivial. (The near-term Pip Architecture doesn't specify a means for a host to obtain a non-ID-tailed Pip Address.) When a host attaches to a subnet, it learns the Pip Address of the attached routers through router discovery.

The host simply adopts these Pip Addresses as its own. The Pip Address gets a packet to the host's subnet, and the host's Pip ID is used to route across the subnet. When the routers advertise new addresses (for instance, because of a new provider), the host adopts the new addresses.

12.2.3 Pip ID and Domain Name Assignment

Once the host has obtained its Pip Addresses and an at-least-locally-unique Pip ID, it can exchange packets with an ID/Domain Name (ID/DN) assignment service. If the host locally created a globally unique Pip ID (using an IEEE 802 number), and the organization it belongs to does not use organizationally structured Pip IDs (which should normally be the case) then it only needs to obtain a domain name. The ID/DN assignment service is reachable at a well-known anycast address [4]. Thus, the host is able to start exchanging packets with the ID/DN assignment service without any additional configuration.

If there is no ID/DN assignment service available, then the host must obtain it's organizational ID or DNS name in a non-automatic way. If the ID/DN assignment service is down, the host must temporarily suffice with just a Pip ID and Address. The host can periodically try to reach the ID/DN assignment service.

The ID/DN assignment service must coordinate with DNS. When the ID/DN assignment service creates a new ID or domain name to assign to a new host, it must know which IDs and domain names are available for assignment. It must also update DNS with the new information.

The design of this service is left for further study.

13. Pip Control Message Protocol (PCMP)

The Pip analog to ICMP is PCMP [7]. The near-term Pip architecture defines the following PCMP messages:

1. Local Redirect
2. Packet Not Delivered
3. Echo
4. Parameter Problem
5. Router Discovery
6. PMTU Exceeded
7. Provider Redirect
8. Reformat Transit Part
9. Unknown Parameter
10. Host Mobility
11. Exit PDN Address

The Local Redirect, Echo, and Parameter Problem PCMP messages operate almost identically to their ICMP counterparts.

The Packet Not Delivered PCMP message serves the role of ICMP's Destination Unreachable. The Packet Not Delivered, has two major differences. First, it is more general in that it indicates the hierarchy level of unreachability (rather than explicit host, subnet, network unreachability as with IP). Second, it indicates when an address is known to be invalid, thus allowing for more intelligent use of DNS (see section 6.2).

The Router Discovery PCMP message operates as ICMP's, with the exception that a host derives its Pip Address from it.

The PMTU Exceeded message operates as ICMP's, with the exception that the Pip header size of the offending Packet is also given. This allows the source host transport to determine how much smaller the packet PMTU should be from the advertised subnet PMTU. Note that if an occasional option, such as the PDN Address option, needs to be attached to one of many packets, and that this option makes the packet larger than the PMTU, then it is not necessary to modify the

MTU coming from transport. Instead, that packet can be fragmented by the host's Pip forwarding engine. (Pip specifies fragmentation/reassembly for hosts but not for routers. The fragmentation information is in a Pip Option.)

The Provider Redirect, Invalid Address, Reformat Transit Part, Unknown Parameter, Host Mobility, and Exit PDN Address PCMP messages are new.

The Provider Redirect PCMP message is used to inform the source host of a preferable exit provider to use when provider-rooted, transit-driven exit routing is used (see section 8.1).

The Invalid Address PCMP message is used to inform the source host that none of the IDs of the destination host match that of the Pip packet. The purpose of this message is to allow for authoritative DNS requests (see section 6.2).

The Reformat Transit Part PCMP message has both near-term Pip architecture functions and evolution functions. Near-term, the Reformat Transit Part PCMP message is used to indicate to the source whether it has too few or too many layers of address in the Routing Directive (see section 8.2). Long-term, the Reformat Transit Part PCMP message is able to arbitrarily modify the transit part transmitted by the host, as encoded by a bit string.

The Unknown Parameter PCMP message is used to inform the source host that the router does not understand a parameter in either the Handling Directive, the Routing Context, or the Transit Options. The purpose of this message is to assist evolution (see section 16.1).

The Host Mobility PCMP message is sent by a host to inform another host (for instance, the host's Mobile Address Server) that it has a new address (see section 14). The main use of this packet is for host mobility, though it can be used to manage any address changes, such as because of a new prefix assignment.

The Exit PDN Address PCMP message is used to manage the function whereby the source host informs the PDN entry router of the PDN Address of the exit PDN system (see section 15).

When a router needs to send a PCMP message, it sends it to the source Pip Address. If the Pip header is in a tunnel, then the PCMP message is sent to the router that is the source of the tunnel. Depending on the situation, this may result in another PCMP message from the source of the tunnel to the true source (for instance, if the source of the tunnel finds that the dest of the tunnel can't be reached, it may send a Packet Not Delivered to the source host).

14. Host Mobility

Depending on how security conscience a host is, and what security mechanisms a host has available, mobility can come from Pip "for free". If a host is willing to accept a packet by just looking at source and destination Pip ID, and if the host simply records the source Pip Address on any packet it receives as the appropriate return address to the source Pip ID, then mobility comes automatically.

That is, when a mobile host gets a new Pip Address, it simply puts that address into the next packet it sends. When the other host receives it, it records the new Pip Address, and starts sending return packets to that address. The security aspect of this is that this type of operation leads to an easy way to spoof the (internet level) identity of a host. That is, absent any other security mechanisms, any host can write any Pip ID into a packet. (Cross-checking a source Pip ID against the source Pip Address at least makes spoofing of this sort as hard as with IP. This is discussed below.)

The above simple host mobility mechanism does not work in the case where source and destination hosts obtain new Pip Addresses at the same time and the old Pip addresses no longer work, because neither is able to send its new address information directly to the other. Furthermore, if a host wishes to be more secure about authenticating the source Pip ID of a packet, then the above mechanism also is not satisfactory. In what follows, the complete host mobility mechanism is described.

Pip uses the Mobile Host Server and the PCMP Host Mobility message to manage host mobility;

The Mobile Host Server is a non-mobile host (or router acting as a host) that keeps track of the active address of a mobile host. The Pip ID and Address of the Mobile Host Server is configured into the mobile host, and in DNS. When a host X obtains information from DNS about a host Y, the Pip ID and Address of host Y's Mobile Host Server is among the information. (Also among the information is host Y's "permanent" address, if host Y has one. If host Y is so mobile that it doesn't have a permanent address, then no permanent address is returned by DNS. In particular, note that DNS is not intended to keep track of a mobile host's active address.)

Given the destination host's (Y) permanent ID and Address, and the Mobile Host Server's permanent IDs and Addresses, the source host (X) proceedes as follows. X tries to establish communications with Y using one of the permanent addresses. If this fails (or if at any

time X cannot contact Y), X sends a PCMP Mobile Host message to the Mobile Host Server requesting the active address for Y. (Note that X can determine that it cannot contact Y from receipt of a PCMP Destination Unreachable or a PCMP Invalid Address message.)

The Mobile Host Server responds to X with the active Pip Addresses of Y. (Of course, Y must inform its Mobile Host Server(s) of its active Pip Addresses when it knows them. This also is done using the PCMP Mobile Host message. Y also informs any hosts that it is actively communicating with, using either a regular Pip packet or with a PCMP Mobile Host message. Thus, usually X does not need to contact the Mobile Host Server to track Y's active address.)

If the address that X already tried is among those returned by Y, then the source host has the option of either 1) continuing to try the same Pip Address, 2) trying another of Y's Pip Addresses, 3) waiting and querying the Mobile Host Server again, or 4) giving up.

If the Mobile Host Server indicates that Y has new active Pip Addresses, then X chooses among these in the same manner that it chooses among multiple permanent Pip Addresses, and tries to contact Y.

14.1 PCMP Mobile Host message

There are two types of PCMP Mobile Host messages, the query and the response. The query consists of the Pip ID of the host for which active Pip Address information is being requested.

The response consists of a Pip ID, a sequence number, a set of Pip Addresses, and a signature field. The set of Pip Addresses includes all currently usable addresses of the host indicated by the Pip ID. Thus, the PCMP Mobile Host message can be used both to indicate a newly obtained address, and to indicate that a previous address is no longer active (by that addresses' absence in the set).

The sequence number indicates which is the most recent information. It is needed to deal with the case where an older PCMP Mobile Host response is received after a newer one.

The signature field is a value that derives from encrypting the sequence number and the set of Pip Addresses. For now, the encryption algorithms used, how to obtain keys, and so on are for further study.

14.2 Spoofing Pip IDs

This section discusses host mechanisms for decreasing the probability of Pip ID spoofing. The mechanisms provided in this version of the near-term Pip architecture are no more secure than DNS itself. It is hoped that mechanisms and the corresponding infrastructure needed for better internetwork layer security can be installed with whatever new IP protocol is chosen.

After a host makes a DNS query, it knows:

1. The destination host's Pip ID,

2. The destination host's permanent Pip Addresses, and

3. The destination host's Mobile Host Server's Pip ID and Addresses.

Note that the DNS query can be a normal one (based on domain name) or an inverse query (based on Pip ID or Pip Address, though the latter is more likely to succeed, since the Pip ID may be flat and therefore not suitable for an inverse lookup). The inverse query is done when the host did not initiate the packet exchange, and therefore doesn't know the domain name of the remote (initiating) host.

If the destination host is not mobile, then the source host can check the source Pip Address, compare it with those received from DNS, and reject the packet if it does not match. This gives spoof protection equal to that of IP.

If the destination host is mobile and obtains new Pip Addresses, then the source host can check the validity of the new Pip Address by sending a PCMP Mobile Host query to the Mobile Host Server learned from DNS. The set of Pip Addresses learned from the Mobile Address Server is then used for subsequent validation.

15. Public Data Network (PDN) Address Discovery

One of the problems with running Pip (or any internet protocol) over a PDN is that of the PDN entry Pip System discovering the PDN Address of the appropriate PDN exit Pip System. This problem is solved using ARP in small, broadcast LANs because the broadcast mechanism is relatively cheap. This solution is not available in the PDN case, where the number of attached systems is very large, and where broadcast is not available (or is not cheap if it is).

For the case where the domain of the destination host is attached to a PDN, the problem is nicely solved by distributing the domain's exit PDN Address information in DNS, and then having the source host

convey the exit PDN Address to the PDN entry router in a Pip option.

The DNS of the destination host's domain contains the PDN Addresses for the domain. When DNS returns a record for the destination host, the record associates zero or more PDN Addresses with each Pip Address. There can be more than one PDN address associated with a given PDN, and there can be more than on PDN associated with a given Pip Address. This latter case occurs when more than one hierarchical component of the Pip Address each represents a separate PDN. It is expected that in almost all cases, there will be only one (or none) PDN associated with any Pip address.

(Note that, while the returned DNS record associates the PDN Addresses with a single Pip Address, in general the PDN Address will apply to a set of Pip Addresses--those for all hosts in the domain. The DNS files are structured to reflect this grouping in the same way that a single Pip Address prefix in DNS applies to many hosts. Therefore, every individual host entry in the DNS files does not need to have separate PDN Addresses typed in with it. This simplifies configuration of DNS.)

When the source host sends the first packet to a given destination host, it attaches the PDN Addresses, one per PDN, to the packet in an option. (Note that, because of the way that options are processed in Pip packets, no router other than the entry PDN router need look at the option.) When the entry router receives this packet, it determines that it is the entry router based on the result of the FTIF Chain lookup.

It retrieves the PDN Address from the option, and caches it locally. The cache entry can later by retrieved using either the destination Pip ID or the destination Pip Address as the cache index.

The entry router sends the source host a PCMP Exit PDN Address message indicating that it has cached the information. If there are multiple exit PDN Addresses, then the source host can at this time inform the entry PDN router of all the PDN addresses. The entry PDN router can either choose from these to setup a connection, or cache them to recover from the case where the existing connection breaks.

Finally, the entry PDN router delivers the Pip packet (perhaps by setting up a connection) to the PDN Address indicated.

When a PDN entry router receives a Pip packet for which it doesn't know the exit PDN address (and has no other means of determining it, such as shortcut routing), it sends a PCMP Exit PDN Address query message to the originating host. This can happen if, for instance, routing changes and directs the packets to a new PDN entry router.

When the source host receives the PCMP Exit PDN Address query message, it transmits the PDN Addresses to the entry PDN router.

15.1 Notes on Carrying PDN Addresses in NSAPs

The Pip use of PDN Address carriage in the option or PCP Exit PDN Address message solves two significant problems associated with the analogous use of PDN Address-based NSAPs.

First, there is no existing agreement (standards or otherwise) that the existence of of a PDN Address in an NSAP address implies that the identified host is reachable behind the PDN Address. Thus, upon receiving such an NSAP, the entry PDN router does not know for sure, without explicit configuration information, whether or not the PDN Address can be used at the lower layer. Solution of this problem requires standards body agreement, perhaps be setting aside additional AFIs to mean "PDN Address with topological significance".

The second, and more serious, problem is that a PDN Address in an NSAP does not necessarily scale well. This is best illustrated with the E.164 address. E.164 addresses can be used in many different network technologies--telephone network, BISDN, SMDS, Frame Relay, and other ATM. When a router receives a packet with an E.164-based NSAP, the E.164 address is in the most significant part of the NSAP address (that is, contains the highest level routing information). Thus, without a potentially significant amount of routing table information, the router does not know which network to send the packet to. Thus, unless E.164 addresses are assigned out in blocks according to provider network, it won't scale well.

A related problem is that of how an entry PDN router knows that the PDN address is meant for the PDN it is attached to or some other PDN. With Pip, there is a one-to-one relationship between Pip Address prefix and PDN, so it is always known. With NSAPs, it is not clear without the potentially large routing tables discussed in the previous paragraph.

16. Evolution with Pip

The fact that we call this architecture "near-term" implies that we expect it to evolve to other architectures. Thus it is important that we have a plan to evolve to these architectures. The Pip near-term architecture includes explicit mechanisms to support evolution.

The key to evolution is being able to evolve any system at any time without destroying old functionality. Depending on what the new functionality is, it may be immediately useful to any system that installs it, or it may not become useful until a significant number

or even a majority of systems install it. None-the-less, it is necessary to be able to install it piece-wise.

The Pip protocol itself supports evolution through the following mechanisms [2]:

1. Tunneling. This allows more up-to-date routers to tunnel less up-to-date routers, thus allowing for incremental router evolution. (Of course, by virtue of encapsulation, tunneling is always an evolution option, and indeed tunneling through IP is used in the Pip transition. However, Pip's tunneling encoding is efficient because it doesn't duplicate header information.)

   The only use for Pip tunneling in the Pip near-term architecture is to route packets through the internal routers of a transit domain when the internal routers have no external routing information. It is assumed that enhancements to the Pip Architecture that require tunneling will have their own means of indicating when forming a tunnel is necessary.

2. Host independence from routing information. Since a host can receive packets without understanding the routing content of the packet, routers can evolve without necessarily requiring hosts to evolve at the same pace.

   In order to allow hosts to send Pip packets without understanding the contents of the routing information (in the Transit Part), the Pip Header Server is able to "spoon-feed" the host the Pip header.

   If the Pip Header Server determines that the host is able to form its own Pip header (as will usually be the case with the near-term Pip architecture), the Pip Header is essentially a null function. It accepts a query from the host, passes it on to DNS, and returns the DNS information to the host.

   If the Pip Header Server determines that the host is not able to form its own Pip header, then the Pip Header Server forms one on behalf of the host. In one mode of operation, the Pip Header Server gives the host the values of some or all Transit Part fields, and the host constructs the Transit Part. This allows for evolution within the framework of the current Transit Part. In another mode, the Pip Header Server gives the host the Transit Part as a simple bit field. This allows for evolution outside the framework of the current Transit Part.

   In addition to the Pip Header Server being able to spoon-feed the host a Transit Part, routers are also able to spoon-feed hosts a Transit Part, in case the original Transit Part needs to be

modified, using the PCMP Reformat Transit Part message.

3. Separation of handling from routing. This allows one aspect to evolve independently of the other.

4. Flexible Handling Directive, Routing Context, and Options definition. This allows new handling, routing, and option types to be added and defunct ones to be removed over time (see section 16.1 below).

5. Fast and general options processing. Options processing in Pip is fast, both because not every router need look at every option, and because once a router decides it needs to look at an option, it can find it quickly (does not require a serial search). Thus the oft-heard argument that a new option can't be used because it will slow down processing in all routers goes away.

Pip Options can be thought of as an extension of the Handling Directive (HD). The HD is used when the handling type is common, and can be encoded in a small space. The option is used otherwise. It is possible that a future option will influence routing, and thus the Option will be an extension of the RD as well. The RD, however, is rich enough that this is unlikely.

6. Generalized Routing Directive. Because the Routing Directive is so general, it is more likely that we can evolve routing and addressing semantics without having to redefine the Pip header or the forwarding machinery.

7. Host version number. This number tells what Pip functions a host has, such as which PCMP messages it can handle, so that routers can respond appropriately to a Pip packet received from a remote host. This supports the capability for routers to evolve ahead of hosts. (All Pip hosts will at least be able to handle all Pip near-term architecture functions.)

The Host version number is also used by the Pip Header Server to determine the extent to which the Pip Header Server needs to format a header on behalf of the host.

8. Generalized Route Types. The IDRP/MLPV routing algorithm is generic with regards to the types of routes it can calculate. Thus, adding new route types is a matter of configuring routers to accept the new route type, defining metrics for the new route type, and defining criteria for selecting one route of the new type over another.

Note that none of these evolution features of Pip significantly slow down Pip header processing (as compared to other internet protocols).

16.1 Handling Directive (HD) and Routing Context (RC) Evolution

Because the HD and RC are central to handling and routing of a Pip packet, the evolution of these aspects deserves more discussion.

Both the HD and the RC fields contain multiple parameters. (In the case of the RC, the router treats the RC field as a single number, that is, ignores the fact that the RC is composed of multiple parameters. This allows for fast forwarding of Pip packets.) These HD and RC multiple parameters may be arranged in any fashion (can be any length, subject to the length of the HD and RC fields themselves, and can fall on arbitrary bit boundaries).

Associated with the HD and RC are "Contents" fields that indicate what parameters are in the HD and RC fields, and where they are. (The Contents fields are basically version numbers, except that a higher "version" number is not considered to supersede a lower one. Typical types of parameters are address family, TOS value, queueing priority, and so on.)

The Contents field is a single number, the value of which indicates the parameter set. The mapping of Contents field value to parameter set is configured manually.

The procedure for establishing new HD or RC parameter sets (or, erasing old ones) is as follows. Some organization defines the new parameter set. This may involve defining a new parameter. If it does, then the new parameter is described as a Pip Object. A Pip Object is nothing more than a number space used to unambiguously identify a new parameter type, and a character string that describes it [9].

Thus, the new parameter set is described as a list of Pip Objects, and the bit locations in the HD/RC that each Pip Object occupies. The organization that defines the parameter set submits it for an official Contents field value. (It would be submitted to the standards body that has authority over Pip, currently the IAB.) If the new parameter set is approved, it is given a Contents value, and that value is published in a well known place (an RFC).

Of course, network administrators are free to install or not install the new parameter set in their hosts and routers. In the case of a new RC parameter set, installation of the new parameter set does not necessarily require any new software, because any Pip routing protocol, such as IDRP/MLPV, is able to find routes according to the

new parameter set by appropriate configuration of routers.

In the case of a new HD parameter set, however, new software is necessary--to execute the new handling.

For new HD and RC parameters sets, systems that do not understand the new parameter set can still be configured to execute one of several default actions on the new parameter. These default action allow for some control over how new functions are introduced into Pip systems. The default actions are:

1. Ignore the unknown parameter,

2. Set unknown parameter to all 0's,

3. Set unknown parameter to all 1's,

4. Silently discard packet,

5. Discard packet with PCMP Parameter Unknown.

Action 1 is used when it doesn't much matter if previous systems on a path have acted on the parameter or not. Actions 2 and 3 are used when systems should know whether a previous system has not understood the parameter. Actions 4 and 5 are used when something bad happens if not all systems understand the new parameter.

16.1.1 Options Evolution

The evolution of Options is very similar to that of the HD and RC. Associated with the Options is an Options Present field that indicates in a single word which of up to 8 options are present in the Options Part. There is a Contents field associated with the Options Present field that indicates which subset of all possible options the Options Present field refers to. Contents field values are assigned in the same way as for the HD and RC Contents fields.

The same 5 default actions used for the HD and RC also apply to the Options.

References

[1] Thomson, F., "Use of DNS with Pip", Work in Progress.
[2] Francis, P., "Pip Header Processing", Work in Progress.
[3] Pip Address Assignment Specification, Work in Progress.
[4] Francis, P., "Pip Identifiers", Work in Progress.
[5] Pip Assigned Numbers, Work in Progress.
[6] Pip Header Protocol, Work in Progress.
[7] Francis, G., "PCMP: Pip Control Message Protocol", Work in Progress.
[8] Pip Router Discovery Protocol, Work in Progress.
[9] Pip Objects Specification, Work in Progress.
[10] Rajagopolan, and P. Francis, "The Multi-Level Path Vector Routing Scheme", Work in Progress.
[11] Francis, P., "Pip Address Conventions", Work in Progress.
[12] Francis, P., "On the Assignment of Provider Rooted Addresses", Work in Progress.
[13] Ballardie, Francis, P., and J. Crowcroft, "Core Based Trees (CBT), An Architecture for Scalable Inter-Domain Multicast Routing", Work in Progress.
[14] Franics, P., "Pip Host Operation", Work in Progress.
[15] Egevang, K., and P. Francis, "The IP Network Address Translator (NAT)", RFC 1631, Cray Communications, NTT, May 1994.

Notes on the References:

As of the publication of this RFC, a version of [12], titled "Comparison of Geographic and Provider-rooted Internet Addressing," was submitted to ISOC INET 94 in Prague. Reference [13] was published at ACM SIGCOMM 93 in San Francisco under the title "An Architecture for Scalable Inter-Domain Multicast Routing".

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