Network Working Group
Internet-Draft
Expires: May 22, 2008

C. Vogt
Ericsson

November 19, 2007

Six/One: A Solution for Routing and Addressing in IPv6
draft-vogt-rrg-six-one-01.txt

Status of this Memo

Copyright Notice

Abstract

There is heightened concern about the growth of the global routing table and the increased frequency of routing table updates. Both is due to the use of provider-independent addressing space in edge networks, which accommodates operators' desire to move traffic quickly and easily from one provider to another. The main recent proposals attempt to solve this problem by hiding provider-independent addressing space through a level of indirection. Unfortunately, indirection requires a non-trivial distribution of address translation information across the Internet. This is either slow, or costly in terms of signaling overhead, or both. Security and transitionability are further issues. This document proposes an alternative solution, which is based on a single, provider-dependent addressing space and hence avoids the problems of indirection. The solution meets the objectives of edge network operators while limiting the size of the routing table and its update frequency.

Table of Contents

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [1].

1. Introduction

The Internet traditionally consists of three classes of players: users, edge network operators, and providers. Users operate hosts for which they desire efficient, available, and reliable Internet connectivity. Edge network operators provide the infrastructure that hosts needs to communicate, collectively called the "edge domain". An edge network can indepently route packets between two attached hosts, but for global Internet connectivity, it must connect to a provider. Providers jointly form a "transit domain" via which packets can be exchanged between edge networks.

Edge network operators are naturally eager to meet the expectations of users because they have a direct business relationship with the users. An important tool edge network operators employ to accomplish this is traffic engineering, which helps them to utilize available edge network capacities in an optimum way. Edge network operators further desire the ability to select providers in a dynamic manner because the quality of a host's Internet connectivity depends on the provider just as well as on the edge network. Technology should permit flexible transition from one provider to another, so-called "rehoming". Technology should also allow an edge network operator to connect to multiple providers, so-called "multi-homing", and to extend traffic engineering to dynamically reroute traffic between those connections.

The crux with the new desire for flexible provider selection is that the Internet was not designed to support it. For scalability reasons, the preferred strategy for allocating Internet addressing space is to hand out contiguous address blocks to providers, which in turn delegate parts of their block to the edge networks they serve. The address that a host uses to describe a point of Internet attachment consequently identifies the provider via which the host is reachable. The benefit of this is more efficient routing inside the transit domain because the global "routing table", the routing state that needs to be maintained and consulted on a per-packet basis, can be limited to one entry per provider. The routes connecting transit and edge domains can then be maintained locally by the respective providers and the edge networks they serve. A smaller routing table furthermore reduces the number of updates it is subject to.

On the other hand, provider-dependent addressing brings about grave limitations for edge domain operators. An edge network operator that wishes to rehome must reconfigure networking equipment for traffic control and analysis as well as applications with preconfigured addresses on hosts. This affects routers just as well as middleboxes that store addresses, such as firewalls and intrusion detection systems. Rehoming is therefore expensive and disruptive. A multi-

homed edge network is limited in its ability to pursue traffic engineering because provider-dependent addresses effectively move the priviledge of provider selection to the hosts: A host obtains an address from each provider, and to ensure bidirectionality and topological correctness, packets that the host sends must go via the provider that is identified by the packet's source address.

A natural desire of edge network operators is therefore to gain provider-independent addressing space. This would facilitate rehoming without reconfiguration costs, as well as flexible provider selection during multi-homing. The drawback of provider-independent addressing space for edge networks is that it would require an entry per edge network in the global routing table. The routing table size would thus increase to a substantial extent and effectively slow down the routing process. Moreover, the only way for an edge network to select the provider through which it was to be reached would be an update to its entry in the global routing table. This would take effect slowly, and it would have the undesired consequence of more frequent updates to the global routing table.

The conflict of interests between the transit domain and the edge domain calls for a new approach to routing and addressing that satisfies the following primary objectives:

1. Routing table size. The size of the global routing table must be kept linear in the number of providers rather than linear in the number of edge networks.

2. Routing table robustness. Traffic engineering in edge networks or edge network rehoming must not require updates to the global routing table.

3. Egress provider selection. A multi-homed edge network must be able to select a provider for egress packets. Such selection must take effect quickly.

4. Ingress provider selection. A multi-homed edge network must be able to select a provider for ingress packets. The mechanism that this requires for packet rerouting across the transit domain must be responsive.

5. Rehoming. An edge network must be able to smoothly change the set of providers it connects to, without incurring costly reconfiguration of applications and networking equipment for traffic control and analysis.

6. Transition. There must be a transition path for incremental deployment of a routing and addressing solution, which allows upgraded parts of the Internet to communicate with legacy parts.

Beside these primary objectives, there are a number of secondary objectives. A routing and addressing solution that satisfies some or all of these objectives would be of additional value for host, edge network operators, or providers:

7. Host preferences. A host should have a means to suggest to the edge network which provider it would prefer its packets to pass through.

8. Routing performance. Traffic characteristics such as packet propagation latencies, packet loss probabilities, or jitter should not change.

9. Deployment incentives. Deployment of a routing and addressing solution should yield direct benefits to those entities investing in the deployment.

10. Integrability on hosts. A host-based routing and addressing solution should be integrable with protocols for host mobility and host multi-homing.

11. Integrability in the network. A network-based routing and addressing solution should be integrable with source address validation technology.

The main existing routing and addressing proposals are based on a level of indirection between addressing space for use inside the edge domain, and addressing space for efficient packet routing across the transit domain. Two indirection functions perform forward and reverse translations between edge domain addresses and transit domain addresses. Specific proposals differ with respect to whether this translation occurs through address rewriting or tunneling, and, more importantly, with respect to where the indirection functions are located. While one indirection function always borders the edge network which address space is being translated, the other indirection function may be located at the border of a remote edge network, or inside the transit domain. A consequence of locating an indirection function inside the transit domain is that edge network addressing space must be provider-dependent. And to avoid a dependency on a single provider, the edge network addressing space should be shared as anycast addressing space by multiple providers, each with its own indirection function.

The strength of indirection is that it requires special support only

in the form of indirection functions. It can thus be kept transparent to hosts, and to the edge network except for the entity providing an indirection function. Like provider-independent addressing space in general, indirection thereby eliminates the costly reconfiguration overhead that edge network rehoming normally entails. It also allows a multi-homed edge network to select a provider for egress packets and -- to the extent a remote indirection functions can be altered quickly enough -- for ingress packets as well. Indirection finally preserves the size of the global routing table or the frequency by which this is updated.

A significant drawback of indirection is the need for a mechanism that can distribute address translation information for an edge network to indirection functions at other edge networks or inside the transit domain. A suitable distribution mechanism needs to affect a difficult trade-off between update latency and signaling overhead, while still remaining responsive to urgent rerouting decisions of edge network operators, such as for the purpose of provider fail-over. A suitable distribution mechanism must also be robust to impersonation and denial-of-service attacks. Both can be provided through authentication. For scalability reasons, this is likely to require public-key-based authentication, and hence notable processing overhead and periodic inquiries about subverted and changed public keys.

Indirection techniques that use provider-independent addressing space inside the edge domain furthermore fail to provide a smooth transition path. From the perspective of a correspondent host, the address of a host behind an indirection function depends on whether the edge network of the correspondent host already supports the indirection. While it is feasible to modify DNS to tailor address resolution to the resolving host, means of address resolution aside from DNS, such as via SIP signaling or peer-to-peer protocols, might not be fixable and become completely defunct [4]. Another undesired consequence is that, in case the edge network of the correspondent host is legacy, but address resolution succeeds nonetheless, the indirection function at the host would have to translate addresses also in the payloads of the correspondent host's packets, thus behave as a classic network address translator.

The routing and addressing solution proposed in this document, Six/One, avoids the problems of indirection by maintaining a single addressing space of IPv6 addresses for both the edge domain and the transit domain. Six/One borrows from Shim6 [5] the concept that multi-homed edge networks use provider-dependent addressing space from each of their providers, and hosts use addresses from all addressing spaces interchangeably without breaking active communication sessions. The novel concept of Six/One is that a

host's addresses differ only in their high-order bits, which enables an edge network or a provider to change the address in a packet on the fly depending on the provider to with the packet is routed. The address from which a host contacts a correspondent host serves as a suggestion to the edge network which provider the host's packets should be routed through. The edge network may follow this suggestion, or rewrite the high-order bits of the address in accordance with a provider of its own choice. Different than in Shim6, edge networks thus retain the ability to select a particular provider for ingress and egress packets. Hosts adapt to address rewrites in that they modify subsequent packets accordingly before injecting them into the network. Like other protocols that enable a host to change its address during an active communication session -- including, besides Shim6, also Mobile IPv6 and the Host Identitity Protocol --, Six/One adds to packets a small piece of information that enables the receiving hosts to reverse address rewrites. The novel concept in Six/One is that this information can also be used inside an edge network to identify a remote host despite address changes. Moreover, the high-order bits in addresses from local addressing space can be masked when stored in applications or networking equipment for traffic control and analysis, so as to spare costly reconfiguration overhead in the event of rehoming. This resembles the concept of 8+8 [6], with the main difference that hosts remain to be aware of their full addresses, including the high-order bits, and that they can suggest to the edge network a provider for their packets as explained above.

Through the use of provider-dependent addressing space in both the edge domain and the transit domain, Six/One preserves both the size of the global routing table as well as the frequency at which the routing table is updated. For the same reason, Six/One precludes degradations in routing performance. A three-phase transition path will smoothly lead towards a universal deployment of Six/One. Deployment will be accelerated by the right incentives, since even partial deployment will provide advantages for the pioneers. The host support that is needed for Six/One integrates well with host mobility and host multi-homing procotols. The required network support facilitates source address validation at no extra costs for the edge network operator or provider. In summary, Six/One satisfies all of the aforementioned primary and secondary objectives.

2. Conceptual Overview

This section gives a conceptual overview of the Six/One protocol, and it provides rationale for the design decisions made.

The refrainment from introducing separate addressing spaces for the edge domain and the transit domain implies that the IPv6 addressing architecture (RFC 4291) and the preferred policy of allocating provider-dependent addressing space to edge networks remain as is. From a host's perspective, an IPv6 address is still split into a 64-bit "subnet prefix" and a 64-bit "interface identifier", as shown in Figure 1. A subnet prefix is advertised by an access router, and the host extends this with an interface identifier to form a full address. From an edge network's perspective, a subnet prefix can further be decomposed into a "routing prefix", which identifies the provider from which the addressing space was obtained, and a "subnet ID", which is used by the edge network to identify internal links. The length of the routing prefix determines the size of the addressing space, which is up to the contract between the edge network and the provider. The length of the subnet ID is such that, together with the routing prefix, it adds up to 64 bit.

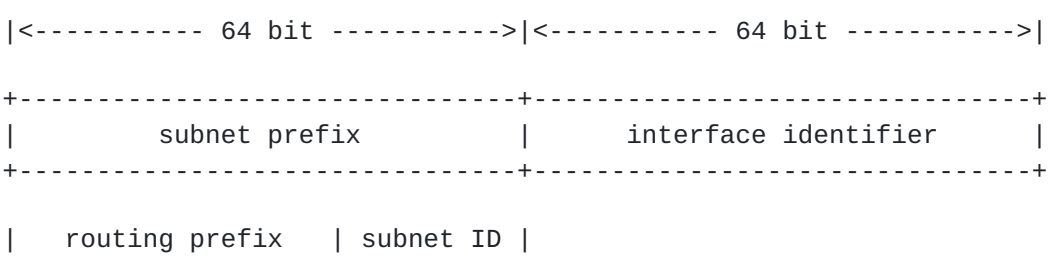

Figure 1: IPv6 address structure

What Six/One adds to IPv6 is an ability for hosts to configure sets of addresses, so-called "address bunches", that differ only in the routing prefixes, and to use these addresses interchangeably without breaking active communication sessions. An instance of Six/One at the host's IPv6 layer ensures that switches between addresses from the same bunch are kept transparent to protocols above.

The concept of address bunches enables a host sending a packet, or a router on the packet's path, to change the routing prefix in an address of the packet without destroying the relationship between the address and the host to which it belongs. (A host generally does not know the length of a routing prefix, so technically, it rewrites an entire 64-bit subnet prefix. But since all addresses in a bunch have the same subnet ID, only the routing prefix can change during this operation.) As the routing prefixes in the addresses of a packet

define the providers through which the packet enters or leaves the transit domain, routing prefix rewriting can force the packet to go via a different provider pair than the one corresponding to the original addresses. The correspondent host receiving the packet recognizes and reverses routing prefix rewrites so that packets are handed to protocols above Six/One as they were generated initially. The correspondent host further memorizes a routing prefix rewrite and adapts in that it directly rewrites the addresses in packets that it generates itself before injecting the packets into the network. This not only relieves the network from the burden of continued routing prefix rewriting, it also ensures that packets exchanged in either direction pass the same pair of providers. The result is a quick, bidirectional adoption of a new provider, which is essential in particular during provider fail-over.

To support the new interchangeability of addresses, a host must be aware of its own address bunch as well as of the address bunch of the "correspondent host" it communicates with. The host must further remember which address from each bunch it is to use at protocols above Six/One. The host maintains all of this information as a per-communication-session "context". A context is securely created on both sides of a connection during session establishment. Each context is assigned a "context identifier", which is carried in all packets of the session to aid the retrieval of the right context at the receiving host.

Six/One itself handles edge network multi-homing, but it is not a solution for host mobility, host multi-homing, or host identity management. These latter challenges must be addressed by a separate protocol such as Mobile IPv6 or the Host Identity Protocol. However, it is possible to integrate Six/One with host mobility, host multi-homing, or host identity protocols so as to reduce network protocol stack complexity and signaling overhead. For example, all of these protocols include information for context retrieval in packets, such as a Six/One context identifier, a Mobile IPv6 home address, or an IPsec security parameter index in the case of the Host Identity Protocol. This information could be shared amongst the protocols. The new paradigm of address bunches and address rewriting in routers may furthermore be incorporated into future-Internet architectures. One example is the Node ID architecture [7], which builds on the Host Identity Protocol and augments the Internet by a mechanism for identity-based overlay routing to help hosts discover their mutual IP addresses.

3. Protocol Operation

This section describes the operation of Six/One in detail.

3.1. IP Address Configuration

To allow the routing prefix of an address to be rewritten without changing the address owner, addresses must be configured in bunches. This may happen through manual preconfiguration, or through stateless or stateful auto-configuration. A host that uses stateless address auto-configuration can configure an address bunch through successive execution of the protocol for each address in the bunch. In the unlikely event of a collision, the host discards the incomplete address bunch and tries anew with a different interface identifier.

The signaling overhead in the above application of stateless address auto-configuration does not differ from the standard use if no collision occurs, or if a collision occurs already during the configuration of the first address in the bunch. Some extra signaling overhead is spent in case a collision occurs after the first address has been configured because every previous address configuration was then in vain. However, if all neighbor hosts on the same link use address bunches as well, a collision can only occur during the configuration of the first address in a bunch.

[To be done: Stateful address auto-configuration with DHCPv6.]

Six/One ensures that routing prefix rewrites are transparent to protocols above Six/One, so that they do not cause communication sessions to abort. In the following it will be assumed that the host configures a single address bunch. This assumption goes without loss of generality.

3.2. Source Address Selection

All addresses in a bunch are visible to protocols above Six/One, and all may be advertised in DNS. An application may select any of these addresses as a local address for a new communication session, and any address that DNS advertises for the correspondent host may be used as a remote address. The local and remote addresses used by protocols above Six/One are called "primary addresses". Primary addresses never change throughout a communication session. The local and remote addresses into which Six/One translates a packet's source and destination addresses are called "active addresses". Figure 2 illustrates which values a packet's source and destination addresses take before and after processing by Six/One.

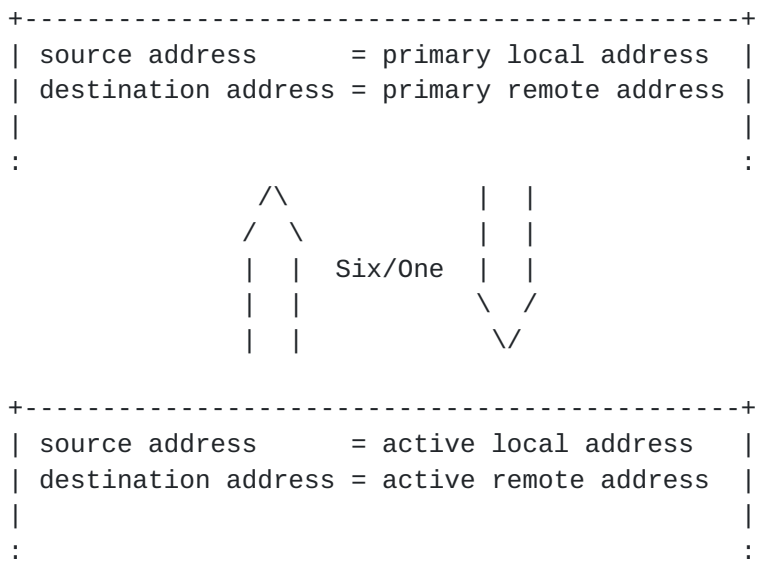

Figure 2: Packet addresses before and after Six/One processing

A host is in general agnostic with respect to the topological meaning of a subnet prefix, and it simply considers the subnet prefix an opaque 64-bit value. The host then selects a source address for a new communication session without preference. However, if a host is able to map routing prefixes onto the corresponding providers, the host can express a preference for a particular provider by picking a source address with that provider's routing prefix. Unless the suggestion gets overwritten by the edge network, this allows the host to have its packets routed via a provider of its own choice.

3.3. Initiating a Session

To be able to pursue, verify, and reverse routing prefix rewrites in the addresses of ingress and egress packets, a host must be aware of the local and remote addresses that are legitimate for a particular communication session as well as which of these addresses are primary. The host maintains this information in "contexts". The "local context" for a communication session comprises the primary local address used in the session, the address bunch from which the primary local address was taken, and a context ID that is unique for all local contexts of the host. The host further maintains a "remote context" for each communication session, which equals the local context of the respective correspondent host. The remote context is retrieved during session establishment. A remote context is identified by the combination of the context ID chosen by the correspondent host, and any address in the correspondent host's address bunch.

A host that wishes to initiate a communication session with a correspondent host first allocates a local context based on the primary local and remote addresses chosen by the application as well as the address bunch to which the local address belongs. The host assigns the local context a context ID that has not been assigned to another local contexts. In case the host already has a local context for the same primary local and remote addresses, then this may be reused for the new communication session. The local context is finally bound to the session in an implementation-specific manner.

The host further allocates a remote context for the communication session. If a remote context already exists for the same primary local and remote addresses, then it may be reused for the new communication session. Otherwise, the host creates a preliminary remote context at this stage, which contains only the primary local and remote addresses. The missing parts of the remote context will be retrieved subsequently during session establishment. The remote context, too, is bound to the session in an implementation-specific manner.

Figure 3 shows an example of the host's local and preliminary remote contexts at the time the correspondent host is contacted. The host has elected to use the local address RP1a:SID1::IID1 as a primary local address in the communication session, where "RP1a" denotes the routing prefix, "SID1" the subnet ID, and "IID1" the interface identifier. The host's address bunch also contains a second address with routing prefix "RP1b". The correspondent host's address RP2a:SID2::IID2 is used as the primary address in the preliminary remote context, where the routing prefix, subnet ID, and interface identifier are "RP2a", "SID2", and "IID2", respectively. The context ID of the host's local context is CID1, whereas the context ID of the remote context is yet unknown.

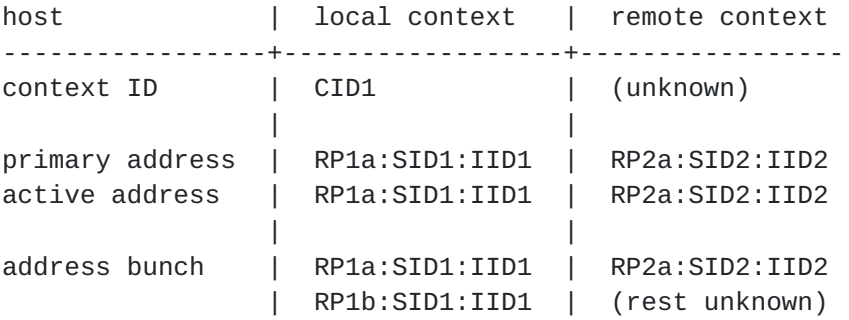

Figure 3: Host's contexts on session establishment

The primary local and remote addresses are used as the source and

destination addresses, respectively, in the first packet that the host sends to the correspondent host. This first packet carries an IPv6 Destination Options extension header with a new "Context Setup option" that includes the parameters of the host's local context.

3.4. Protecting Address Bunches

The ability to send and receive packets via an active address that differs from the primary address visible for protocols above Six/One must be protected against misuse. Failure to provide appropriate security measures would enable a malicious host to communicate with a correspondent host on behalf of a victim host. Such an "impersonation attack" calls for the malicious host to register with the correspondent host an address bunch that includes the victim host's address as primary, and an address at which the malicious node itself is reachable as active. The malicious host could then exchange packets with the correspondent host via the active address, whereas protocols above Six/One on the correspondent host would see the primary address and hence assume to be communicating with the victim host. The interface identifier of the active address would have to be the same as that of the primary, that is, the victim host's address. But a malicious host can easily configure an appropriate active address if the malicious host resides on an access link that permits stateless address auto-configuration. Even with other address configuration means, it may still be possible for the malicious host to send packets from a spoofed address with the desired interface identifier, and to overhear packets delivered to that same address.

To prevent impersonation attacks, Six/One requires a host to prove to its correspondent host that it is the legitimate owner of its primary address. This address ownership proof is provided through a cryptographic binding between the subnet prefixes of the addresses in the host's bunch and the common interface identifier of these addresses. The host creates the cryptographic binding when it generates an address bunch, and the correspondent host verifies the binding during context establishment. This makes it infeasible for a malicious host to create an address bunch that contains both a victim host's address and an address at which it is reachable itself.

The technique Six/One uses to create the cryptographic binding between the different subnet prefixes and the common interface identifier in an address bunch is based on the generation and verification algorithms for Cryptographically Generated Addresses [2] and Hash-Based Addresses [3]. The technique differs, however, in that it yields a single interface identifier for all addresses in a bunch, although these addresses have different subnet prefixes. A

host generates an address bunch according to Section 6 in [3] with the exception of steps 6.1 and 6.5, which are modified such that the CGA parameters no longer emphasize a particular subnet prefix:

6.1. Concatenate from left to right the final Modifier value, 64 zero bits, the collision count, the encoded public key or the encoded Extended Modifier (if no public key was provided) and the Multi-Prefix extension. Execute the SHA-1 algorithm on the concatenation. Take the 64 leftmost bits of the SHA-1 hash value. The result is Hash1[i].

6.5. Form the CGA Parameters data structure that corresponds to HBA[i] by concatenating from left to right the final modifier value, Prefix[i], the final collision count value, the encoded public key or the encoded Extended Modifier and the Multi-Prefix extension.

Six/One further requires a correspondent host to verify the cryptographic binding between the different subnet prefixes and the common interface identifier in an address bunch according to Section 7 in [3] with the exceptions that step 2.2 is omitted, and that step 1 is modified to no longer emphasize a particular subnet prefix:

1. Verify that the 64-bit HBA prefix is included in the prefix set of the Multi-Prefix extension. If it is not included, the verification fails. Otherwise, fill the Subnet Prefix field of the CGA Parameter data structure with 64 zero bits.

It should be emphasized that, although the above modifications effectively eliminate the Subnet Prefix field in a CGA Parameters data structure, the Multi-Prefix extension of the data structure still lists the set of subnet prefixes of an address bunch. This upholds the cryptographic binding between the subnet prefix and the interface identifier for each address in the bunch.

The described cryptographic binding between the subnet prefixes and the interface identifier in an address bunch is the default address ownership proof if Six/One is used without a host mobility, host multi-homing, or host identity protocol, such as Mobile IPv6 or the Host Identity Protocol. However, if Six/One is combined with one of these protocols, that protocol effectively provides the primary address, and the address ownership proof performed by that protocol can replace the default address ownership proof of Six/One. For example, if Six/One is combined with the Host Identity Protocol, a host's primary address becomes a host identity tag, and the proof of ownership of the host identity tag is accomplished through IPsec-based authentication.

3.5. Responding to a Session Initiation

When a correspondent host receives from a host a packet that includes an IPv6 Destination Options extension header with a Context Setup option, the correspondent host allocates a remote context for the host based on the parameters in the Context Setup option. This may be a new context or a reused one. The active local and remote addresses in the context are set to the source and destination addresses in the received packet. Since the subnet prefixes of the addresses in this first packet may already have been rewritten -- either by Six/One on the host prior to packet transmission, or later on in the network --, the active local and remote addresses may differ from the corresponding primary addresses.

The correspondent host further allocates a local context for the communication session based on the address bunch that includes the destination address of the incoming packet. The local context may again be new or reused, and it uses the same active addresses as the remote context for this session.

Figure 4 continues the example of Figure 3 with an illustration of the contexts that the correspondent host has at this point. The figures differ in that the roles of the local and remote contexts are reversed, and in that both contexts are now complete. The correspondent host's local context has context ID "CID2". And besides the primary address RP2a:SID2:IID2, its address bunch includes two more addresses, RP2b:SID2:IID2 and RP2c:SID2:IID2, where "RP2b" and "RP2c" are additional routing prefixes. In this example, the routing prefix of the packet's source address was rewritten in the network because the active address in the correspondent host's remote context is RP1b:SID1:IID1, whereas the active address on the host's local context used to be RP1a:SID1:IID1 at the time the packet was sent.

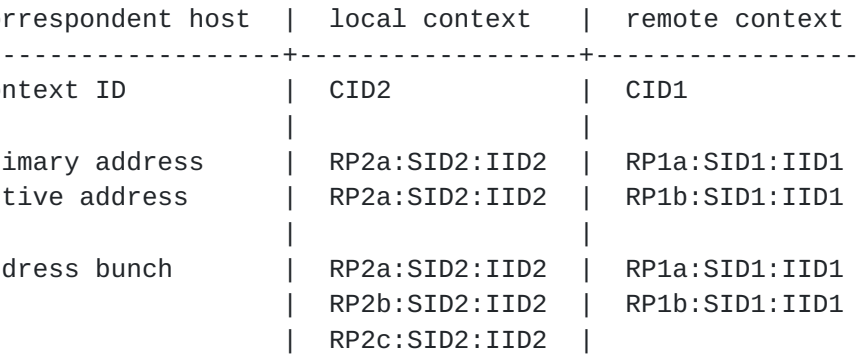

Figure 4: Correspondent host's contexts on session establishment

The correspondent host typically returns a packet to the host as a response to the packet received. This packet is used to communicate the parameters of the peer's local context back to the host, again with a Context Setup option in an IPv6 Destination Options extension header.

3.6. Completing a Session Initiation

The host completes the preliminary remote context for the correspondent host when it receives the first packet from the correspondent host that includes an IPv6 Destination Options extension header with a Context Setup option. Both the host and the correspondent host have then complete local and remote contexts for the communication session. As with all subsequent packets received from the correspondent host, the host also resets the active addresses in its local and remote contexts to the received packet's destination and source addresses, respectively, so as to adapt to any address rewrites.

Figure 5 illustrates the local and remote contexts at the host at this point, concluding the example from the previous figures. Since the correspondent host has sent the packet to the active address in its remote context, which is RP1b:SID1:IID1, the host has changed the active address in its local context to RP1b:SID1:IID1 accordingly.

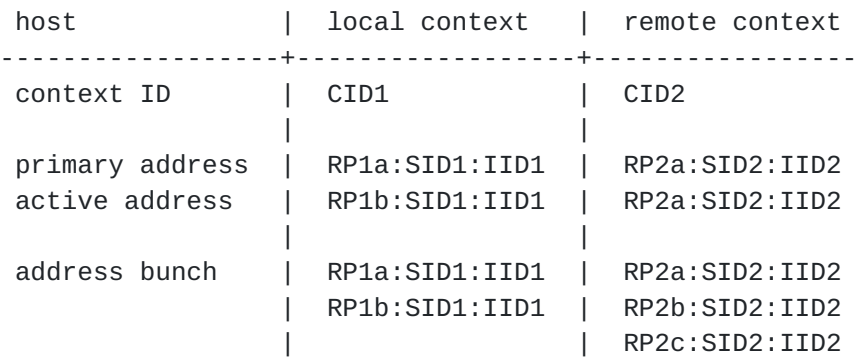

Figure 5: Host's contexts for established session

3.7. Handling Outgoing Packets

When a host has a packet to send that was delivered by a protocol above Six/One, the host may have to rewrite the routing prefixes of the packet's source and destination addresses so as to direct the packet via a provider that was recently chosen by the network. The host must further include sufficient information in the packet to

enable the correspondent host to reverse these address changes before handing the packet to protocols above Six/One at the receiving side.

The source and destination addresses that the host is supposed to use when sending a packet are, respectively, recorded in the active addresses of the local and remote contexts of the communication session. As delivered by the protocol above Six/One, the source address in the packet contains the primary local address. The host replaces this with the active address from the local context. Moreover, the packet's destination address contains the primary remote address when the packet is received from the protocol above Six/One. This is replaced with the active address from the remote context.

To enable the correspondent host to map the packet onto the correct local and remote contexts, all packets except the first in a new communication session are endowed with the sending host's local and remote context IDs for the session. This information is carried in a "Context ID option" of an IPv6 Destination Options extension header.

3.8. Handling Incoming Packets

A correspondent host that receives a packet from the network uses the Context ID option in the packet's IPv6 Destination Options extension header to retrieve the correct local and remote contexts. The local context is obtained based on only the local context ID given in the option. It is used to replace the destination address in the received packet with the primary local address. The remote context is obtained based on the combination of the remote context ID given in the option and the packet's source address. It is used to replace the packet's source address with the primary remote address.

The correspondent host must further memorize which source and destination addresses the packet arrived with, so that the same local and remote addresses can be used for packets that it sends subsequently in the same communication session. This is accomplished by resetting the active addresses in the local and remote contexts to the packet's destination address and source address, respectively.

3.9. Network-Side Provider Selection and Address Rewriting

It is up to the edge network to decide via which provider an egress packet shall be routed. One possibility is for the edge network to accept the provider that is indicated by the routing prefix of the egress packet's source address. The provider is then effectively selected by the host sending the packet. On the other hand, traffic

engineering strategies or other local policies may require the edge network to route packets via a different provider than implied by their source addresses. The edge network can then preempt the provider selection of a sending host.

In case an egress packet is routed via a different provider than the one implied by the packet's source address, the subnet prefix of the source address must be rewritten to one with a routing prefix from the provider chosen by the edge network. Besides ensuring topological correctness, this makes egress and ingress packets of a particular communication session go via the same provider, which is important to support efficient provider fail-over. Network-side address rewrites do not effect a packet's destination address, nor any addresses that appear outside the packet's IPv6 header. Like address rewrites that are pursued by Six/One on a sending host, network-side address rewrites are reversed by Six/One on the receiving host.

Without loss of generality, it can be assumed that provider selection and address rewrites be accomplished by routers. Both can happen anywhere in the edge network, like directly on a first-hop router, somewhere inside the edge network, or at the border to the edge network's providers. Address rewriting may furthermore be delegated to a provider.

In rewriting the subnet prefix of an address, a router must ensure that the old and the new address belong to the same host. The router must hence in general know the set of subnet prefixes that are valid on the link from which a packet originates. This requirement can be eliminated through a wise assignment of subnet prefixes to individual links, as described in Section 4.1 and Section 4.2.

3.10. Session Shutdown

Contexts are soft-state and do not explicitly need to be revoked. Hosts keep both remote and local contexts as long as there are protocols above Six/One that use these contexts. A context is kept for a while also after all protocols above Six/One have stopped using it since the context might be picked up by a new communciation session shortly thereafter. Contexts are finally disposed after a predefined idle time, where the idle time for local contexts should be a bit longer than the idle time for remote contexts. A remote context reused by a host is therefore very likely to still exist as a local context at the correspondent host even after a communication pause.

4. Recommended Access Network Practices

Since hosts configure and select addresses without considering the structure of a subnet prefix, edge network operators have the freedom to constuct subnet prefixes in arbitrary manner. On the other hand, a clean separation between routing prefixes and subnet IDs can reduce the cost of edge network rehoming substantially, as explained in this section.

4.1. Subnet Prefix Assignment

The recommended practice for assigning subnet prefixes to links in an edge network is to give each link a single subnet ID that is unique across the edge network. A link's subnet ID is combined with a each of the edge network's routing prefixes to form the set of subnet prefixes for the link. Within the scope of the edge network, a link can then be identified based on its subnet ID alone, and the routing prefix attached to the subnet ID does not have to be considered.

4.2. Router Configuration

For a router to inform hosts and other routers about the subnet prefixes that are valid on a link it attaches to, the router must learn these subnet prefixes up front. Nowadays, routers are typically configured with entire subnet prefixes. This practice requires costly reconfiguration of subnet prefixes in each individual router when the edge network rehomes. The reconfiguration overhead can be reduced substantially if edge network operators follow the recommended practice for subnet prefix assignment described in Section 4.1, and configure routers with the subnet ID of each link they attach to and, separately, a list of the routing prefixes shared by all links in the edge network. A router then builds the set of subnet prefixes for a link automatically by concatenating each of the routing prefixes with the subnet ID for this link.

In the event of edge network rehoming, the list of routing prefixes gets updated, but the subnet ID of each link remains unchanged. It is then sufficient to distribute a new list of routing prefixes across all routers in the edge network. Since this list is the same for all links in the edge network, router reconfiguration becomes simpler than it would be if entire subnet prefixes were to be replaced. Once a router has automatically reconstructed the subnet prefixes of a link that it attaches to, it starts advertising the new subnet prefixes to hosts on the link, and announces them via the edge-network-internal routing protocol. This causes hosts to automatically reconfigure their network protocol stacks and updates

the routing table in other routers.

The reconfiguration process during rehoming can be further simplified by using only the link-local subnet prefix, or unique local [8] subnet prefixes on links to which no hosts attach. These subnet prefixes include provider-independent routing prefixes that do not change during edge network rehoming.

The recommended subnet prefix assignment practice also reduces the cost of configuration and rehoming-related reconfiguration of routers that are responsible for rewriting the addresses in packets. These routers must generally know the set of subnet prefixes that are valid on the link from which a packet was sent, so as to ensure that a rewritten address belongs to the same address bunch (and hence to the same host) as the original address. Replacing the routing prefix alone may in general not be sufficient since subnet prefixes of the same link may differ also in their subnet IDs. (The IPv6 addressing architecture [9] leaves edge network operators the freedom to assign subnet prefixes of arbitrary structure to their links.) On the other hand, if edge network operators follow the recommended subnet prefix assignment practice, all subnet prefixes on a link use the same subnet ID. Address rewriting then does affect only a routing prefix, and it can be accomplished without knowledge of the subnet prefixes in use on the link from which a packet originates.

4.3. Host Configuration

It is in some cases desired to manually preconfigure a host with an address for itself or for certain correspondent hosts. As an example, a simple application might hard-code the address of a server that it needs to contact, or a network management script may included hard-coded addresses for test or debugging purposes. Such address preconfiguration increases the costs of edge network rehoming because each of the preconfigured addresses must then be changed. This is oftentimes a cumbersome manual process.

With Six/One, host reconfiguration due to rehoming can be eliminated by using unique local addresses for address preconfiguration. Unique local addresses have a randomized, provider-independent routing prefix that the edge network operator chooses autonomously. Hosts in the same edge network can use unique local addresses -- as both source and destination addresses -- to communicate with each other. Applications on the host can furthermore use the unique local address as a primary local address when communicating with a correspondent host in a different edge network. The unique local address then always gets rewritten to an address in the host's address bunch, which the host has automatically configured based on the subnet

prefixes of the link it attaches to. To enable the correspondent host to verify that the unique local address and the address bunch belong together, the preconfigured address can be included in the computation of the interface identifier.

By definition, two hosts in the same edge network never use the same unique local address. Chances that the unique local addresses of hosts in different edge networks collide are negligible given the randomized selection of routing prefixes for these addresses.

It would theoretically be possible to preconfigure a host also with the unique local address of a correspondent host in a different edge network. The unique local subnet prefix of the other edge network would then have to be known in the host's edge network, and a router on the path of the host's packets would have to rewrite the correspondent host's unique local address to an address with the other edge network's provider-dependent routing prefixes. However, for simplicity, it is recommended here that hosts obtain the addresses of correspondent hosts in other edge networks through DNS.

4.4. Configuration of Traffic Control and Analysis Equipment

Networking equipment for traffic control and analysis, such as a firewall or an intrusion detection system, generally uses the addresses in intercepted packets to identify the sending and receiving hosts. This is based on the assumption that a host's address remains stable throughout the duration of a communication session. For the networking equipment to function correctly in the presence of Six/One, hosts must be identified in a way that allows the subnet prefixes in a host's address to change during a session.

Unique identification of hosts that reside in the same edge network as the piece of networking equipment under consideration is straightforward if edge network operators follow the recommended practice for subnet prefix assignment described in Section 4.1. All subnet prefixes of a particular link then have the same subnet ID, and the addresses in a host's address bunch differ only in their routing prefixes. Networking equipment can thus uniquely identify a host in the local edge network based on only the subnet ID and the interface identifier of the host's addresses.

In practice, networking equipment may be preconfigured with a "routing mask", indicating the common length of the routing prefixes that the edge network has obtained from its different providers. Assuming the routing prefix length is L, the routing mask can be thought of as a string of 128 bits of which the first L bits are "1" and the last 128-L bits are "0". A bit-wise And operation between a

host's address and the inverted routing mask then yields a bit string that uniquely identifies the host across Six/One-related address changes.

Since networking equipment in one edge network has in general no knowledge on the structure of subnet prefixes used in remote edge networks, it must identify correspondent hosts in remote edge networks differently than hosts in the local edge network. Even if the operator of a remote edge network also followed the subnet prefix assignment practice described in Section 4.1, there could still be a link in a third edge network with the same subnet ID and a different routing prefix. Networking equipment could hence confuse correspondent hosts in different remote edge networks if it identified those correspondent hosts based on only their subnet IDs and interface identifiers. On the other hand, since a host in the local edge network selects a separate context identifier for each correspondent host it communicates with, a correspondent host can be uniquely identified based on this context identifier in combination with the subnet ID and interface identifier in the address of the host in the local edge network.

Figure 6 illustrates the operation of a firewall that identifies hosts as described above. The figure shows an edge network that multi-homes with two providers. Based on an estimated need to address hosts on up to 2^16 links, the edge network operator has requested one 48-bit routing prefix from each provider. Provider P1 has assigned routing prefix PP1::/48 to the edge network, and provider P2 has assigned PP2::/48. Accordingly, the edge network's routing mask amounts to FF:FF:FF:FF:FF:FF::/48. The remaining 16 bits inside 64-bit subnet prefixes are then used to form subnet IDs. The single link visible in Figure 6 is assigned subnet ID SID. This in combination with the two routing prefixes obtained by the edge network yields two subnet prefixes, PP1:SID::/64 and PP2:SID::/64, that are to be advertised by the router shown in the figure. Figure 6 furthermore shows two hosts, X and Y, that attach to the same link as the router. The interface identifiers that the hosts use to configure address bunches are IIDx and IIDy, respectively. Since the interface identifier is the same for all addresses in a bunch, and so is the subnet ID, hosts X and Y can be identified by the address postfixes ::SID:IIDx and ::SID:IIDy, respectively. This is exactly what the firewall inside the edge network does for host X, which at that point is communicating with correspondent host Z. The firewall furthermore identifies correspondent host Z based on the context identifier CIDx, which host X uses for the communication session with host Z, and host X's address postfix ::SID:IIDx. The firewall hence continues to function correctly when Six/One causes either of the hosts to switch to a different address in its respective bunch. There is also no need to reconfigure the firewall

when the edge network rehomes.

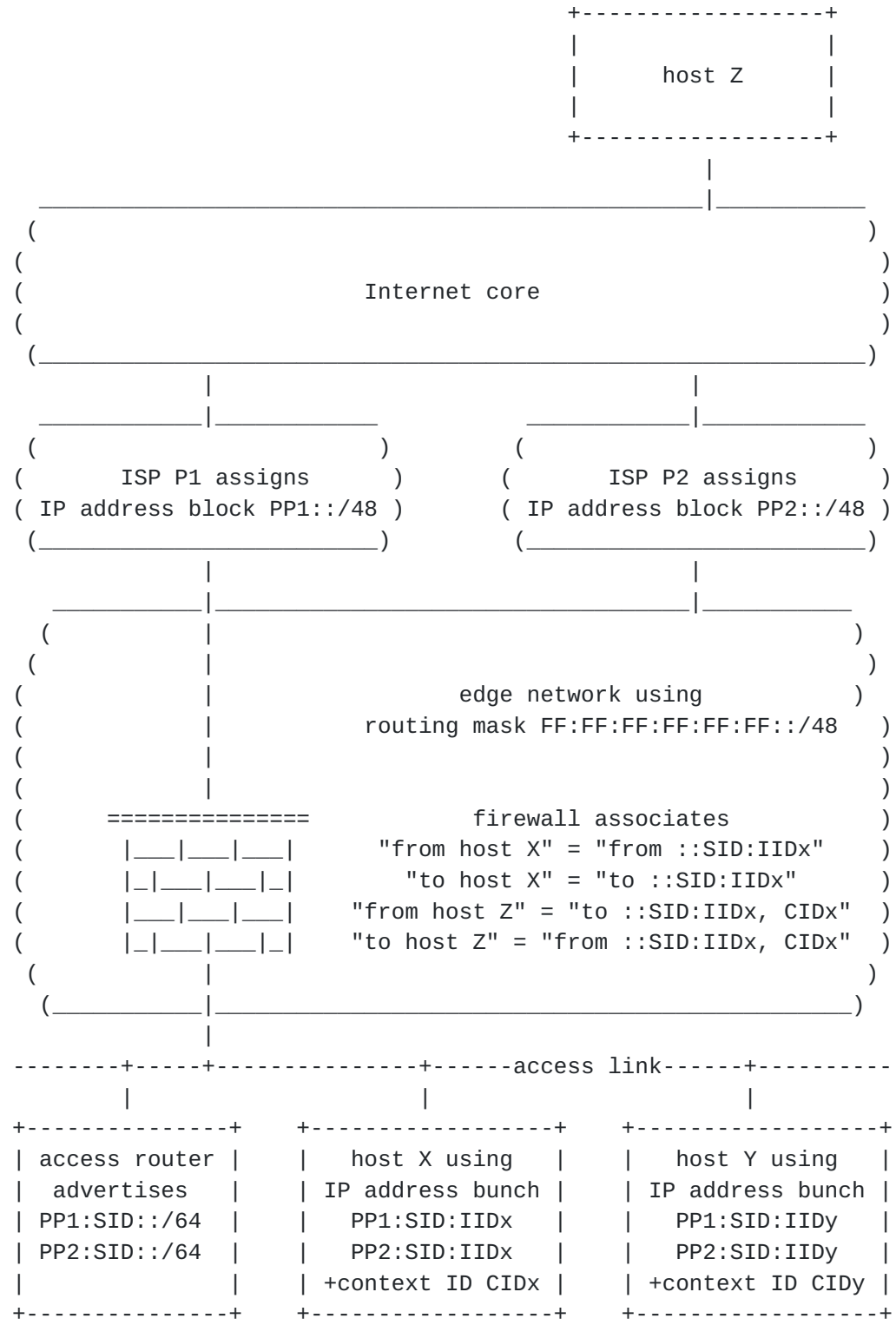

Figure 6: Operation of a firewall in the presence of Six/One

5. Discussion

This section attends to the feasibility of Six/One as a universal solution for the routing and addressing problem. It discusses the incentives Internet players have to deploy and use Six/One, and it describes a transition plan for moving the current Internet towards a Six/One-capable Internet. The section furthermore explains how Six/One could help bringing forward a universal solution for source address validation.

5.1. Incentives for Deployment

The success of a solution for the routing and addressing problem hinges on its acceptance by the three Internet players -- end users, edge network operators, and providers. Acceptance, in turn, depends on the degree to which a solution satisfies the objectives of a player relative to the costs that a deployment entails for the player.

In the case of Six/One, the main hurdle for universal deployment is a widespread support in hosts. At first glance, it may seem difficult to enforce such widespread support, in particular since a significant part of Six/One's functionality is for the benefit of edge network operators and does not directly benefit the hosts themselves. On the other hand, end users are oftentimes not directly involved in host upgrades. Mobile phones or Internet-ready entertainment equipment typically have a rather short lifetime, which makes a medium-term introduction of new networking protocols well feasible. Major operating systems for personal computers incorporate highly automated upgrade routines, which allows for even short-term introduction of new software. The introduction of new host-based networking technology may hence turn out to be more facile than the introduction of technology that requires technical, and oftentimes also administrative, co-operation of edge network operators or providers.

Edge network operators are likely to benefit most from Six/One. It is therefore legitimate to expect them to willingly adopt the practices recommended in Section 4.

Providers are expected to embrace the introduction of Six/One because the sole use of provider-dependent addressing space inside the transit domain will facilitate a smaller routing table size and less frequent routing table updates. This benefit is considered paramount, and therefore likely to compensate for the small cost of slightly higher packet sizes, which Six/One incurs due to in-band signaling for context setup and identification. Barring the increased packet size, providers remain unaffected by an introduction

of Six/One. A provider may at most agree to perform address rewriting on behalf of an edge network. Such a co-operation would take place as part of a business relationship between the provider and the edge network, and would hence be driven by economic objectives.

5.2. Transition

Transition towards a widespread deployment of Six/One could proceed in two phases.

1. In the first phase, support for Six/One will be introduced into hosts, and edge network operators will gradually adopt the practices recommended in Section 4. To avoid disruptions of communication sessions with legacy correspondent hosts, hosts with support for Six/One rewrite an address only after a received Context Setup option in an IPv6 Destination Options extension header indicates that also the correspondent host supports Six/One. For the same reason, routers refrain from rewriting an address in packets that do not contain a Context ID option in an IPv6 Destination Options extension header. The Context ID option, too, indicates bilateral Six/One support since it is used only by Six/One hosts that have received a Context Setup option from the correspondent host.

2. The second phase begins when there is reasonably widespread Six/One support in hosts. Both host and edge networks can now begin to initiate address rewrites.

To aid transition, two operational modes should be differentiated for Six/One software on hosts. In "conservative mode", Six/One will only adapt to address rewrites that were performed in the network or by the instance of Six/One on a correspondent host. Six/One software running in conservative mode should also provide support for legacy correspondent hosts, which do not understand Context Setup and Context ID options in an IPv6 Destination Options extension header.

More sophisticated Six/One software may further support a "progressive mode", in which address rewrites may also be initiated by the Six/One instance on the host itself. While conservative mode provides the necessary functionality to adapt to network-side address rewrites, progressive mode additionally enables a host to replace the source address selected by an application with another address that corresponds to a different provider. Progressive mode is useful when a Six/One instance is aware of alternative providers and the quality of service these providers deliver. For example, if a provider selected at application level has recently performed poorly or become defunct, Six/One may autonomously rewrite the source address in

packets received from the application in order to suggest to the edge network that the packets be routed via a different provider. During the first transition phase, Six/One software on hosts should operate in conservative mode only, since Six/One support on a correspondent host can then not necessarily be expected. Six/One software that supports progressive mode may be switched to that mode when the second transition phase begins.

End users benefit from Six/One in that the protocol enables their hosts to suggest packets to be routed via a preferred provider. End users can hence be expected not to hinder Six/One-related host upgrades. Host upgrades will furthermore occur mostly transparently to the end users, either through replacement of old networking equipment, or through automated software upgrades. The introduction of Six/One software in the first transition phase will hence occur rather smoothly. Adoption of the recommended practices in edge networks will be driven by the prospects of reduced network reconfiguration costs during rehoming, which apply independently of Six/One support on hosts or the practices of other edge network operators. Multi-homed edge network will further be motivated by the prospective ability to rewrite addresses as of the second transition phase, since this ability will accommodate their traffic engineering strategies.

5.3. Easing Universal Source Address Validation

Protection against IP spoofing has until today never become universal. The reason for this is that available source address validation techniques such as ingress filtering are missing deployment incentives for edge network operators or providers. Six/One can help in this regard because it enables routers to rewrite subnet prefixes in packets' source addresses, thus automatically fixing subnet prefixes that are topologically incorrect.

In a typical deployment of Six/One, border routers -- either inside the edge network or in the providers's network -- ensure that the source addresses in packets leaving the edge network have the routing prefix of the provider they are going through. (Failure to do so would prevent return traffic to go via the same provider and hence defeat efficient provider fail-over.) An efficient way of implementing a routing prefix check would be for a border router to simply overwrite the routing prefix in the source address of every packet leaving the edge network. Access routers then only need to verify the subnet ID in the packets they forward. This is a simple task given that there is only a single, stable subnet ID per link. In contrast to ingress filtering today, this task would not require reconfiguration when the edge network rehomes.

6. Security Considerations

This section addresses potential security threats for Six/One, and it describes how Six/One protects against these threats.

- Impersonation. The default address bunch protection mechanism described in Section 3.4 mitigates the threat of impersonation attacks. Alternatively, if Six/One is combined with a host mobility, host multi-homing, or host identity protocol, such as Mobile IPv6 or the Host Identity Protocol, the address ownership proof performed by that protocol can replace the default address ownership proof of Six/One, as described in Section 3.4.

- Inverse impersonation. In an inverse impersonation attack, an attacker tricks a correspondent host into believing that it is communicating with the attacker, while in fact it communicates with a victim host. This attack would require the attacker to build an address bunch that includes the victim host's address. Such fraud that is prevented by the default address bunch protection mechanism described in Section 3.4.

- Redirection-based flooding. The protection against impersonation described in Section 3.4 mitigates attacks against a specific victim host because an attacker cannot easily construct an address bunch that includes the victim host's address. Redirection-based flooding attacks against networks could be protected against through reachability verification -- as it is done in Shim6 or Mobile IPv6, for example. On the other hand, redirection-based flooding attacks require IP spoofing. So given the fact that Six/One eases universal source address validation (see Section 5.3), extra signaling for flooding prevention might actually no longer be necessary.

7. IANA Considerations

This document has no actions for IANA.

8. Acknowledgment

The author would like to thank Jari Arkko, Pekka Nikander, Steven Blake, Lars Westberg, Mikko Sarela, Mark Doll, Lixia Zhang, Tony Li, Geoff Huston, Brian E. Carpenter, Marcelo Bagnulo, Erik Nordmark, Lars Eggert, Pekka Savola, Magnus Westerlund, Jonne Soininen, Loa Andersson, David Ward, John Scudder, Gert Doering, Olaf Kolkman, Stig Venaas, Thomas C. Schmidt, Kotikalapudi Sriram, Jordi Palet, Andras Csaszar, Jan M. Melen, Petri Jokela, Patrik Salmela, Borje Ohlman, Anders Eriksson, and Attila Mihaly for valuable feedback on the solution to the routing and addressing problem presented in this document, and for interesting discussions on the routing and addressing problem as such.

This document was generated using the XML2RFC tool.

9. References

9.1. Normative References

[1] Bradner, S., "Key Words for Use in RFCs to Indicate Requirement Levels", IETF BCP 14, RFC 2119, March 1997.

[2] Aura, T., "Cryptographically Generated Addresses (CGA)", IETF Request for Comments 3972, March 2005.

[3] Bagnulo, M., "Hash Based Addresses (HBA)", IETF Internet draft draft-ietf-shim6-hba-03.txt (work in progress), June 2007.

9.2. Informative References

[4] Nikander, P., "Generic Proxying as a Deployment Tool (GEPROD)", IETF Internet draft draft-nikander-ram-generix-proxying-00.txt (work in progress), January 2007.

[5] Nordmark, E. and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6", IETF Internet draft draft-ietf-shim6-proto-08.txt (work in progress), May 2007.

[6] O'Dell, M., "8+8 - An Alternate Addressing Architecture for IPv6", IETF Internet draft draft-odell-8+8-00.txt (work in progress), October 1996.

[7] Schuetz, S., Winter, R., Burness, L., Eardley, P., and B. Ahlgren, "Node Identity Internetworking Architecture", IETF Internet draft draft-schuetz-nid-arch-00.txt (work in progress), September 2007.

[8] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", IETF RFC 4193, October 2005.

[9] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", IETF RFC 4191, February 2006.

Appendix A. Change Log

The following is a list of technical changes that were made from version 00 to version 01 of the document. Editorial revisions are not explicitly mentioned.

- Section 4.3 -- Clarified support for preconfigured addresses in applications.

- Section 5.2 -- Enhanced backward-compatibility: Hosts and routers rewrite addresses only after both end hosts are known to support Six/One.

Author's Address

Christian Vogt
Ericsson Research, NomadicLab
Hirsalantie 11
02420 Jorvas
Finland

Email: christian.vogt@ericsson.com

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).