# Flexible Routing and Addressing For a Next Generation IP

*Paul Francis*
NTT Software Laboratories
Tokyo, Japan
francis@slab.ntt.jp

*Ramesh Govindan*
Bell Communications Research
Morristown, NJ, USA
rxg@thumper.bellcore.com

## Abstract

Due to a limited address space and poor scaling of backbone routing information, the Internet Protocol (IP) is rapidly reaching the end of its useful lifetime. The Simple Internet Protocol Plus (SIPP), a proposed next generation Internet Protocol, solves these problems with larger internet layer addresses. In addition, SIPP provides a number of advanced routing and addressing capabilities including mobility, extended (variable-length) addressing, provider selection, and certain forms of multicast. These capabilities are all achieved through a single mechanism, a generalization of the IP loose source route. We argue that, for reasons of simplicity and evolvability, a single powerful mechanism to achieve a wide range of routing and addressing functions is preferable to having multiple specific mechanisms, one for each function.

## 1 Introduction

The Internet Protocol (IP, [18]) is rapidly reaching the end of its useful life as a global internetwork protocol. Fundamentally, the IP address space is too small to indefinitely satisfy current Internet growth rates [10]. It is difficult to accurately predict when IP addresses will run out, but most estimates range from 5 to 10 years.

A related problem with IP is poor scaling of internet routing. IP routing has historically scaled proportional to the number of IP networks, which is on the order of the number of organizations using IP. Classless Inter-Domain Routing (CIDR) [9], a technique which proposes changes in the definition and allocation of IP addresses, may alleviate this problem in the short term. With CIDR, scaling in routing can be nearly proportional to the number of IP service providers (also called IP backbones). Some of the scaling benefits are lost if many organizations choose to change service providers without renumbering their networks to reflect provider attachments, or many organizations are each connected to more than one provider.

To find a longer term solution to these problems, the IETF (Internet Engineering Task Force—the standards body with oversight of TCP/IP and related protocols) has initiated a standardization effort for the *next generation* IP (IPng), a replacement protocol for IP. IPng is required to solve the address depletion and scaling problems of IP. In addition, IPng may add new features to IP.

A number of protocols (Pip [7], SIP [4], and TUBA [15]) have been proposed for IPng. All of the proposals solve the basic problems of scaling and address depletion by using larger internet layer addresses. A recent proposal, SIPP (the Simple Internet Protocol Plus), uses SIP's syntax but borrows some of Pip's advanced routing and addressing features.

SIPP's advanced routing and addressing mechanism is a generalization of IP's loose source route. This mechanism enables such features as variable-length addressing, mobility, automatic host address configuration, and provider selection. Moreover, the mechanism is designed to be flexible and evolvable, allowing new routing and addressing features to be incorporated relatively easily into the Internet.

## 2 Overview

The mechanism for obtaining advanced routing and addressing features from SIPP is a generalization of the loose source route mechanism of IP. In SIPP, the analogous mechanism is called the *routing header*. The SIPP routing header encodes a list of SIPP addresses in the packet header (see Section 7). This list of addresses is called a *route sequence*. Each address in this list identifies an intermediate destination (a *target*) on the path to the final destination. The intermediate destination to be visited next is called the *active* target. In an IP loose source route, targets are individual *nodes* (hosts or routers). In SIPP, targets can be individual nodes, or any node in a cluster of nodes.

When it receives a packet, a node compares the active target in the route sequence against a set of addresses it has stored. This is the set of addresses for which the node is a target. If the comparison succeeds, the next target in the route sequence becomes the active target. This way, the packet visits each target listed in the route sequence.

At first glance, the number of uses for this mechanism appears to be rather limited. Indeed, IP hosts use the loose source route mechanism very infrequently. In actual fact, however, virtually all known routing mechanisms can be modeled as loose source routes [8]. For instance, routing on hierarchical addresses—the most common form of routing in the Internet—can be modeled as a loose source route, as we now show.

An IP hierarchical address has three levels; network, subnet, and host. Conceptually, when a packet is routed to an IP address, routers

[0]This work was partially funded by the Advanced Research Projects Agency (ARPA) under contract No. DABT63-93-C-0042.


SIGCOMM 94 -8/94 London England UK
© 1994 ACM 0-89791-682-4/94/0008..$3.50

initially examine the network number. When the packet reaches a router within the addressed network, that router forwards the packet towards the addressed subnet. Finally, when the packet reaches a router on the subnet, that router examines the host number and routes the packet to the addressed host. In loose source routing terms, this packet can be said to have traversed two intermediate targets, the network and subnet, on its way to the destination host. There are differences between processing a source route and a hierarchical address. For example, when processing a source route, the active target is explicitly labeled, and it is not necessary to examine previous (already visited) targets; with a hierarchical address, all previous targets (hierarchical numbers) must be examined by each router. Such differences, however, are *mechanistic* not *semantic*. It is possible to obtain the semantics of a hierarchical address using the mechanics of a source route. This approach, in its full generality, was first used in Pip [7].

In this paper, we show how the generalized source route mechanism is used in SIPP to achieve *flexible* routing and addressing features. In particular, we illustrate how mobile routing, variable-length addressing, provider selection, and certain forms of multicast routing may be modeled using this mechanism. For these features to work correctly, every SIPP node should be able to reverse the order of addresses in a received route sequence, and use the resulting route sequence to return the packet to the originator. We also show how this ability of SIPP nodes to reverse route sequences also enables an incremental *evolution* of the internet towards new routing and addressing paradigms that can be modeled using route sequences.

The remainder of the paper is organized as follows. Section 3 describes the format and assignment of SIPP addresses. Section 4 discusses the rules for route sequence reversal by nodes. The use of route sequences to model advanced routing and addressing paradigms is illustrated in Section 5. Section 6 describes how the use of route sequences impacts routing algorithms and address assignment. Section 7 lists performance, flexibility and security implications of the use of route sequences. Section 8 discusses related work and Section 9 summarizes the contributions of this paper.

# 3 Addresses and Address Sequences

A route sequence is a list of SIPP *addresses*. Like the IP address, a SIPP address serves two purposes; 1) it identifies a node or a group of nodes, and 2) it may specify a node's network location to aid the routing function.

Abstractly, a route sequence in a SIPP packet encodes the location of the source and destination in the internet (among other information). We allow the location of a SIPP node to be specified by an *address sequence* which is an ordered list of one or more 64-bit SIPP addresses.

This section describes SIPP addresses and address sequences in greater detail.

## 3.1 SIPP Address Types

Two types of packet delivery services are supported in SIPP. With *multicast* service, a packet is (unreliably) delivered to all of the nodes identified by the address. With *unicast* service, a packet is (unreliably) delivered to exactly one node. The destination address in a SIPP packet determines the type of delivery service given to the packet. Thus, SIPP addresses fall into two classes: unicast and multicast. Multicast and unicast addresses are assigned from the same 64-bit address space, but are distinguished by multicast addresses having a fixed prefix. A SIPP address' identification function does not necessarily imply a packet delivery service; for example, a unicast address can identify a set of nodes (see below).

This section describes unicast and multicast SIPP addresses. As of this writing, SIPP is in the early stages of standardization, so address formats and assignments could change. The general principles described here, however, are expected to remain the same.

Apart from a fixed prefix, a SIPP multicast address has three fixed length fields: *flags*, *scope* and *group ID*. The *flags* field is used to distinguish between permanently assigned (or well-known) multicast groups and transient multicast groups. The *scope* field is used to delimit the topological region within which delivery of the multicast packet is confined; possible values of this field denote "within subnet", "within private network", and so on. Finally, the *group ID* is a numerical identifier for a multicast group.

A 64-bit SIPP unicast address also consists of different fields. SIPP unicast addresses are contiguous bit-wise maskable; that is, field are not fixed length and not necessarily byte aligned, but fields representing topologically related entities are contiguously assigned. The encoding of fields within an address depends on SIPP unicast address assignments; these include the global hierarchical unicast address, and the cluster address. The following subsections describe these in greater detail.

### Hierarchical Unicast Addresses

Initially, each SIPP node is assigned one or more globally hierarchical 64-bit SIPP addresses. The hierarchical SIPP address is divided into a number of fields, each of which denotes one level of the routing hierarchy. The larger size of the address allows encoding more levels of hierarchy than that permitted by an IP address.

A SIPP hierarchical unicast address is *provider-oriented* (Figure 1(a)). That is, the highest-order field (the *provider ID*) within the address is assigned to Internet service providers, with each provider being assigned different values for the field. This is similar to the assignment of IP addresses under CIDR. Each provider then assigns different *subscriber ID* values to different directly attached subscribers. For given provider and subscriber IDs, the *subnet ID* identifies a topologically connected group of nodes within the subscriber network. The group of nodes identified by the subnet ID may all be attached to the same link, or may be spread among multiple links. Finally, for given provider, subscriber and subnet IDs, the *node ID* identifies a single node among the group of nodes in the subnet. The term *subscriber prefix* refers to the high-order part of the address up to and including the subscriber ID field. Provider and subnet prefixes are similarly defined.

The internal structure of a SIPP hierarchical unicast address cannot be gleaned by examination of the address alone (unlike the pre-CIDR IP network number, which could be determined by examining the IP address). SIPP hosts and routers may have varying degrees of knowledge of the internal structure of the SIPP address.

At a minimum, a host may have no knowledge of the internal structure of any SIPP address, including its own. A more sophisticated host may additionally be aware of the subnet prefixes for the link(s) it is attached to. Still more sophisticated hosts may be aware of their subscriber and provider prefixes, for uses described

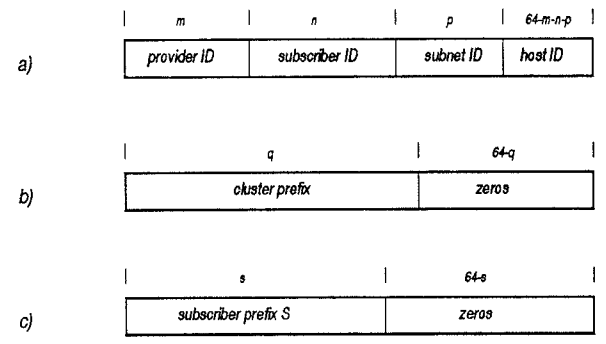


Figure 1: *SIPP Unicast Addresses:* Initially, the globally hierarchical unicast address is assigned as shown in (a). (b) shows the generic form of the SIPP cluster address. (c) shows a subscriber cluster address.

in Section 5. A host can either be manually configured with these prefixes or discover them through configuration protocols.

Though a very simple router may have no knowledge of the internal structure of SIPP unicast addresses, routers will more generally have knowledge of one or more of the hierarchical boundaries for the operation of routing protocols. Which boundaries a router knows depends on what positions it holds in the routing hierarchy.

### Cluster Addresses

A *cluster* is a set of nodes identified by a common prefix $C$ in the SIPP unicast routing hierarchy. Each node in such a cluster has at least one unicast address with prefix $C$. A *boundary router* of this cluster has at least one link to a node outside the cluster.

A *cluster address* is a unicast addresses that may be used to reach the "nearest" one (according to unicast routing's notion of nearest) of the set of boundary routers of a cluster. Cluster addresses have the general form shown in Figure 1(b). Thus, to reach the nearest boundary router for the routing domain identified by subscriber prefix S, a packet may be sent to the cluster address shown in Figure 1(c).

Cluster addresses are most commonly used as intermediate addresses in a SIPP route sequence (Section 5), to cause a packet to be routed through one or more specific clusters on the way to its final destination.

## 3.2 Address Sequences

A SIPP address uniquely identifies the node (or set of nodes) to which the address belongs. We call such a SIPP address the node's (or set of nodes') *identifying address*. A node's identifying address may be used by transport protocols for endpoint identification and pseudo-header checksumming.

A SIPP address also specifies the location of the addressed node(s) in the internet topology, to facilitate routing. Each SIPP address is said to have a certain *routability 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.

A node's identifying address may not contain sufficient location information to route the packet to its destination(s). In such a case, its routability scope is enlarged by prepending additional addresses to the identifying address. These additional addresses, together with the identifying address, form an *address sequence*.

More generally, the current location of a SIPP node (or a set of nodes) in the internet is completely specified by one or more address sequences. Our notation for an address sequence is $\langle A_n, A_{n-1}, \ldots, A_1, A_0 \rangle$, where each $A_i$ is a SIPP address and $A_0$ is an identifying address for the SIPP node. A node's address sequences may change over time, for instance, if the node's location in the internet changes.

In the simplest case, a node's location in the internet may be completely specified by a one-element address sequence containing its global hierarchical unicast address. Similarly, the location of a group of nodes may be completely specified by a one-element address sequence containing the SIPP multicast address for that group.

An address sequence may also be used to specify the location of a mobile host $M$. Suppose $M$ has moved away from the location specified by its identifying address $A$ and is now in the vicinity of a router $R$ whose subnet cluster address is $B$. The address sequence $\langle B, A \rangle$ now completely specifies the location of $M$ in the internet. In this address sequence, $A$ only uniquely identifies the node and does not provide any location information.

Address sequences can also be used if the routability scope of the identifying address is not sufficient (as may happen if the internet grows too large to fit globally-routable addresses into 64-bits). 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 "lowest-order" address in the address sequence must globally uniquely identify the node. The implications of modeling extended addresses using address sequences on host and router software are described in Section 7.

# 4 Route Sequences

The *route sequence* in a SIPP packet (the list of target addresses in the routing header of a SIPP packet) includes the address sequences of the source and destination. A node must be able to recognize that its own location and other nodes' locations may be represented as address sequences in transmitted and received packets. It must also correctly formulate route sequences in outgoing packets and reverse route sequences in incoming packets. This section describes the need and the rules for correct formation and reversal of route sequences by nodes.

## 4.1 Motivation For Route Sequence Reversal

To achieve traditional unicast IP address semantics, it is not necessary for IP implementations to correctly reverse IP loose source routes (in fact, there exist IP host implementations that do not reverse source routes, [13]). For example, when an IP host receives a packet with a loose source route, it may simply use the sender's IP address as the destination for outgoing packets and completely discard everything else in the source route.

Such host behavior is incorrect when the source route mechanism

is used to model more advanced routing and addressing semantics. In Section 3.2, we discussed how higher-order addresses may be prepended to a node's identifying address to enable mobile routing or extended addressing. A SIPP node cannot discard addresses from the route sequence in incoming packets without violating the semantics of these address sequences. For this reason, all nodes are required to correctly implement route sequence formation and reversal.

A benefit of this requirement is that a node that does not understand advanced routing semantics can still operate correctly when receiving packets from a node that does. Thus, new routing and addressing semantics that can be modeled with the source route mechanism can be incrementally deployed in the internet.

## 4.2 Node Formation of Route Sequences

SIPP route sequences contain the address sequences of the source and destination as well as other SIPP addresses used to effect advanced routing and addressing semantics (e.g. provider cluster addresses for policy routing, Section 5). This section describes how these components are encoded in a route sequence.

Each SIPP node $N$ is represented by a set of address sequences $\mathcal{Q}$. The destination address sequence in the route sequence in any packet destined for $N$ must belong to $\mathcal{Q}$. Thus, $\mathcal{Q}$ contains not only the unicast address sequences for $N$ but the addresses of all multicast groups that $N$ currently belongs to. Some of the address sequences in $\mathcal{Q}$ are *source-capable*. Only source-capable address sequences can validly be used as the source address sequence in a route sequence in packets sent by $N$. For instance, while both unicast and multicast address sequences can belong to $\mathcal{Q}$, unicast address sequences are source-capable and multicast address sequences are not.

Suppose, in general, that the source of a packet is represented by the address sequence $\langle S_n,\ S_{n-1}, \ldots,\ S_1,\ S_0\rangle$ and the destination by the address sequence $\langle D_m,\ D_{m-1}, \ldots,\ D_1,\ D_0\rangle$. A simple route sequence in packets from the source to the destination would look like:

$$\langle S_0,\ S_1, \ldots,\ S_n,\ \star D_m,\ D_{m-1}, \ldots,\ D_0\rangle$$

That is, 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. The identifying addresses $S_0$ and $D_0$ are always at opposite ends of the route sequence. The *active target* in the route sequence, denoted by a "$\star$" immediately preceding it, is set to the SIPP address immediately following $S_n$, the highest-order SIPP address in the source address sequence.

The above route sequence is encoded in SIPP headers [5] as follows (Figure 2). $S_0$ and $D_m$ are placed in the source and destination address fields of the SIPP header. The rest of the route sequence is placed, in order, in the SIPP Routing Header (the Routing Header is similar to the IP Loose Source Route option but has larger addresses). A *next address* field in the Routing Header points to the address immediately following the route sequence's active target. Advancing the active target of a route sequence corresponds to swapping the destination address and the address pointed to by the next address field, and incrementing the next address field by one.

When a node initiates an *association* (e.g., a transport connection) with another node, it learns the destination address sequence through a Domain Name System (an Internet name to address translation service) request or by other means (e.g., user input). It then chooses a source address sequence from among its source-capable address sequences, and forms the route sequence indicated above[1]. The node may optionally insert one or more SIPP addresses (typically provider cluster addresses) before the destination address sequence in the route sequence.

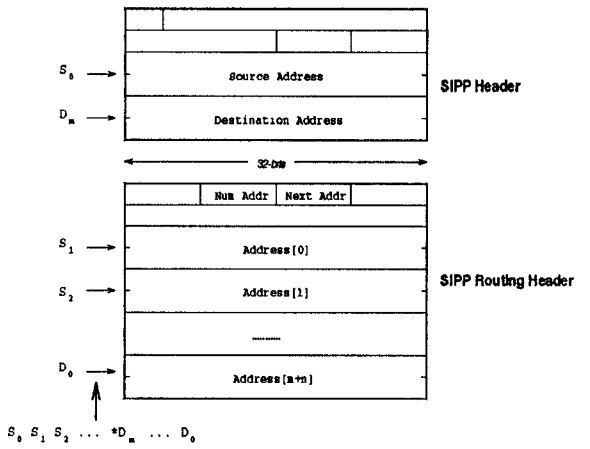


Figure 2: *SIPP Headers:* Every SIPP packet carries a SIPP packet header. Optional internet-layer information in SIPP is encoded in separate headers. The SIPP routing header is one such optional header. The first address and the active target of a route sequence are placed in the SIPP header's source and destination fields. The remaining addresses are placed in the SIPP routing header.

When a node does not initiate an association, it must extract the remote end's destination address sequence by reversing the route sequence in an incoming packet. Two cases arise: 1) where a node is the destination of the packet, and 2) where the node is a router that has encountered an error in processing a packet and must return an error message. We describe the rules for these below.

## 4.3 Destination Node Reversal of Route Sequence

Suppose node $N$ receives a packet with the route sequence $\langle R_0,\ R_1, \ldots,\ R_{k-1},\ R_k\rangle$. $N$ compares each element in $\mathcal{Q}$ against the tail of the received route sequence, looking for the best match. A best match address sequence in $\mathcal{Q}$, $\langle S_n,\ S_{n-1}, \ldots,\ S_1,\ S_0\rangle$, has the largest $i$ such that $S_0 = R_k,\ S_1 = R_{k-1}, \ldots,\ S_i = R_{k-i}$. At a minimum, one of the node's identifying addresses must match the destination identifying address from the received route sequence,

[1] The use of address sequences in SIPP to represent the location source and destination impacts the the service interface offered by SIPP to transport protocols. In IP, when sending a packet the transport layer specifies the 32-bit IP addresses of the source and destination. In SIPP, however, the transport layer must specify the complete address sequences of the source and destination. Similarly, after processing a received packet (see Section 4.3), the SIPP layer passes up to the transport layer the source and destination address sequences in the incoming packet.

$R_k$.

The node $N$ then reverses the remaining (unmatched) addresses in the incoming route sequence, to get

$$\langle R_{k-i-1},\ R_{k-i-2},\ \ldots,\ R_1,\ R_0 \rangle.$$

As far as $N$ is concerned, this is a valid address sequence representing the destination (actually, this may contain the destination address sequence prepended with some additional SIPP addresses, representing, for example, a policy route). Using this, $N$ forms the route sequence for outgoing packets as described in Section 4.2. If the best match source address sequence is also source-capable, then that is used as the source address sequence in the route sequence. $R_{k-i-1}$ is set to be the active target in the outgoing route sequence.

## 4.4 Intermediate Node Reversal of Route Sequence

We now consider the situation where a router has detected an error in processing a packet and needs to send a error message to the packet's source. Let the route sequence in the packet causing the error be $\langle R_0,\ R_1, \ldots,\ R_{k-1},\ R_k \rangle$. Let $R_j$ be the active target in this route sequence. By our formation rules, note that $j \geq 1$.

The route sequence in the outgoing error message must be

$$\langle r_0,\ r_1, \ldots,\ r_l,\ \star R_{j-1},\ R_{j-2}, \ldots,\ R_0 \rangle,$$

where $\langle r_l,\ r_{l-1}, \ldots,\ r_0 \rangle$ is any source-capable address sequence for the router generating the ICMP error message. The active element in this route sequence is $R_{j-1}$. Intuitively, the "consumed" portion of the invoking packet's route sequence is used to route the error message back to the source.

# 5 Using Route Sequences for Advanced Routing and Addressing

This section illustrates the use of route sequences to achieve a variety of unicast and multicast routing and addressing semantics. The following subsections describe these examples in detail.

## 5.1 Unicast with Route Sequences

A variety of unicast routing and addressing features including traditional IP semantics, extended addresses, and exit provider selection are desirable in an IPng. This section shows how SIPP achieves these through route sequences and correct route sequence reversal by nodes.

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

### Non-Extended Addresses

In this example, node $H$ initiates an association with node $I$. The subscriber domains $D$ and $E$ to which $H$ and $I$ belong are each connected to two network service providers. Since nodes are assigned provider-oriented addresses (Section 3.1) in SIPP, nodes $H$ and $I$ each have two addresses. For this example, we assume that 64-bits are sufficient to form a node's globally routable address.

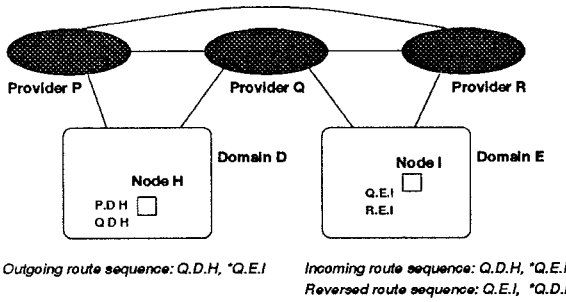


Figure 3: *Non Extended Addresses :* Nodes $H$ and $I$ are each represented by two 64-bit provider-oriented addresses. Node $H$ initiates an association with node $I$. The route sequence at $H$ and the route sequence reversed by $I$ are shown.

Let $H$'s provider-oriented addresses be denoted by $P.D.H$ and $Q.D.H$, and let $I$'s provider-oriented addresses be denoted by $Q.E.I$ and $R.E.I$, where each of these denotes a 64-bit SIPP address (Figure 3). The $D$ in $P.D.H$ and $Q.D.H$ are subscriber numbers assigned by providers $P$ and $Q$ respectively, and are not necessarily the same value. Each of these addresses is also an identifying address for the corresponding node. Thus, each node has two one-element address sequences.

Before initiating an association with $I$, $H$ learns $I$'s addresses from the Domain Name System (DNS, [17]). To send a packet to $I$, $H$ may use either of $I$'s addresses. Assume that $H$ chooses $Q.E.I$, since that address matches best with one of its own addresses. $H$ forms the following route sequence, using the rule described in Section 4.2: $\langle Q.D.H,\ \star Q.E.I \rangle$.

This packet traverses domain $D$ and provider $Q$ before entering domain $E$. When the packet arrives at $I$, the route sequence is unchanged (i.e., the active target does not advance). $I$ applies the reversal rule in Section 4.3 to get: $\langle Q.E.I,\ \star Q.D.H \rangle$.

### Provider Selection

Subscriber networks may be connected to more than one service provider (as in our example above). For cost or quality of service reasons, a node may choose to select different network service providers for different packets originating from the node. This *provider selection* feature is expected to be an important requirement in a commercial Internet.

The previous example is in fact a simple form of provider selection, since $H$ implicitly chooses provider $Q$ by choosing $Q.E.I$ as the destination address for $I$. Suppose, however, that $H$ wishes to send its packet through provider $P$ (Figure 4). Since $I$ is not attached to provider $P$ (and therefore does not have a $P$-oriented address), $H$ must explicitly indicate that it wants its packet to go through provider $P$. To do so, it forms the following route sequence:

$$\langle P.D.H,\ \star P.0,\ Q.E.I \rangle.$$

The active target of this route sequence is provider $P$'s clus-

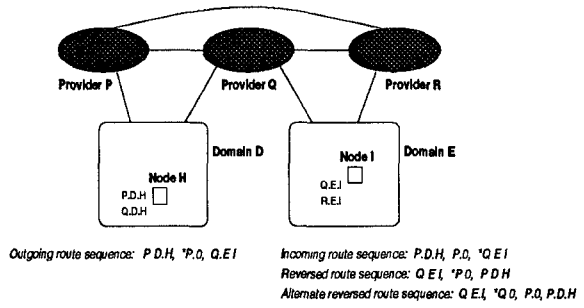


Figure 4: *Provider Selection:* To select provider $P$ for outgoing packets, node $H$ inserts $P$'s cluster address before $I$'s address. $I$ applies the reversal rules to direct return packets correctly. $I$ may, in turn, select $Q$ for return packets by inserting $Q$'s cluster address before $P$'s.

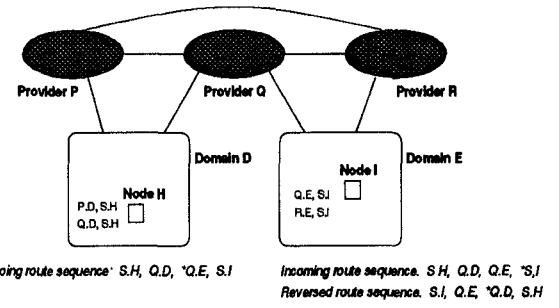


Figure 5: *Extended Address:* When 64-bits are not enough to form a globally routable address, the locations of nodes $H$ and $I$ are completely specified by address sequences with two 64-bit addresses each. This example shows how route sequences are formed with extended addresses.

ter address. Routers in domain $D$ direct packets with this route sequence to provider $P$. When a boundary router in provider $P$ receives this packet, it recognizes the active target as being itself, and advances the active target to the next element of the route sequence, producing:

$$\langle P.D.H,\ P.0,\ \star Q.E.I \rangle.$$

The packet is then routed to provider $Q$, eventually enters domain $E$ and is routed to $I$.

Upon receiving this packet, $I$ uses the reversing rules in Section 4.3 to get:

$$\langle Q.E.I,\ \star P.0,\ P.D.H \rangle.$$

In reversing the incoming route sequence, $I$ need not know that $H$ has explicitly selected a provider; from $I$'s perspective, $\langle P.0,\ P.D.H \rangle$ is an address sequence for $H$. Actually, 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 $P$'s provider prefix, $I$ cannot know that $P.0$ is redundant, and so includes both elements. With a cluster address discovery protocol, $I$ could learn $H$'s provider cluster address and optimize the header accordingly.

A packet carrying the above route sequence may exit domain $E$ via either provider $Q$ or $R$, depending on what routing considers the best path to provider $P$. If $I$ wants to route the return packet through provider $Q$, it would insert a "policy route" (in this case $Q$'s cluster address) to the route sequence to get:

$$\langle Q.E.I,\ \star Q.0,\ P.0,\ P.D.H \rangle.$$

### Extended Addresses

Assume now that 64-bits are no longer sufficient to form a globally routable address for nodes $H$ and $I$. Suppose instead that two 64-bit addresses suffice for this purpose. Then, nodes $H$ and $I$ are each represented by two address sequences (Figure 5). We denote $H$'s address sequences as $\langle P.D,\ S.H \rangle$ and $\langle Q.D,\ S.H \rangle$. The $S$ in the lower-order address is a subnet within domain $D$. Similarly, $I$'s address sequences are $\langle Q.E,\ S.I \rangle$ and $\langle R.E,\ S.I \rangle$. The lower order address in each address sequence must be a globally unique identifying address for the node.

As before, $H$ learns $I$'s address sequences from DNS and forms the following route sequence to send a packet to $I$:

$$\langle S.H,\ Q.D,\ \star Q.E,\ S.I \rangle.$$

The active target $Q.E$ causes the packet to be routed to domain $E$, where the active target is advanced to:

$$\langle S.H,\ Q.D,\ Q.E,\ \star S.I \rangle$$

and delivered to I.

$I$ applies the reversal rules to the incoming route sequence. $\langle Q.E,\ S.I \rangle$ matches the tail of the route sequence, so $I$ extracts the unmatched portion and reverses it to get $\langle Q.D,\ S.H \rangle$. It prepends the matched address sequence (this also happens to be source-capable) and forms the following outgoing route sequence

$$\langle S.I,\ Q.E,\ \star Q.D,\ S.H \rangle$$

to reply to $H$.

## 5.2 Multicast with Route Sequences

This section describes how the source route mechanism can be used to accomplish multicast. We describe two kinds of multicast: traditional multicast [3] and core-based tree (CBT) multicast. A node's notion of multicast addressing is extended so that a "multicast address" is seen as an address sequence rather than a single multicast address as is the case with IP. As with unicast, SIPP multicast address sequences are described using a series of 64-bit address elements. The final element of the multicast address sequence is always the SIPP multicast address.

When a node $N$ joins a multicast group, it appends the multicast address sequence to $\mathcal{Q}$ (Section 4.2). The address sequence formed depends on the type of multicast routing used. A multicast address sequence is not source-capable.

### Traditional Multicast

In traditional 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 multicast address for traditional multicast contains only one element—the SIPP multicast address for the group.

Suppose we consider the extended address case shown in Figure 5. Assume that $H$ is transmitting a traditional multicast with multicast address $M$, and that $I$ is a member of group $M$. Thus, $\langle M \rangle$ is a valid address sequence for $I$. $H$ attaches the route sequence $\langle S.H,\ Q.D,\ \star M \rangle$ to the multicast packet.

The active target in this route sequence is the SIPP multicast address representing the group. Intermediate routers forward the packet to all members of the group including $I$. Each member receives the packet with the route sequence unchanged.

If $I$ wishes to respond unicast to $H$, it executes the reversing rules (Section 4.3) in the following way. The address sequence $\langle M \rangle$ matches the tail of the received route sequence best. $I$ reverses the unmatched part to get $\langle Q.D,\ S.H \rangle$. Since a multicast address cannot be used as a source address (that is, multicast address sequences are not source-capable), $I$ knows to prepend one of its unicast address sequences to the route sequence, producing:

$$\langle S.I,\ Q.E,\ \star Q.D,\ S.H \rangle.$$

### Core-based Tree Multicast

In core-based tree multicast, multicast packets are routed to their destinations in two phases [2]. In the first phase, a packet from a sender to a multicast group is unicast towards a *core* router. The delivery tree for the multicast group is rooted at this router. The core router initiates the second phase of CBT multicast: it multicasts the packet to the group members along the delivery tree. Thus, a core-based tree multicast address sequence contains multiple addresses—one or more unicast addresses to encode the address sequence for the core router, and one SIPP multicast address.

Suppose we consider the simple address example shown in Figure 4. Assume that $H$ is transmitting a core-based tree multicast with multicast address $M$, that $C$ is the unicast address for core router for the group $M$, and that $I$ is a member of group $M$. Therefore, $\langle C, M \rangle$ is a valid address sequence for node $I$. To initiate the first phase of the multicast, $H$ forms the following route sequence:

$$\langle P.D.H,\ \star C,\ M \rangle.$$

This packet is routed on the unicast address C until it reaches the core router. This router advances the active target, to get the route sequence:

$$\langle P.D.H,\ C,\ \star M \rangle.$$

This causes the packet to be multicast along the delivery tree for $M$ towards each member of $M$.

Each group member (including $I$) receives the packet with the above route sequence. To generate a (unicast) reply, $I$ follows the reversing rules described in Section 4.3. The address sequence, $\langle C,\ M \rangle$ matches the tail of the incoming route sequence best. $I$ reverses the unmatched portion and prepends a source-capable address sequence to get $\langle Q.E.I,\ P.D.H \rangle$.

## 5.3 Host Mobility with Route Sequences

This section shows how route sequences can be used for mobile communication. First, we introduce some terminology. A *mobile host* is a node that is able to maintain its network connections even after being physically moved. A *correspondent host* is a node that has a network connection open to a mobile host. A correspondent host may itself be mobile. The *foreign agent* is the SIPP router to which the mobile host is attached after it moves. Finally, the *home agent* is a SIPP node that is aware of the mobile host's active location. Each mobile host has a preconfigured home agent.

In our example, assume that $H$ is a mobile host and that $I$ is its correspondent host, both with the (extended) address sequences shown in Figure 5. Initially, a unicast packet from $I$ to $H$ carries the route sequence

$$\langle S.I,\ Q.E,\ \star Q.D,\ S.H \rangle.$$

Now suppose $H$ moves to the vicinity of a foreign agent with a cluster address $B.0$. $H$ discovers the foreign agent's cluster address through periodic router advertisements (similar to an IS hello advertisement in ES-IS [11]). $H$ also periodically advertises its addresses (similar to the ES hello advertisement in ES-IS). This latter advertisement contains $H$'s identifying address.

Then, through a mechanism beyond the scope of this paper, $H$ informs its home agent of its new foreign agent. Packets carrying the old route sequence from $I$ are intercepted by the home agent. The home agent forwards these packets to the foreign agent, which forwards them to $H$.

After the $H$ has discovered $B.0$, it alters the route sequence on all outgoing packets to:

$$\langle S.H,\ B.0,\ \star Q.E,\ S.I \rangle.$$

It is sufficient to include only $H$'s identifying address in the route sequence; we assume that the foreign agent is within $H$'s identifying address's $(S.H)$ routing scope. When $I$ reverses the incoming route sequence from $H$, it forms the following route sequence:

$$\langle S.I,\ Q.E,\ \star B.0,\ S.H \rangle.$$

This causes packets to $H$ to be routed through the foreign agent.

# 6 Implications of Address Sequences

Representing the location of a node by a sequence of 64-bit addresses impacts unicast address assignment, routing protocols, and router performance. This section discusses these implications in greater detail.

## 6.1 Impact on Unicast Address Assignment

Like SIPP addresses, an address sequence is structured into a number of fields. The bit position of the field boundaries is not constrained to be byte-aligned. However, the use of the source routing mechanism for extending SIPP unicast addresses places two restrictions on the assignment of unicast addresses.

First, since the source route mechanism requires a router to make a forwarding decision based on 64-bit chunks, a field in an address sequence cannot straddle a 64-bit boundary.

Second, when a multi-element address sequence is used to represent a node's extended address, the lowest-order address must be globally unique for uniquely identifying the node, and the highest-order address must be globally unique so that packets can always

be delivered successfully to the top of the hierarchy. The other addresses need not be globally unique (because, by carefully configuring routing, it is possible to parse those addresses in the correct context).

## 6.2 Router Handling of Extended Addresses

SIPP routing algorithms are identical to those used with the CIDR version of IP, except that the 32-bit IP addresses are replaced with 64-bit SIPP addresses. This is true even when extended addresses are in use. This is possible because a SIPP unicast forwarding table lookup is made by looking at only a single (64-bit) SIPP address. The result of such a forwarding table lookup may be to advance the route header, causing the router to look at the following address in the route sequence. This latter routing table lookup, however, is made without consideration of the previous lookup.

Because the forwarding table lookup only involves a single address, the routing algorithm only need carry a single address. If extended addresses are used, however, care must be taken in the distribution of routing information. In particular, routing information cannot be leaked across routing hierarchy boundaries that coincide with the boundary between two SIPP addresses in an extended address. For instance, consider the case where the subscriber prefix is encoded in the upper address of a two-element extended address, and the subnet and host parts are encoded in the lower address. Because the subnet part is not globally unique, if the lower address were advertised outside of the subscriber network, routing in outside networks could fail. This failure mode does not exist in traditional internet protocols such as IP and CLNP ([12]), where the entire destination address is examined by every router. If routing protocols in SIPP advertise complete extended addresses, then this failure condition can be detected, though not corrected without manual router configuration.

Finally, for extended addresses to work with single-address routing algorithms, routers must recognize addresses that identify themselves, and advance the routing header upon receiving such an address. The high-order part of a router's extended address is among the addresses that a router identifies as its own for the purpose of advancing the routing header.

## 6.3 Processing Source Address Sequences

For certain types of multicast routing—namely those based on building multicast trees from the source—it is necessary for a router to examine the source address as well as the destination (multicast) address when forwarding a packet. Since a SIPP source may be represented by an address sequence (e.g., a mobile address sequence or an extended address), the router must examine the source address sequence in such situations.

All of the addresses of the source address sequence (except for the identifying address) are in the routing header. To make its forwarding decision, the router examines one or more addresses immediately preceding the active target. These are the addresses in the SIPP Routing Header immediately preceding the address indicated by the *next address* field, upto and including the address in the source address field. We call this behavior "peek-behind".

# 7 Possible SIPP Enhancements

The use of route sequences in SIPP also impacts host and router performance, flexibility in modeling different routing and addressing paradigms, and security. This section discusses solutions for these problems.

## 7.1 Using SIPP Flow Labels To Improve Performance

The SIPP *flow label* (Figure 2) may be used by senders to label packets which require special handling (e.g. "real-time" service) by SIPP routers. The flow label field consists of two sub-fields, one of which is the *flow ID* field (for a more detailed description of the SIPP flow label, see [5]). Packets from the same source and carrying the same (non-zero) flow ID are associated with the same special handling state established (by mechanisms which are beyond the scope of this paper) in routers on the path from source to destination.

SIPP hosts may exhibit poor performance because they need to apply the route sequence reversal rules on every incoming packet (Section 4.3). SIPP nodes can use the following property of flow IDs to improve performance: if a source assigns a non-zero flow ID to outgoing packets, then, when the outgoing route sequence changes, the source must assign a new flow ID on outgoing packets. Receivers can cache the flow ID on the last packet to have arrived from a sender; receivers reverse incoming route sequence only when the incoming packet's flow ID is different from the cached value or when the flow ID is zero.

The above property of flow IDs can be used to improve router performance as well. In each packet, the source address and a non-zero flow ID uniquely determines the next hop router and outgoing interface(s) at each router on the path of the packet. Routers can cache this mapping to speed up packet forwarding. This is particularly useful for forwarding decisions where "peek-behind" is necessary (Section 6.3).

## 7.2 Peek Ahead

In SIPP, all *unicast* forwarding decisions are made by looking at a single 64-bit address only. As a consequence, some other forwarding decisions may require "peek-behind". It is instructive to consider what happens if "peek-ahead" (the ability of routers to look beyond the active target of a route sequence without advancing the active target) is allowed. This feature results in a more expensive forwarding algorithm, but has some advantages.

One advantage is that the restrictions in distribution of routing information and address assignment (Section 6.1) are eliminated.

Another advantage is the increased flexibility in modeling CBT multicast. In the example of Section 5, the packet travels all the way to the core of the multicast tree before starting the second, multicast phase of its transmission. If peek-ahead is allowed, then routers on any of the trees of the core could peek-ahead to the multicast address and determine if they are on the tree for that particular multicast group. If they are, then they could immediately initiate the multicast phase of the transmission, resulting in lower latency for the transmission.

Also, with peek-ahead, peek-behind is no longer necessary. This

is accomplished by replicating the source address sequence in the route sequence after the multicast address. If the active target in a packet were a multicast address, routers would peek-ahead to examine the source address without advancing the active address of the route sequence.

In summary, with peek-ahead, forwarding can be accomplished with one mechanism (source routing with peek-ahead) rather than with two (source routing without peek-ahead plus source routing with peek-behind). This allows for a cleaner implementation overall.

## 7.3 Partial Route Sequence Reversal

The route sequence reversal rules described in Section 4.3 can, in some cases, lead to non-optimal, undesirable, or even incorrect behavior. An example of non-optimal behavior is the provider selection example of Section 5.1, where the reversed cluster address is redundant. An example of undesirable behavior is a policy route, when the desired policy in reverse direction is different from that in the forward direction.

An example of incorrect behavior is a node-level source route, where the nodes have extended addresses. For example, consider the case where the source route in the forward direction is $\langle \ldots A_h,\ A_l,\ B_h,\ B_l,\ C_h,\ C_l, \ldots \rangle$, where $A$, $B$, and $C$ are target nodes in the source route, and the subscripts $h$ and $l$ denote the high- and low-order parts of the address sequence. When this source route is reversed, producing $\langle \ldots C_l,\ C_h,\ B_l,\ B_h,\ A_l,\ A_h, \ldots \rangle$, the low-order part of each extended address precedes the high-order part. This will not work in general, because a packet must first be routed to the high-order address before it can be routed to a low-order address.

In all of the above examples, the problem arises when the the route sequence contains more than the minimum information necessary to return the packet to the source (that is, more than just the source's and destination's address sequence). One solution is to include a field in the Routing Header that indicates the length of the sender's address sequence. By default, a receiver discards all addresses in the route sequence that are not part of its own address sequence and not in the sender's address sequence.

## 7.4 Security

With SIPP, a host automatically reverses the route sequence of any received packet for the purpose of forming the route sequence for the return packet. This opens SIPP hosts to the following eavesdropping attack.

Assume that SIPP hosts $X$ and $Y$, with addresses $A$ and $B$ respectively, are exchanging packets with route sequences $\langle A, B \rangle$ in the direction from $X$ to $Y$ and $\langle B, A \rangle$ in the opposite direction. Now, assume that a SIPP host $P$ with address $C$ that is not otherwise on the path between $X$ and $Y$ wishes to receive the packets of the exchange. Node $P$ sends a packet to $Y$ with route sequence $\langle A, C, B \rangle$. $Y$ will reverse the new route sequence, and send return packets with route sequence $\langle B, C, A \rangle$. This will cause all packets between $X$ and $Y$ to be routed through $P$.

There a number of ways to defend against this particular attack. The strongest defense is to use the optional SIPP Authentication Header [1]. This contains the sender's digital signature, making it difficult for the eavesdropper to modify the route sequence undetected. A simpler but weaker defense is to use the flow ID enhancement described in Section 7.1, and to add an optional header that allows a host to indicate what the previous flow ID was when it modifies the routing header. Flow IDs are pseudo-random [5], and since the eavesdropper is not generally able to observe packets from the hosts it wants to eavesdrop on, it is difficult for the eavesdropper to guess the current flow ID. Finally, a receiver can learn (from a trusted third party) the extended address of the sender, and refuse to include any addresses in the route sequence other than source and destination addresses.

# 8 Related Work

Other work has proposed the use of source routing to achieve one or more of the features described in Section 5. For instance, both Inter-Domain Policy Routing [21] and Source Demand Routing [6] use source routing to encode specialized policy routes. Similarly, [20] describes the use of source routing to solve the host mobility problem. SIPP's route sequence mechanism generalizes these uses of source routing to different routing and addressing paradigms.

Other existing network protocols, such as CLNP (Connection-Less Network Protocol, [12]) and IP have loose source routing mechanisms. Unlike SIPP, however, CLNP and IP nodes are not required to correctly reverse loose source routes (route sequences) or recognize that packet sources and destinations may be represented as address sequences.

Finally, other mechanisms may be defined for achieving one or more of the advanced routing and addressing features enabled by SIPP's used of generalized source routing. For instance, the CLNP packet header allows variable length node addresses. Encapsulation [16] can be used to emulate provider selection [19] and mobility [14] in IP.

However, we believe that the SIPP approach of having a single framework for providing advanced routing and addressing functions is more desirable than the other protocols' approach of having multiple different mechanisms for different functions. A single mechanism is more efficient than having multiple mechanisms. Separate mechanisms require separate checks and actions, one for each mechanism; a single mechanism, provided that it is itself efficient (which is the case with SIPP's source route mechanism), allows checking for and accomplishing multiple functions to be done with one action.

A single powerful mechanism installed in all systems also allows easier evolution to a new routing and addressing function, if that new function can be implemented with the single mechanism. Since the loose source route mechanism of SIPP is general in nature (as evidenced by the number of different functions it can support), we expect that it will support a wide variety of new functions.

# 9 Summary

In this paper, we describe the methods used to achieve advanced routing and addressing functions in SIPP, a proposed replacement for IP. Our basic approach generalizes IP's loose source route mechanism; we show how a number of routing and addressing paradigms can be modeled as routing to a series of intermediate destinations on the path to the final destination.

In SIPP, the addresses representing these intermediate destina-

tions can be one of: the multicast address, the hierarchical unicast address, and the cluster address. The multicast and hierarchical unicast addresses are analogous to those of IP, except they are 64 bits in length. The cluster address is used to send a packet to one of a number of SIPP nodes in a hierarchical cluster of nodes, and is useful for, among other things, provider selection.

To use loose source routing for advanced routing and addressing, two functions are required of SIPP nodes. First, SIPP nodes must be able to view their own address as a sequence of 64-bit addresses. Second, they must be able to reverse the loose source route in a received packet for use in return packets. We illustrate that, if nodes perform these functions correctly, loose source routing can be used to achieve mobility, extended (variable-length) addressing, provider selection, and source-based and core-based multicast.

We argue that having a single powerful mechanism to achieve a wide range of routing and addressing functions is preferable to having multiple specific mechanisms, one for each function. A single mechanism is easier to implement and more efficient than multiple mechanisms, and it makes future evolution easier.

Unfortunately, the use of route sequences has some undesirable side-effects. For instance, SIPP has a new failure mode resulting from the fact that routers do not necessarily need to examine the high-order part of an extended address to forward a packet. SIPP is also prey to an eavesdropping attack that does not exist with IP. We discuss techniques for overcoming some of these shortcomings.

## Acknowledgements

Steve Deering, the inventor of SIP, is responsible for many of the original ideas described in this paper, as well as much of the terminology. Bob Gilligan, Bob Hinden, Christian Huitema and Erik Nordmark also contributed to the design of SIPP routing and addressing. Finally, Susan Thomson, Andrew Heybey, and the anonymous referees provided insightful comments on earlier versions of this paper.

## References

[1] R. Atkinson. SIPP Authentication Header. Internet Draft, work in progress, available via anonymous FTP from ds.internic.net, April 1994.

[2] A. Ballardie, P. Francis, and J. Crowcroft. An Architecture for Scalable Inter-Domain Multicast Routing. *ACM SIGCOMM Symposium on Communication Architectures and Protocols*, pages 85–95, September 1993.

[3] S. Deering. Multicast Routing in Internetworks and Extended LANs. *ACM SIGCOMM Symposium on Communication Architectures and Protocols*, August 1988.

[4] S. Deering. SIP: A Simple Internet Protocol. *IEEE Network*, 7(6):16–28, May 1993.

[5] S. Deering. Simple Internet Protocol Plus (SIPP) Specification. Internet Draft, work in progress, available via anonymous FTP from ds.internic.net, February 1994.

[6] D. Estrin, S. Hotz, and Y. Rekhter. A Unified Approach to Inter-Domain Routing. Request for Comments 1322, DDN Network Information Center, May 1992.

[7] P. Francis. A Near-term Architecture for Deploying Pip. *IEEE Network*, 7(6):30–37, May 1993.

[8] P. Francis. Advanced Routing and Addressing using Generalized Source Routing. *Submitted to the Journal of Internetworking*, 1994.

[9] V. Fuller, T. Li, J. Yu, and K. Varadhan. Classless Inter-domain Routing (CIDR): an Address Assignment and Aggregation Strategy. Request For Comments 1519, DDN Network Information Center, September 1993.

[10] P. Gross and P. Almquist. IESG Deliberations on Routing and Addressing. Request For Comments 1380, DDN Network Information Center, November 1992.

[11] International Organization for Standardization. *End System to Intermediate System routeing exchange protocol for use in conjunction with the Protocol for providing the connectionless-mode network service (ISO 8473)*, ISO 9542, 1988.

[12] International Organization for Standardization. *Protocol for Providing Connectionless-mode Network Service*, ISO 8473, 1988.

[13] J. Ioannidis. Source Routing Experiments. IETF Mobile-IP Working Group mailing list, July 1992.

[14] J. Ioannidis, D. Duchamp, and G. Q. Maguire. IP-based Protocols for Mobile Internetworking. *Proceedings of 1991 ACM Sigcomm Symposium on Communication Architectures and Protocols*, September 1991.

[15] D. Katz and P. S. Ford. TUBA: Replacing IP with CLNP. *IEEE Network*, 7(6):38–47, May 1993.

[16] D. Mills and R. Woodburn. A Scheme for an Internet Encapsulation Protocol: Version 1. Request for Comments 1241, DDN Network Information Center, July 1991.

[17] P. Mockapetris. Domain Names - Concepts and Facilities. Request For Comments 1035, DDN Network Information Center, November 1987.

[18] J. Postel. Internet Protocol. Request For Comments 791, DDN Network Information Center, September 1981.

[19] Y. Rekhter. Selecting a Direct Provider. Internet Draft, work in progress, available via anonymous FTP from ds.internic.net, April 1994.

[20] Y. Rekhter and C. Perkins. Loose Source Routing for Mobile Hosts. Internet Draft, work in progress, available via anonymous FTP from ds.internic.net, February 1994.

[21] M. Steenstrup. An Architecture for Inter-Domain Policy Routing. Request for Comments 1477, DDN Network Information Center, July 1993.