S. Deering (Xerox PARC)
P. Francis (Bellcore)
R. Govindan (Bellcore)
October 1993

Simple Internet Protocol Plus (SIPP): Overview of Routing and Addressing Extensions to SIP

Status of the Memo:

This document is an Internet Draft. Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. Note that other groups may also distribute working documents as Internet Drafts.

Internet Drafts are draft documents valid for a maximum of six months. Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a "working draft" or "work in progress."

Please check the I-D abstract listing contained in each Internet Draft directory to learn the current status of this or any other Internet Draft.

1. INTRODUCTION

The Simple Internet Protocol Plus (SIPP) retains most of the simplicity of SIP [1], while adding some of the features provided by Pip [2]. Specifically, SIPP uses the same syntax as SIP, but the use of the SIP Routing Header (an option similar to the loose source route option of IP) has been enhanced to enable the features provided by the Pip FTIF Chain style of routing.

This document primarily describes the additions to SIP that enable provider selection, mobility, auto-configuration, and variable-length addressing. Thus, this document targets those people who already understand SIP, and wish to understand what additional features come with SIPP. A future document will provide a complete description of SIPP routing and addressing.

The authors would like to acknowledge the contributions of Bob Gilligan (Sun), Bob Hinden (Sun), Christian Huitema (INRIA), and Erik Nordmark (Sun).

1.1. Terminology

The terminology defined in the SIPP Specification [8] applies to this

document. The first three terms below are copied from [8] for clarity. The following additional terminology is defined below that:

address: A 64-bit SIPP layer identifier for a node or set of nodes. An address can be used for both routing and identification purposes.

uniqueness scope: The topologically defined region over which an address may be assigned to no more than one node or set of nodes.

routing scope: The topologically defined region over which hosts and routers have sufficient routing information to forward

a packet toward the node(s) identified by that address.

route sequence: The sequence of addresses consisting of the source address, the destination address, and the addresses encoded in the optional Routing Header of the SIPP packet.

address sequence: A sequence of addresses that forms part of the route sequence. The address sequence is used for the purpose of routing to a node in the case where a single (64-bit) address has insufficient routing scope.

identifying address: The low-order address in an address sequence. Of the addresses in an address sequence, only the identifying address is used to identify the source and destination of a packet (for instance, by the transport protocol).

2. SIPP ADDRESSING

A SIPP address serves two purposes. One is to uniquely identify the node (or set of nodes) to which the address belongs. The other is to specify the location of the addressed node(s) in the internet topology, to facilitate routing.

SIPP addresses are unlike IP addresses (and similar to OSI NSAP [9] addresses) in that they identify nodes (i.e., hosts or routers) rather than interfaces. This having been said, it is possible to assign SIPP address prefixes on a per-interface basis, with the result that packets to a node will tend to arrive over the interface from which the node's address was derived.

A SIPP address is said to have a certain "routing scope", which is the topological region over which nodes have sufficient routing information to deliver a packet to the node(s) identified by that address. Most SIPP addresses have global routing scope, meaning they are routable from anywhere in the internet. However, some addresses have less than global routing scope. For example, during bootstrapping a SIPP node may use an address derived from its link-level address (e.g., an Ethernet address) that is only locally routable.

Every SIPP packet contains two Identifying Addresses (IADs), identifying the source and destination nodes of the packet. The transport-layer pseudo-header checksum for the packet is calculated using the two IADs.

The two IADs may or may not contain sufficient location information to route the packet to its destination(s) (or to route an error

message back to its source). When they are insufficient, an optional SIPP Routing Header is included in the packet to carry the additional addressing information required for routing. These additional addresses may be viewed as high-order extensions of the IADs. The additional addresses and the IAD, taken together, are called an address sequence.

For instance, an address sequence can be used for a mobile node that is attached to a place in the internet other than the location specified by its IAD. Or, an address sequence can be used if the routing scope of the IAD is not sufficient (as may happen if the internet grows too large to assign globally-routable addresses to all nodes). This way, the address sequence can be used to achieve the effect of a variable length address. Even when the address sequence is used to extend the address length beyond 64 bits, however, the identification function uses only the "low order" 64 bits (the IAD).

2.1. Unicast Addresses

The SIPP unicast address or address sequence is contiguous bit-wise maskable, similar to IP addresses under CIDR [3].

The use of the optional Routing Header mechanism as the means of hierarchically extending SIPP addresses puts some constraints on the assignment and use of SIPP unicast addresses.

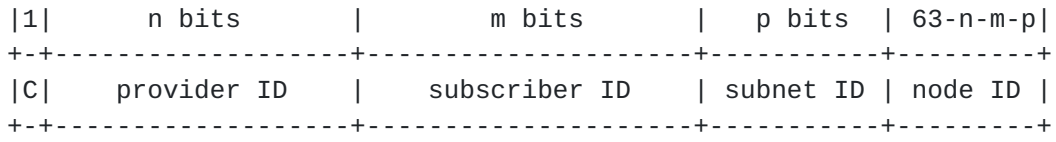
First, since the Routing Header mechanism only allows a route to examine 64 bits at a time, a single "hierarchy element" from a SIPP hierarchical unicast address sequence cannot straddle a 64-bit boundary.

Second, when using a 128-bit address sequence, both the low and high order 64 bits must individually be globally unique. (If the address sequence is greater than 128 bits, the middle 64-bit addresses may not have to be globally unique. In the event that SIPP address sequences grow beyond 128 bits, this possibility can be considered.)

Third, routing must be configured such that, once a given 64-bit segment of an address sequence is examined by a router for the purpose of forwarding a packet, previous (higher order) 64-bit addresses cannot subsequently be used as part of the forwarding decision. In other words, once the Routing Header is advanced to a certain 64-bit address, previous addresses cannot be examined. This places a minor constraint on routing's ability to selectively "look into" other parts of the hierarchy.

2.1.1. Unicast Address Assignment

There are several forms of unicast address assignment in SIPP, including the global hierarchical unicast address, the cluster address, and the local-use address. The global unicast address is initially assigned as follows:



The first bit is the IP compatibility bit, or C-bit [6]. It indicates if the node represented by the address is IP-only or SIPP [8].

SIPP addresses are provider-oriented. That is, the high-order part of the address is assigned to providers, which then assign portions of the address space to subscribers, etc. This is similar to assignment of IP addresses under the CIDR scheme [3]. The provider ID numbers are assigned in such a way that an additional layer of hierarchy can be added above or below the provider ID layer. The node ID corresponds to IP's host number.

Appendix A gives more details of the proposed approach to SIPP global unicast address assignment.

2.1.2. Cluster Addresses

Cluster addresses are a special form of unicast address that allows a packet to be sent to one of multiple destinations (this type of delivery service is sometimes called anycast). Cluster addresses are of the form <prefix><zero>. Cluster addresses allow a packet to be sent to the "nearest" node (according to unicast routing's notion of nearest) of a group of nodes. The purpose of a cluster address is to be able to route a packet to one of a group of nodes characterized by sharing a certain prefix in the unicast global hierarchical address space.

When the packet is being sent from "outside" the group of nodes (that is, being transmitted by a node that has no prefix in common with the cluster address), the resulting behaviour is that the packet is accepted by the first router whose own address prefix matches the prefix in the cluster address. When the packet is being sent from "within" the group of nodes (that is, being transmitted by a node whose own address prefix matches that of the cluster address), the resulting behaviour is that the packet is transmitted up the hierarchy until it reaches a router that is acting "at the hierarchy level" of the prefix.

The SIPP routing and addressing specification will contain a precise and general definition of which nodes get assigned which cluster addresses. However, in the context of the initial format of SIPP hierarchical unicast address, the following cluster addresses are defined:

Provider.0
: Routers that comprise the provider's network (that is, not comprising subscriber networks attached to the provider) identified by the Provider ID are assigned these cluster addresses.

Provider.Subscriber.0
: Routers whose addresses match with Provider.Subscriber, and that share a link with routers that have cluster address Provider.0 are assigned these cluster addresses.

Provider.Subscriber.Subnet.0
: Routers whose addresses match with Provider.Subscriber.Subnet, and that share a link with routers that have cluster address Provider.Subscriber.0 are assigned these cluster addresses.

Packets with address Provider.0 will be accepted by the first router belonging to the indicated provider. This cluster address is used for provider selection and for forming provider-level policy routes.

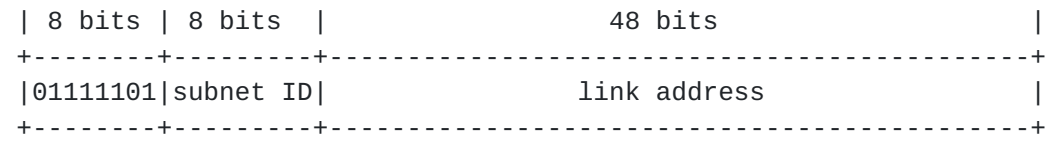
Packets with address Provider.Subscriber.Subnet.0 will be accepted by the first router in the indicated subnet. This cluster address is useful for auto-configuration and for mobile hosts where the scope of mobility is the subnet. If the scope of mobility is the subscriber network, then the cluster address Provider.Subscriber.0 can be used.

The routing and addressing specification will describe how a host learns the various cluster addresses.

2.1.3. Local-Use Address

A SIPP node can form a SIPP address from its own link address (for instance, its Ethernet address). This address is only guaranteed to be unique on the local link (from which the link address was derived). The link address can be used for local communication in a site, for instance for networks that are not connected to the internet, or temporarily for auto-configuration purposes.

Local-Use Addresses have the following format:



If the link address is less than 48 bits, then it is positioned at the low-order portion of the link address field and padded with zeros. If the subnet ID is unknown (for instance, because there is no router on the subnet), then the subnet ID is set to all zeros.

2.2. Other Address Formats

The other address formats described in [8] (Multicast, Unspecified, Loopback, Reserved Multicast, All Nodes, All Hosts, and All Routers Addresses) are as specified in [8].

3. ADDRESS SEQUENCE HANDLING BY NODES

For address sequences to be an effective means of extending SIPP addresses or enriching SIPP routing semantics for use in provider selection, mobility, auto-readdressing, etc., nodes must be able to manipulate address sequences appropriately. A node must be able to recognize that its own addresses and other nodes' addresses may be represented as address sequences, for instance in transmitted and received packets and in DNS. A node must also be able to reverse address sequences for sending return packets.

3.1. Address Sequence Notation

The SIPP mechanism for encoding address sequences is the optional Routing Header. With this mechanism, the active address of the optional Routing Header is in the destination address field of the SIPP header, and the remaining addresses in the address sequence (those that were previously active and those that will be active) are in the Routing Header. Thus, the fields of the Routing Header rotate through the destination address field as each becomes active.

Notated literally, the mechanism would look like this:

	source address	dest address	Routing Header
initial	S	A	>B D
next	S	B	A >D
final	S	D	A B >

This shows a packet from S, routed through A and B on its way to D. The '>' symbol indicates which field is next in the Routing Header

(i.e., the field pointed to by the Next Addr field of the Routing Header). While this notation accurately depicts what happens in the packet header, it is unwieldy, so the following equivalent notation is used instead:

```
  initial     S,*A, B, D
     next     S, A,*B, D
    final     S, A, B,*D
```

In this notation, the first element is in the source address field of the SIPP header. The '*' denotes the active element of the Routing Header, which is in the destination address field. The remaining elements are in the Routing Header, with the Next Addr field pointing to the element after the active one. This notation is easier to read, and not incidentally very similar to the Pip notation, thus illustrating that the original SIP Source Routing Header is semantically similar to the Pip FTIF Chain.

3.2. Node Formation of Address Sequences

This section describes a simple set of rules for the handling of address sequences. These rules allow nodes to form and transmit SIPP packets with traditional IP address semantics. More importantly, they allow nodes to receive and "reverse" SIPP packets with advanced routing and addressing semantics, such as policy routing. Thus nodes that do not understand advanced routing semantics can still operate appropriately when receiving packets from a node that does.

Every node has a set of address sequences that it considers its own. These address sequences are a series of 64-bit addresses of the form <Si, Si-1, Si-2, ..., S0>, where S0 is the low-order address and also the IAD, and Si is the high-order address. Note that the terms low-order and high-order do not necessarily indicate that the high-order terms are hierarchically above the low-order terms. Rather, the implication is that the high-order terms must come first in an address sequence.

> [Note that, for the purpose of allowing simple implementations, an address sequence of three addresses is considered sufficient to encode a node's source address for any reasonable forseeable requirement.]

An address sequence of a node constitutes the sum total of information needed in the packet header to route the packet to that node. Only the low-order address is required to identify the node. Some of a node's address sequences are source-capable and others are not. A source-capable address sequence is one that can validly be used as a "source address" in a transmitted packet. For instance, unicast

address sequences are source-capable and multicast address sequences are not, though both can be considered by the node to be its own address sequences.

A route sequence may contain a number of components, such as a source address sequence, a destination address sequence, a policy route, a mobile host base station, or a multicast tree core address. From the perspective of a "simple" node, however, a route sequence contains only two parts, the source address sequence and the destination address sequence:

<S0, S1, ..., Si, Dj, Dj-1, ..., D0>

For a transmitted packet, the source address sequence is one of the node's own source-capable address sequences, and the destination address sequence is everything else. For a received packet, the destination address sequence is (or at least should be) any of the node's own address sequences, and the source address sequence is everything else.

In an address sequence, the addresses of the source address sequence are ordered with the "low order" parts first, while the addresses of the destination address sequence are ordered in the opposite direction, with the "high-order" parts first. Thus the first and last addresses are always the identifying addresses--the "low order" parts of the source and destination address sequences.

In the common case with a single-component source and destination address, the complete route sequence simply has the form: <S0, D0>. Such packets contain no Routing Header.

When a node has an "association" with another node (that is, a transport connection or an application "connection" running over UDP), it must maintain the following information with respect to the correspondent node (the node with which it has the association):

1. The source and destination IADs for the entire association.

2. The source and destination address sequences currently in use.

The low-order parts of the source and destination address sequences must match the IADs.

When a node initiates an association, it will normally learn the destination address sequence through DNS or from local means (for instance, the user typing it in). It extracts the destination IAD for the association from the low-order part of the destination address sequence. It chooses from among its source-capable address

sequences for the source address sequence, and forms the header as indicated above.

When a node is not the initiator of an association, it must extract the appropriate information from the received packet. A node can isolate its own address sequence from the received route sequence by matching the tail of the route sequence against its own address sequences. (Note that the multicast groups that the node belongs to are included in its list of address sequences for this isolation process.) Once the node isolates its own address sequence from the route sequence, what remains is the address sequence of the correspondent node.

Thus, to return a received packet to the correspondent node, the node:

1. strips off and stores its own address sequence from the tail of the route sequence of the received packet,

2. reverses the order of the remaining elements of the route sequence, and places them on the tail of the route sequence of the returned packet,

3. prepends a valid source-capable address sequence to the route sequence, and

4. sets the active address (that is, the address to which the Next Addr field of the Routing Header points) to be the first address of the destination address sequence.

If the node's own address sequence in the received packet is source-capable, then the transmitted (source) IAD must match that of the received (destination) IAD, and the transmitted address sequence should match that of the received address sequence.

Every node must implement these reversal rules. Even if a node has no notion of, say, provider selection, it will successfully return packets to a node that does if it implements these rules.

3.3. Application Handling of SIPP Addresses

The exact nature of the interface between the SIPP layer and higher layer protocols is still an open issue. At a minimum, an application interface that requires only the source and destination IAD must exist. This allows for a simple interface, but limits the control that the application has over routing.

It should also be possible for an application to control the complete

route sequence if desired. The details of such an interface are under study.

4. ROUTING ALGORITHMS

Initially, SIPP routing algorithms will be virtually identical to those used with the CIDR version of IP, except that the address used will be 64 bits rather than 32. Over the near term, cluster addresses can be configured into routers along with their native addresses, and advertised in the normal way. Eventually it would be desirable to have routers automatically determine their own cluster addresses.

If it ever becomes necessary to extend SIPP addresses beyond 64 bits, then the routing algorithms can be modified if necessary to reflect that change. (Note that it is not clear that routing algorithms would have to be modified for this.)

5. UNICAST EXAMPLES

In this section, we give several typical unicast examples that demonstrate the use of the Routing Header mechanism for provider selection and address extension. Later sections give typical examples for mobility, multicast, and auto-configuration. A forthcoming full specification of SIPP routing and addressing will describe the use of the Routing Header mechanism more thoroughly.

The examples assume the following topology. Domain D contains node H. Domain E contains node I. Domain D is attached to providers P and Q. Domain E is attached to providers Q and R.

5.1. Simple (Non-Extended) Addresses

Assume that the addresses of node H are P.D.H and Q.D.H, and the addresses of node I are Q.E.I and R.E.I, where the notation "a.b.c" represents a 64-bit SIPP address. (Note that the 'D' of P.D.H and Q.D.H are subscriber numbers assigned by P and Q respectively, and are therefore in general not the same value.) H initiates an association with I by querying DNS and learning the two addresses for I. Assume that H chooses Q.E.I, since it has the best "match" with its own addresses. H forms the following packet:

Route sequence at sender H: Q.D.H, *Q.E.I

I receives this packet, reverses the (in this case simple) route sequence, and returns:

Reversed route sequence at receiver I: Q.E.I, *Q.D.H

5.2. Simple Addresses with Provider Selection

The previous example is in fact a form of provider selection, but it is simple in that both ends have the same provider, so nothing explicit has to be done. Assume in this case, however, that H wishes to send its packet through provider P. Since I is not attached to provider P, H must explicitly indicate that it wants its packet to go through provider P, and therefore forms the following packet:

Route sequence at sender H: P.D.H, *P.0, Q.E.I

The active element of the route sequence is the cluster address of provider P. This causes routers in domain D to route the packet to provider P. When the first router in provider P receives this packet, it recognizes the packet as being for "itself", and advances the Routing Header, producing:

Advanced route sequence at provider P router: P.D.H, P.0, *Q.E.I

which causes the packet to be routed to I. Upon receiving this packet, I uses the reversing rules stated above, producing the following packet:

Reversed route sequence at receiver I: Q.E.I, *P.0, P.D.H

This packet has a couple of noteworthy characteristics. First, the packet may exit domain E via either provider Q or R, depending on what routing considers the best path to provider P. Second, the P.0 element is redundant, since the destination address P.D.H will also cause the packet to be routed to P. However, without knowing the provider mask of P, I has know way of knowing that P.0 is redundant with P.D.H, and so includes both elements. In the future, it may be possible for I to learn H's cluster address and optimize the header accordingly.

To continue this example, assume that I does care which exit provider is used to reach H, and further that I chooses provider Q. In this case, I would insert the following "policy route" (actually, one address) to force the packet to go through Q outgoing:

Alternative reversed route sequence: Q.E.I, *Q.0, P.0, P.D.H

Note that this example describes a node that is more sophisticated than the simple one described previously. In particular, the node in this example understands three components of the source route (the source and destination addresses and a provider) rather than just two (the source and destination addresses). The node further understands that when it inserts the provider selector in the route sequence, it

sets the active element to be that of the provider selector on transmitted packets.

5.3. Extended Address (Address Sequence)

Now assume that at some point 64 bits become inadequate and addresses are extended to an address sequence of two 64-bit addresses. In this case, H's address sequences are P.D:S.H and Q.D:S.H, where the colon ':' indicates a 64-bit boundary, and S represents a subnet inside domain D. I's address sequences are Q.E:S.I and R.E:S.I.

For H to send a packet to I, it could form:

Route sequence at sender H: S.H, Q.D, *Q.E, S.I

The active element Q.E would cause the packet to be routed to domain E, where the Routing Header would be advanced to:

Advanced route sequence at router in Domain E: S.H, Q.D, Q.E, *S.I

and delivered to I.

The above reversing rules executed by I would produce:

Reversed route sequence at receiver I: S.I, Q.E, *Q.D, S.H

thus returning the packet to I.

6. MULTICASTING IN SIPP

This section describes how traditional multicast [5] works using SIPP route sequences. As with unicast, SIPP multicast address sequences are described using a series of 64-bit address elements. Thus, the node's notion of multicast addressing is extended such that a "multi-cast address" is seen as an address sequence rather than a single multicast address as is the case with IP. The final element of the multicast address sequence is the group ID.

When a node joins a multicast group, it considers the multicast address sequence to be one of its own address sequences for the sake of packet reception and reversal. The multicast address sequence is not source-capable.

In traditional multicast (also known as Deering multicast or source-based multicast), a packet from a sender to a multicast group is sent on the shortest-path delivery tree (rooted at the sender) to members

of the group. The traditional SIPP multicast address sequence contains only one address--the group ID. This section describes the Routing Header that is formed by the sender, the reversed Routing Header formed by the receiver and the necessary extensions to the multicast forwarding algorithm.

6.0.1. Example

Assume the same scenario as described above (with nodes H and I), further assuming that H and I have extended addresses as described above. (We do an extended address example here because the simple address example is the same as with current IP.) Assume further that H is transmitting a traditional multicast with multicast address M, and that I is a member of group M. H forms the following header:

Route sequence at sender H: S.H, Q.D, *M

The route sequence is received unchanged at each of the receivers.

If I wishes to respond unicast to H, it executes the reversing rules described above in the following way. First, it isolates its own address from the route sequence. Since this is multicast, and since I is a member of the multicast group M, I considers M to be one of its "own" addresses, and strips it. Reversing what remains produces <Q.D, P.H>. Since a multicast address cannot be used as a source address, I knows to prepend its own unicast address sequence to the route sequence, producing:

Reversed route sequence at receiver I: S.I, Q.E, *Q.D, S.H

6.0.2. Multicast Forwarding Extensions

With traditional multicast, each router's next hops are dependent both on the multicast group address and the sender address. When the sender's address is an address sequence next hop determination is potentially influenced by all of the addresses in the sequence.

In the header formed by the sender, the SIPP Routing Header part contains all but the low-order address. Therefore, each multicast router will need to parse SIPP options for every packet containing a multicast destination address. For instance, in the above example, the routers would need to look at the destination address to determine that it is multicast, then look in the Routing Header at "Q.D", then if necessary (for instance, because the packet is still inside domain D) look at "S.H" in the source address field.

While this represents additional overhead, routers will not need to

incur this "peek-behind" overhead until 64-bit addresses are insufficient for global routability of Internet nodes.

7. MOBILITY IN SIPP

This section shows how SIPP source routing and SIPP route sequences might be used for mobile communication. Note that the Mobile IP group of IETF is deliberating on various solutions for mobility, and may choose a different approach than the one outlined here. At the same time, more approaches are possible with SIPP than with IP, so the Mobile IP group may choose a different solution for use with SIPP. First, we introduce some terminology.

Mobile Host (MH)
A node that is able to maintain its network connections even after being physically moved.

Correspondent Host (CH)
A node that has a network connection open to an MH. A CH may itself be mobile.

Base Station (BS)
The SIPP router to which the MH is attached after it moves.

Home Station (HS)
A SIPP node that is aware of the MH's current location. Each MH has a preconfigured home station.

7.1. Mobility Example

Assume that H is an MH and that I is the CH, both with the (extended) address sequences from above. Initially, a packet from the CH to the MH carries the following route sequence as in the above example:

Route sequence from CH to MH: S.I, Q.E, *Q.D, S.H

Now suppose the MH moves to the vicinity of a BS with an address D.d. MH discovers D.d through SIPP "I-Am-Here" advertisements. These are multicast by the BS to the SIPP All Nodes address (similar to an IS hello advertisement in ES-IS). MHs also periodically multicast SIPP "I-Am-Here" advertisements to the SIPP All Routers multicast address (similar to the ES hello advertisement in ES-IS). This latter advertisement contains the MH's identifying address.

Through a mechanism beyond the scope of this document, the MH informs the HS of its new base station. Packets carrying the old route sequence from the CH are intercepted by the HS. The HS tunnels them

to the BS, which forwards it to the MH.

After the MH has discovered D.d, all subsequent packets to the CH carry the following route sequence:

> Route sequence from MH to CH after move: S.H, D.d, *Q.E, S.I

It is sufficient to include only MH's identifying address in the route sequence; we assume that the BS is within I's IADs (S.I) routing scope. When the CH reverses the incoming route sequence from the MH, it created the following route sequence:

> Reversed route sequence from CH to MH after move: S.I, Q.E, *D.d, S.H

This causes packet to the MH to be routed through the BS at D.d, producing the desired behavior.

8. HOST AUTO-CONFIGURATION

This section describes how a host constructs a temporary IAD when it boots and uses that to contact DNS or some configuration server to obtain host configuration information. We are interested in the four scenarios comprised of the combinations whereby 1) either a router is or is not on the host's local link, and 2) either the host can or cannot contact a configuration server.

When a host boots up, it assigns itself a temporary local-use IAD formed from its link address. This IAD is routable only within the link on which the host is located.

The host then sends out a small number of SIPP "Where-Are-You" solicitations with the local-use IAD as the source address. If it receives no advertisements, the host assumes that there is no router on the link and uses the temporary IAD to communicate with other nodes on the link. If, using this IAD, the host is able to contact a configuration server on the link, then it may obtain a different, possibly global, address at this time.

Once it hears a router advertisement, a host can use one of the advertised prefixes to form an address with greater routing scope. This address can be used to contact the configuration server. (The address of the configuration server could be a well-known multicast address.) If any of the prefixes advertised by the router is small enough to accommodate the host's link address (e.g., in the case of a multi-link domain not connected to the Internet), the host can concatenate that prefix with its link address to form its address. Otherwise, the host can assign itself the address sequence <C, T>, where

T is the host's local-use IAD, and C is the cluster address of the subnet learned from the router advertisement.

If a configuration server responds with a new address, the host uses that address. Otherwise, the host continues to use the previous address to communicate with other nodes (either in its domain or globally). Note that the design of a configuration server is an open issue.

References

[1] S. Deering, "Simple Internet Protocol (SIP) Specification", Internet Draft, work in progress.

[2] P. Francis, "Pip Near-term Architecture", Internet Draft, work in progress.

[3] V. Fuller, T. Li, K. Varadhan, J. Yu, "Supernetting: an Address Assignment and Aggregation Strategy", RFC 1338.

[4] P. Tsuchiya, "On the Assignment of Subnet Numbers", RFC 1219.

[5] S. Deering, "Host Extensions for IP multicasting", RFC 1112.

[6] R. Gilligan et al, "SIPP Transition Mechinisms", Internet Draft, work in progress.

[7] P. Francis, "On the Assignment of Provider Rooted Addresses", Internet Draft, work in progress.

[8] S. Deering, "Simple Internet Protocol Plus (SIPP) Specification", Internet Draft, work in progress.

[9] R. Colellea, E. Gardner, R. Callon, "Guidelines for OSI NSAP Allocation in the Internet", RFC1237.

APPENDIX A: PROPOSED UNICAST GLOBAL SIPP ADDRESS ASSIGNMENT

This appendix proposes an initial assignment scheme for SIPP global hierarchical unicast addresses that has the following characteristics:

1. it accommodates existing IP addresses,

2. it is an extension of the CIDR address assignment scheme,

3. it leaves address space open for several avenues of future growth.

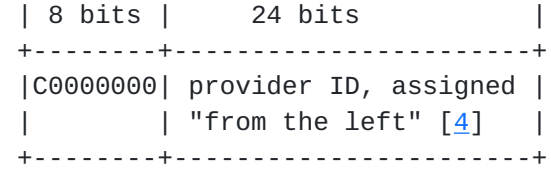
The initial assignment scheme for SIPP unicast addresses is provider-based, as follows:

C-bit

The left-most bit is the IPv4 compatibility flag [6], known as the C-bit. Both SIPP and IPv4 hosts can be assigned SIPP addresses in the SIPP unicast address space. Even though IPv4 hosts may be listed in the DNS with 64-bit SIPP addresses, they only "know" the low-order 32-bit part of their address. The C-bit is used by SIPP nodes to differentiate SIPP systems from IPv4 systems. SIPP systems are always assigned addresses with the C-bit set to 0. IPv4 systems are always given addresses with the C-bit set to 1.

Provider Assignments

Initially, the provider ID will be 31 bits in length. The provider mask is 32 bits in length (covering the provider ID and the C-bit). Provider IDs initially have the following format:



The leftmost 7 bits of the provider ID (not including the C-bit) are assigned as 0. The subsequent 24 bits are assigned values according to the technique of assigning IP subnet numbers outlined in RFC 1219 [4]. That is, the "1" bits of the values are filled in from left-to-right rather than from right-to-left. This style of assignment reserves 0's on the right side of the provider ID, thus allowing the mask of a given provider to shrink in the future, if it is found that the provider needs more bits, for instance to identify its subscribers.

For example, initial provider ID assignment would proceed as follows:

```
      first provider:  C0000000 10000000 00000000 00000000
     second provider:  C0000000 01000000 00000000 00000000
      third provider:  C0000000 11000000 00000000 00000000
     fourth provider:  C0000000 00100000 00000000 00000000
      fifth provider:  C0000000 10100000 00000000 00000000
                       ....
      tenth provider:  C0000000 01010000 00000000 00000000
```

This initial assignment of the provider ID space (zeros to both the left and right of the assigned bits) allows for several avenues of growth.

For instance, if internet growth results in a small number of very large providers, then the masks of the providers can be shrunk, thus giving each provider more space, which could be used to add another level of hierarchy within the provider. If providers grew so large that they required even more space, they could be allocated codes in the most significant byte of the provider ID space.

The reserved space in the most significant byte could also be used for different kinds of number assignments, such as geographical or organizational, if that becomes desirable in the future. (Strictly speaking it wouldn't be necessary for such different number assignments to have their own contiguous number spaces. For instance, the geographical codes could come from a portion of the provider space. However, having separate contiguous number spaces for different types of addresses simplifies address administration within each space.)

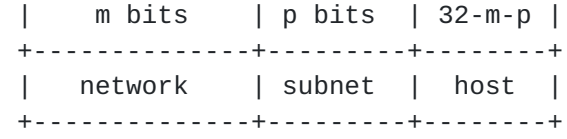
If internet growth results in a large number of small providers, then it might be necessary for the zero bits in the most significant byte to be used as an additional layer of hierarchy above the provider level.

Assignment of Lower 32 Bits

The lower 32-bits of the SIPP address is initially nothing more than

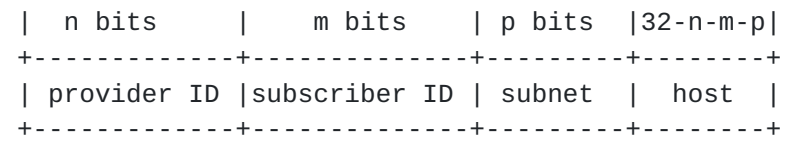
the current IP address. After the IP address space expires (is no longer globally unique), then the lower 32 bits of the SIPP address will no longer by itself be globally unique, and will be assigned by the addressing authority identified by the higher 32 bits of the SIPP address (rather than being assigned by the current IP top-level address assignment authority).

IP addresses currently exist under two formats, class-based (pre-CIDR) and class-less (CIDR) [3]. Pre-CIDR assignments have the following format:

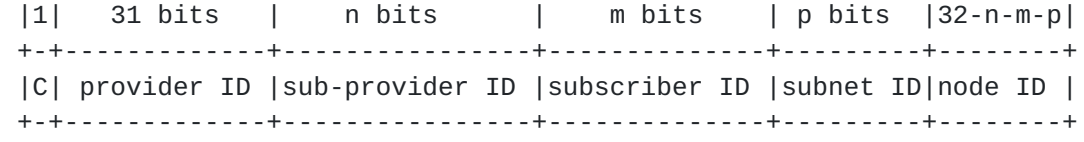


The IP network number corresponds to the SIPP subscriber ID. The SIPP subnet ID and node ID correspond to the IP subnet and host numbers respectively. The network number is globally unique among network numbers. Thus, providers have no control over the assignment of subscriber IDs derived from pre-CIDR IP addresses.

CIDR assignments have the following format:



Under CIDR, providers have control of subscriber assignments. After IP addresses are no longer unique, providers will also have control over the top n bits of the "IP address" (the lower 32 bits). These bits can be used either to assign directly to more subscribers, or to create a level of hierarchy above the subscriber level, resulting in:



As mentioned above, it will also be possible for the sub-provider ID to grow into the provider ID space, by shrinking the provider mask for the provider. In general, as the internet grows, the structure of the SIPP global address may evolve to accommodate the growth. In the extreme case, the SIPP global address can expand to greater than 64 bits.

Note that the use of provider-based addresses results in multiple

address prefixes for subscriber domains that are attached to multiple providers [7]. While this has the advantage of giving nodes a provider selection feature, it has the disadvantage of added complexity in nodes, DNS, and address administration.