IPv6 Site Multihoming Using a Host-based Shim Layer

Pekka Savola
CSC/FUNET, Finland
psavola@funet.fi

Abstract

Site multihoming is the process of an end-site, such as an enterprise, to obtain simultaneous IP connectivity from multiple ISPs, done for a number of reasons, such as increased resilience against failures. A new IETF working group was chartered at the end of June 2005 to work on designing an IPv6 site multihoming solution.

A prevailing approach, called "shim6", inserts a shim layer between the IP and transport layer. Sites have to deploy multiple provider-assigned IP address prefixes from multiple ISPs on hosts, instead of a single provider-independent prefix. These IP addresses ("locators") are used by applications and if a session becomes inoperational, shim6 can switch to using a different address pair. The switch is transparent to applications as the shim layer rewrites and restores the addresses at the sending and receiving host. The solution has been designed with ease of deployment and transition in mind, and changes in typical applications are not required.

We describe and analyze this solution, and as the design is still in progress so we note and analyze many open issues.

1. Introduction

Site multihoming is the process of an end-site, such as an enterprise, to obtain simultaneous IP connectivity from multiple ISPs. This is done for a number of reasons, such as increased resiliency against failures (see more of the motivations in [24]).

Site multihoming with IPv4 is typically achieved by routing (with Border Gateway Protocol, BGP) or using Network Address Translator -based mechanisms [24]. While using BGP might scale up to a certain extent, it is an unsuitable mechanism for every site's needs particularly because it requires that every site is visible in the global routing table [23, 24].

IPv6 address allocation policies and BGP advertisement prefix length filters have so far restricted the allocation of address space in such a manner that BGP-based IPv6 site multihoming has not been feasible. The hope has been to avoid creating a similar routing table "swamp" as exists today with IPv4 to maintain future scalability.

At the same time, IETF IPv6 Site Multihoming (multi6) working group [14] has explored the design requirements for the multihoming solution [1], threats [22] and architectures at depth. In late 2004, consensus emerged that a particular solution, called shim6, should be developed in a new dedicated working group.

The multihoming solution proposals have been surveyed and compared in previous studies (e.g., [24]). That paper also provides background to the multihoming problem space. For the sake of brevity, we do not repeat that here.

In this paper we provide a summary, analysis, and evaluation of the current proposal. As the IETF shim6 working group was chartered at the end of June 2005, this paper is very timely in bringing the reader up to speed with the latest developments in this area.

Unless stated otherwise, the paper represents only the author's personal working group member point of view.

2. Structure of the Paper

We assume prerequisite knowledge of the identifier/locator split concept (e.g., [19, 7]) and the background and motivation to IPv6 site multihoming proposals (e.g., [24]). In the sake of brevity and avoiding duplication of work, we do not repeat or summarize that work here.

This paper is organized as follows.

The first section very briefly describes and refers to the background of IPv6 site multihoming work. This section describes the prerequisites for understanding this paper, and outlines the structure of this paper. The third section describes the generic issues with hosts having multiple addresses from different providers, in order to gain non-shim6 specific background to the problem space. The fourth section describes the summary of the shim6 proposal. The fifth section provides explicit analysis and discussion on a number of larger issues. The sixth section provides conclusions.

The focus of this paper is providing a concatenated overview of the proposal, and discussing and analyzing the

most interesting and relevant design choices and options.

While the paper should be read in sequence, an impatient reader may start by taking a quick overview of the protocol in Section 4.

3. Multiple Addresses on Each Host

The shim6 proposal assumes that multihomed sites obtain multiple IP address prefixes, one from each ISP the site has connectivity to. These IP addresses are then deployed on each node which should be capable of multihoming.

This has a number of important implications which we need to discuss briefly first. When a failure occurs at a multihomed site,

1. the site should be able to initiate new sessions (internal or external),
2. hosts in the Internet should be able to initiate new sessions to the servers and hosts at the site (if any), and
3. the existing sessions (internal or external) should continue to work without disruption ("session survivability").

Ingress filtering restricts the set of addresses that can be used for new or established sessions. First in Section 3.1 we describe these prerequisites, and then look at source address selection in Section 3.2. The third goal is fulfilled by session survivability, which we discuss in Section 3.3.

3.1. Ingress Filtering

The host must choose the source and destination addresses properly and the site's border routers must forward the packets appropriately to pass the ISPs' ingress filtering. That is, packets with the source address from ISP A's aggregate prefix must be forwarded on the link to ISP A, and similarly for ISP B's prefix [6]. This implies source-address based policy routing (with a very simple policy) at the border routers, where all the border routers must connected either physically or through a tunnel. [12, 24]

When an ISP, a link to an ISP, a border router or some other network component fails, the prefix assigned from that ISP's aggregate route typically ceases to work, because the only path where the prefix would pass ingress filtering just became unusable. This problem can be worked around by using tunnels to build backup connectivity to each ISP [10], which also would obviate the address selection and session survivability issues.

3.2. Address Selection

For hosts at a multihomed site to be able to initiate new sessions after a failure, the hosts need to be able to select a source address which works between the source and the destination. For a typical external failure, it is enough to pick the source address from the working ISP's prefix. More complex failure scenarios, e.g., where some destinations work only through one ISP and others via another, would require more fine-grained prefix selection methods.

For hosts in the Internet to be able to initiate new sessions to the hosts at the no-longer-fully-multihomed site, (1) they have to find a working destination address (typically from DNS), and (2) the responding host needs to be able to pick a working source address in the response packets. [11]

Sometimes (2) is implied by (1): for example, the TCP SYN-ACK source address must be the same as the initiator chose as the destination address; in these cases the address pairs must be working bidirectionally. Many protocols do not have this address selection requirement, and the communication may also work with unidirectionally working address pairs. (See Section 4.5 for more.)

Destination address selection [8] goes through all the IP addresses in a certain order if the application has been developed in a proper manner; almost all the IPv6-capable applications do so, due to having to support both IPv4 and IPv6 [25].

So, the specific requirements are:

- Destination address selection needs to be quick and reliable in cycling through all the addresses, and
- Source address selection must try multiple addresses, instead of using just one.

Unfortunately, the fallback to the next address is not necessarily quick or reliable. Network elements may end up discarding packets without sending any indication to the affected hosts that the particular address (through this path) does not work. In fact, this is rather common – we observe that ingress filtering typically does not send packets because failing packets assumably have a forged source address and doing so wouldn't help. Further, ISP's aggregate prefix is often installed as a discard route, and when a more specific site prefix goes missing, the packets are just silently discarded. Sometimes, however, an ICMP unreachable message is sent, but while the mechanisms exist to take these into considerations, a subset of errors often aren't [9]. So, the application would have to rely on the transport protocol timeouts to notice that the communication did not start properly, and this can take even minutes per tried address [9].

There are proposals to improve source address selection to retry [11], but such a mechanism will also need to deal with the the fact that there may not be any feedback from the network on failing attempts. It is also not clear which component of the stack should perform the retries; presumably this could be done as part of the getaddrinfo() loop for applications which don't bind to a specific address.

Another mitigation technique applicable in most failure modes is trying to withdraw the non-working prefixes from being used as soon as possible; for source address selection, they might be marked Deprecated thus being less preferred; for destination address selection, they might be removed from the DNS. However, we note that these have some obvious issues: marking a source address requires information that it no longer works and updating DNS dynamically for temporary failures might be very impractical (ignoring the operational challenges of DNSSEC key provisioning) and the old data would still persist in the DNS caches for the lifetime of the record's previous TTL.

3.3. Session Survivability

Session survivability is a more difficult problem because the existing protocols such as TCP and UDP can't switch to using different addresses while preserving the session.

Stream Control Transmission Protocol (SCTP) provides this functionality but would have significant deployment hurdles for the generic use; in any case, the IETF multi6 WG decided that the right place to fix this is below the transport layer. [24]

Address selection must also be performed when switching a session to use a new address. The fact that the session has already been successfully established before the failure mitigates the generic selection issues slightly. In particular, we conclude:

- Already having an established session allows preemptively probe or test alternative address pairs, and
- Such testing does not need to be done for short-lived sessions, meaning less packets and bytes sent.

3.3.1 Security of the Session Survivability

Being able to redirect a session between two addresses to use different addresses has significant security threats [22].

Because a global trust infrastructure does not exist, the designs have had to cope with a different trust model. [22] studied how plaintext communications may be disrupted as of today. Currently, if an attacker is on the path between a source and the destination (or attached to the same link at either end), the attacker can typically eavesdrop and usually redirect communications. The security of the shim6 solution must not be worse than that; therefore shim6 does not need to be secure against attackers which are on the path for the entire duration of the attack.

In the examples below, we have hosts A and B, and an attacker X. The main redirection threats are similar to Mobile IPv6 (MIPv6) binding update security, but in general caused by the identifier/locator separation. The threats and some fixes are [22]:

1. The attacker could claim that A's new location is at his address or at an unroutable address; the ownership and reachability of the IP address must be verified first,
2. The attacker can redirect packets if it can be on the path for a while and then move out and continue the attack; there must be an upper limit how long the on-the-path verification is valid, and
3. The attacker on a slow link could subscribe a large transmission from A to himself, then start flooding by redirecting the session to B; one must not use a new locator until its ownership is verified first.

Mobile IPv6 design introduced a periodical return routability test: by sending a nonce to the correspondent node (CN), and being able to show in the further messages that the mobile node has received a reply (with a secret nonce of CN's choosing), the mobile node is able to prove that it has an address or is at least on the path where the secret was exchanged. Sending a similar packet to the CN through Home Agent proves the relationship and ownership of the home address, because otherwise the home agent would not forward the mobile node's packets.

While the MIPv6 security design could address the redirection threats, unfortunately it does not quite work with multihoming. MIPv6 relies on home agent and home address always being reachable – multihoming design cannot assume that. All the prefixes used by the site are equal, and any of them could fail. Periodical return routability tests could guarantee the safe redirection until the maximum return routability lifetime[1] expires (7 minutes), but most multihoming outages last much longer than that. As the return routability would no longer work after the failure, even extending that maximum would not solve the most fundamental long-term failure scenarios and would increase the potential damage of temporary on-the-path attacks.

Finally, to ensure privacy, it is important for the hosts to be able to have multiple identifiers. The protocol itself must also be resistant to denial-of-service attacks, i.e., defer creating state or performing expensive operations until the sender has proved its genuine desire to communicate.

3.4. Conclusions

We draw some conclusions at this point, as these have impact on the shim6 design:

- If tunneling [10] can be used, session survivability and address selection work; typically only ingress filtering

[1] The lifetime ensures that if an attacker manages to be on the same LAN or on the path between the sender and the receiver, but then moves elsewhere, after the maximum lifetime it can no longer redirect communications.

requires configuration at the border routers. However, we do not assume this solution is present.

- The applications and stacks need more robust mechanisms to fall back to the next address, also working when there is no feedback from the network [15]. Source address selection must support retries as well. Otherwise reliable address selection is impossible.

- Mechanisms to remove or deprefer the non-working choices would be very useful as they would mitigate the simplest failure modes of the fallback problem; however, these have certain big issues such as the infeasibility of using DNS for quick updates.

- As a result, address selection (and to a much lesser degree, ingress filtering) is a very difficult problem for a host to solve entirely on its own. However, hosts being able to have "dialogue", possibly in conjunction with session survivability, with other network elements or their correspondents might help them in isolating the problem and doing better selection.

4. A Host-based Shim Layer

The proposal defines a new virtual layer, "shim", at the IP layer, below Fragmentation, Reassembly and IPsec processing (see Figure 1 [21]). When a failure occurs, the IP addresses used by the applications (ULIDs) stay the same, while the shim translates the packets to use different addresses at egress and rewrites them back at ingress (see Figure 2 [21]); the mapping is therefore reversible [21]. Further, as all the state is stored at the fate-sharing endpoints of a session, this approach is fully compliant with the Internet end-to-end principle.

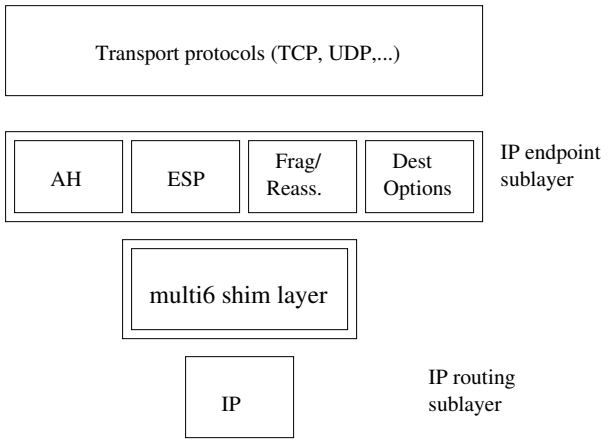

Figure 1. The placement of the shim

Our analysis is that shim6 only provides session survivability. Additionally methods for address selection and ingress filtering management need to be enhanced. The solution does not provide provider independence (and consequently, does not eliminate the need for renumbering) or traffic engineering [24]. This is discussed at more length in Section 5.3.

One of the most important goals of shim6 design has been easy deployability. We discuss this separately in Section 5.5.

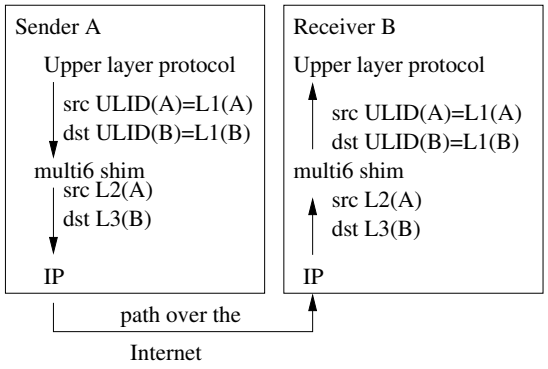

Figure 2. The shim mapping with changed locators

4.1. Interaction with Applications

An important fundamental design choice of shim6 has been that it requires no changes in most applications, even when the session survivability is needed.

Applications use IP addresses as identifiers in many ways: client/server (short- or long-lived), referrals (host A contacts host B, which says to talk to host C), callbacks (Host A contacts B, B sometime later contacts A using the same identifier), or for identity comparison (Host A contacts B, B stores A's identity; later when a host contacts A, B compares the identities to see if they are the same). [20]

There are multiple choices on what the applications could use as identifiers of a session [20]:

- An IP address (like today),
- A special non-routable identifier, from a different name space (like in Host Identity Protocol),
- A hostname or some other identifier string, or
- (Somehow compiled) list of all the possible IP addresses.

The key difference lies in how the application can handle referrals and callbacks. If the identifier is a routable IP

address, these could work just fine, as long as the IP address works. On the other hand, if the identifier comes from a different kind of name space, there would have to be a way to map both ways between locators and identifiers; this is challenging especially if the name space is flat like in HIP [18]. More deployment considerations are listed in Section 5.5.

The use of hostnames or similar rendezvous tags could be beneficial because it would ensure applications know all the locators of a host, thus requiring no mapping functions (in addition to the resolution of names itself) at all. The list of all IP addresses would have similar benefits, although the list would not have temporal flexibility in case the list of locators changes often. However, as these would require application modifications, we do not consider these approaches very attractive in the short term.

Shim6 does not introduce a separate identifier name space, but uses the IP addresses (locators) of the host as upper-layer identifiers (ULIDs). Therefore applications using referrals and callbacks require no up-front modifications prior to shim6 deployment. However, to be able to use such applications in the event of a failure, the nodes may need to find a way to obtain a listing of alternative locators. This requires enhancing the Application Programming Interface (API), and a way to perform this mapping; as IP addresses are used as ULIDs, and they are allocated in a structured manner, obtaining the list using a reverse and forward DNS lookup might be possible (see Section 5.4); there may be other alternatives [20].

A relatively small drawback of using existing locators as ULIDs are issues with mobility and renumbering, as elaborated in Section 5.1.

4.2. Capability Detection and Multihoming Timing

Detecting shim6 capability and establishing the multihoming state (e.g., the information about locators) are very important and somewhat interreleted topics.

The naive approach for detection and establishment would be to insert a special DNS records for ULID or locators; the resolver would first try to look up those, and if successful, start the shim6 negotiation.

While this is necessary for approaches using separate name spaces such as HIP, shim6 gets away with just using the AAAA records because the locators and identifiers are indistinguishable and there is no particular reason to be able to tell them apart.

Therefore there is no need to detect shim6 capabilities or establish any multihoming state prior to starting communications with a node. We explore this issue below.

4.2.1 When to Detect or Establish Multihoming

With shim6, the hosts can start sessions as if they were singlehomed or didn't implement shim6 at all – the shim6 protocol negotiation can happen later in the lifetime of a session, measured based on some policy (for example, number of minutes connected, bytes transferred, etc.).

That is, in most cases it may not be worth the packet exchanges and added overhead to negotiate shim6 capability for all the sessions (including, e.g., quick one round-trip UDP messages) as the chance that a failure occurs during such exchanges and is serious enough is very small.

Being able to delay the set-up of multihoming state thus enables policy control on how aggressively the site wants to protect against failures. We consider this a unique property in the sense all the other protocols for session survivability seem to require up-front negotiation of the state and/or application modifications.

The state needs to be established before failure occurs, though. This is required to ensure security between the initial and additional locators, as described in Section 4.4.

4.3. Establishing the Multihoming State

The multihoming state needs to be exchanged in a secure manner. A four-way handshake allows protecting the receiver against denial-of-service attacks [5].

It is still an open issue how to actually design the exchange – whether as an extension header, destination option in the packets, or using TCP, UDP or some other protocol. The former two could be carried inside the data frames, with the cost of decreasing the MTU for the "piggybacked" packets. In addition to the packet size issues, there are various other concerns, mainly:

- How well the packet can be processed by intermediate nodes, e.g., firewalls (the firewalls may not know the format of the extension header, but the destination options format is predefined), and
- How simple it is to implement and use; destination options can be placed in many places in the IPv6 header chain, and the ordering of options inside the options header is not specified.

The protocol obviously has to exchange the list of additional locators. These locators need to be secured, e.g., using means described in Section 4.4.

There are a couple of interesting open design points about the state exchange:

- Whether to always exchange all the locators or just some (differential vs atomic)? – To avoid a combinatiorial explosion, we believe it is best if the host would never tell more than a couple of locators to a

peer. Therefore the list could be exchanged at once, also avoiding synchronization issues with differential exchanges.

- Does the list need to be periodically refreshed? – We believe that is not necessary, as only the endpoints share the state. If the peer reboots, the multihoming state is not useful anymore in any case.

- How does one close the multihoming state (inform the peer vs quietly)? – While quiet removal might allow the peer to relinquish some state (e.g., flow label reservations), there does not seem to be a particular need to ensure that the state should be explicitly dismantled.

We summarize the different data that may need to be passed in the multihoming protocol in Table 1.

Table 1. Summary of Multihoming State

State	Size	Section
List of host's locators	at least 32 bytes	4.3
Security data structure	at least 100 bytes	4.4
HBA Signature	at least 40-50 bytes	4.4
Public key (for CGA)	100-500 bytes	4.4
Address pair pref.	1 byte	4.5
Context tag (if needed)	4-6 bytes?	4.6
Flow label to be used[2]	4 bytes	4.6.1

4.4. Secure Locator Exchange

In Section 3.3.1 we described generic threats with multiple locators and why just using return routability is not sufficient to obtain security of locator changes. Here we describe the security model, the security building blocks, and how to secure the locator exchange.

4.4.1 Authorization without Identification

Locator changes require protecting against identity spoofing. Specifically, the host must be able to ensure that if it starts communicating with a peer (say, "A"), only peer A is allowed to add or remove locators for the peer.

On the other hand, the host is explicitly not required to strongly or even weakly identify peer A prior to starting the communication. This intentional lack of authentication is not a completely new paradigm, and has been employed in SSH host key databases and "Better-than-nothing Security" model as well [13].

This has the major benefit of not requiring public key infrastructure, but exposes the hosts to man-in-the-middle (MITM) attacks when establishing a session. However, as launching such MITM attacks is possible for cleartext communications without the multihoming protocol as well, this has been found to be an acceptable security tradeoff.

The protocol still needs to ensure protection against 3rd party flooding attacks and temporal on-the-path attacks, i.e., issues 2 and 3 of Section 3.3.1. As described below, these can be addressed e.g., by sending light-weight reachability test packets.

4.4.2 CGAs and HBAs

Cryptographically Generated Addresses (CGAs) provide a means to encode a public key in the 64-bit interface identifier of the IPv6 address [3]. The receiver can verify that only the owner of the corresponding private key could have signed the messages.

Hash Based Addresses (HBAs) [4] is an extension of CGAs. The list of prefixes (along with a public key or a random number) is similarly encoded in the interface identifier, so when a prefix is added[3], all the addresses need to be changed.

Both approaches provide authorization without identification. CGA requires heavier cryptographic operations but is more flexible than HBA because the addition or removal of a prefix does not require changing all the addresses. An address can also have both CGA and HBA properties.

The strength of CGAs and HBAs is O($2^{59+16*Sec}$), where Sec is a security value, 0-7. An attacker would need to launch a brute-force attack to find a data structure which includes the hijacked and target prefixes. This would allow redirection of an IP address to a (random) HBA address in the target prefix. We assume HBA/CGA addresses are sufficiently strong. [4]

HBA or CGA do not prevent Man-in-the-Middle (MITM) attacks, but require that the attacker must be on the path when the data structure is exchanged and stay on the path for the duration of the attack. That is, the attacker must be able to change the interface identifiers of addresses used in the session (in addition to the data structures and signatures). [4]

HBAs do not provide protection against third party bombing against a subnet. That is, the attacker X can initiate communications with host A, generating HBA addresses including the prefixes of both X and target B, and redirect the session to a (random) HBA address of prefix B [4]. Similarly, CGAs do not provide protection against third party bombing against a host of attacker's choosing.

Addressing third party bombing requires either a return routability check before the locator is used for communications [4], or requiring that the sender of the locator update shows a certificate (one that the recipient can verify) that

[3] It is not strictly required to change the addresses at prefix deletion, because leaving it in is doing no particular harm.

the sender "owns" the prefix. As the latter requires a significant trust infrastructure, we conclude that using return routability is likely going to be a simpler choice.

4.4.3 Securing the Locator Exchange

Securing exchanging the list of locators (or securing a later change of locators) requires that the shim6 protocol is used to pass the data structure, and the message is signed using a CGA signature. The receiver has to store the data structure for as long as the multihoming state persists between the prefixes, and verify the signature. All the further locator changes must similarly include the data structure and must be signed.

With more mobile scenarios, it may be useful to use only CGAs, to avoid the need for renumbering all the addresses when a prefix changes. If one wants to support adding new locators on the fly, the addresses must support CGA by including the public key in the address generation. In this case using CGA+HBA allows for easy verification with HBAs in stable conditions, but authorizes the addition of a locator with the exchanged public key as needed.

An open question is how the message is signed and the message is sent, as CGAs are specified only for link-local Neighbor Discovery. If building an ad-hoc IPv6-in-IPv6 tunnel between the endpoints is not an option, the CGA mechanism would need to be retrofitted to a destination option, extension header, or some other means of communications. This requires changes to the code though – and has a potential IPR issue: there have been patent applications on CGAs, but free use has been granted for the current specifications. Augmenting CGAs to use something other than Neighbor Discovery might get these issues back on the surface.

4.5. Network Failure Detection and Reaction

As we described in Section 3.2, relying on the input from the network to achieve quick and reliable address selection did not seem like a good idea. Similarly, when a failure occurs, depending on the network to somehow "report" the error (instead of just discarding the packets) to the session endpoints is not robust. These can still work as optimizations, but more generic failure detection and reaction methods are needed.

The proposal [2] is to select one primary address pair for each session. Presumably this is the one used to set up the session, before the multihoming context has been set up. However, it is possible that the multihoming state is created between hosts, and later additional sessions would need to be established. If necessary, the peer could tell in the shim6 exchange which addresses should be preferable in the future. If so, some kind of interface to the default address selection rules would be needed. We do not see this as a strict requirement: if the host would have so many addresses that choosing among them would be difficult, the simplest approach might be not telling the peer about a subset of addresses [2].

There is plenty of reachability information available scattered through the protocol stack. At least the following information could be used to monitor the operational address pairs [2]:

- Positive feedback from the upper layers; for example, TCP connection is progressing.
- Negative feedback from the upper layers; for example, TCP is not getting ACKs.
- Lower layer information; as this information is not end-to-end, it can only provide reliable negative feedback about sessions using a failed local component.
- Reachability tests; mechanisms done by the failure detection protocol.
- ICMP error messages; for example, certain ICMP errors designate persistent failure scenarios.

A good question is which part of the protocol stack should deal with the end-to-end failure detection. It would seem best to make the shim6 failure detection aware of the transport protocol specific information, in the same way that lower-layer local indications (link up/down, IP address operational, etc.) can be used as input.

It is important to realize that some address pairs may only work in one direction, i.e., being able to receive with (src=A_1, dst=B_1) does not mean that responses would necessarily get back; they might need to use a different address pair, like (B_2, A_1) (see Figure 3 [2]). This needs to be taken into account when designing the failure detection protocol. The case when there exist only unidirectionally operational address pairs is also theoretically possible [2]. However we believe having to deal with them should not be required for stable conditions, possibly only for session survivability, as establishing a TCP session requires a bidirectionally working address pair.

An important consideration is how the secondary address pairs should be selected. Assuming two hosts A and B had two addresses each, one could consider the following:

1. Try the address pair that diverges the most from the currently used first; this should be able to deal with a failure at either end.
2. If there is indication that most sessions still work, the failure has typically been at the remote site.

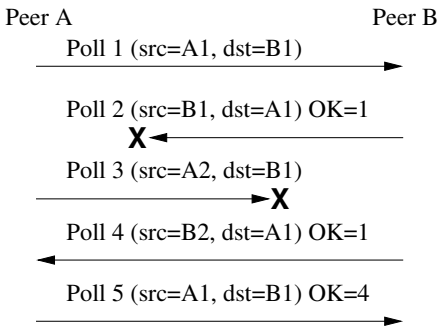

Figure 3. Detecting unidirectional failures

3. Otherwise, the local locator should typically be changed first; this addresses cases such as source address selection not being able to retry (see Section 3.2), ingress filtering, or an address no longer being locally operational (e.g., link down).

4.6. Multiplexing and Demultiplexing

When a failure occurs, the IP addresses used by the applications (ULIDs) stay the same, while the shim translates the packets to use different addresses at egress and rewrites them back at ingress. This is called multiplexing and demultiplexing. [21]

This is challenging because the parties must ensure the mapping is reversible; in particular, the receiving host must the able to distinguish which multihoming context (between the host and different peers, or multiple contexts between the same peers) each packet belongs to. At worst, inappropriate demultiplexing could result in corrupting the data stream with unrelated packets, so demultiplexing must be done with care. [21]

Two main approaches have been proposed: using the Flow Label field (in one of several ways) or defining a specific destination option or extension header to carry an identifier. We'll explore these below.

There are also some packets, specifically ICMP packets, which are sent in the network in response to a data packet, including the rewritten IP addresses in the payload. The demultiplexing function must therefore capture such ICMP messages, translate them accordingly, and pass them up in the stack. [21]

The typical assumption in the design has been that the peer must be notified prior to starting to use a different locator, so that the demultiplexing can succeed. However, we could imagine that it would be possible to "piggyback" that notification on the packet sent using the new locator as long as that packet (1) includes sufficient security information, and (2) carries the multihoming state update.

4.6.1 Flow Label vs Explicit Tag

Obviously, identifying is trivial when no multihoming context has been set up. As a host cannot know when a failure occurs, it needs to be ready at all times after the multihoming state has been established. Thus the labels/tags need to be agreed on, even if they would not need to be used, prior to the failure.

It is also important to remember that the operating system kernel already knows how to demultiplex (non-shim) packets; this is done by IP addresses and TCP/UDP port numbers, and in some cases, using other mechanisms. These could potentially used as well. However, this does not help in cases where such information is not available. We also note that this assumes that the port number space is unique across all the IP addresses: on a single host, application 1 cannot use the IP address A_1 with the same port as application 2 with IP address A_2.

If flow label (which is unique across {srcIP, dstIP, flow}) could be used, it would have two main advantages:

- no packet size increase, which could have potentially led to fragmentation and PMTUD problems, and
- no complications with firewalls or packet filters, which might not be able to parse or jump over a new header or option.

On the other hand, using an explicit tag would also have advantages:

- potentially simpler design because the tag allocation mechanisms can be defined as deemed fit, and
- would not use the flow label field; shim6 use might place (minor) constraints on the future use of the flow label.

For flow label design is still open, but there are at least two main candidates:

1. Sender-based allocation and reservation, and
2. Receiver-based allocation for backup locator pairs.

In the former, proposed by us, when the flow label is chosen, the particular label is reserved until the host runs out of flow labels. In particular, the label must be reserved so that it won't be used to communicate with the other addresses in the peer's locator set (current or future). Thus when the failure occurs, the sender can just switch to using different source and destination locators while preserving the flow label. The receiver may have other peers which have chosen (at random) to use the same flow label, but as the receiver knows the locators of the sender through the shim6 protocol, it can unambiguously demultiplex the packets. Hence, the flow label for shim6 would need to be unique across

{source locator set, destination locator set, flow label}. The main issue with this appears to be whether we need to deal with the case of running out of the 20-bit flow label space; it may or may not be feasible to assume a host would need to have a million concurrent sessions[4].

In the latter, establishing the multihoming context triggers the reservation of a flow label for the backup address pairs. This allocation would be done by the receiver, because the receiver just needs to choose it so that it is able to perform the demultiplexing. The flow label reservation would be communicated in the multihoming exchange, and would be used by the shim only if a failure occurs when rewriting the packets at either end.

A minor downside with all the session survivability approaches is that if flow label would also be used in Quality of Service or some other use, the signalling for a different treatment of {src, dst, flow label} would need to be done again. With the former approach, the routers could, if they were shim6-cognizant, snoop the shim exchanges and set up the state automatically as well, but this would be an architecturally bad idea.

5. Further Analysis and Discussion

Previous Sections already include quite a bit of analysis and discussion, but a few lengthier topics deserve to be analyzed separately, in subsections below.

5.1. Multihoming vs Mobility

The multihoming and IP mobility[5] both require session survivability, and the question is often raised why not create a solution that solves both the problems at the same time.

We believe that the differences and similarities of these two problems and the assumptions have not been sufficiently well understood to make such a decision now.

Note that most of the IP mobility issues would be mitigated if the mobility were able to use the"make-before-break" concept. While predictive mobility modes exist, typically depending on them it is not feasible.

We analyze that the main differences between multihoming and mobility are:

- Mobile nodes do not know their new IP address before they move, while with multihoming the secondary address is known from the start, and even renumbering is a slow process. Thus when a mobile node moves, it is no longer possible to prepare for the move, and the connectivity to the old IP address has been lost.
- Mobile nodes move or must be prepared to move much more frequently (even once a second) than sites renumber (typically at most once a year).
- Mobile node is not expected to return to using the old address when moving; the multihomed site's addresses are going to be valid again after a failure has been corrected.
- Mobile nodes typically have a helper "home agent" which is assumed to be always on; there is no such thing for multihoming. However, there is desire to find a mobility solution that would not have such a dependency, or at least narrow the responsibilities to just be a "my current location" referral service.

These seem to have the following implications:

- As HBA address set must be changed if there is any change in the prefix set, HBAs are not usable for frequent renumbering or mobility. On the other hand a CGA or CGA+HBA address could possibly be used.
- Shim6 uses locators as ULIDs. These change rapidly, and applications keep using them even after the IP address has been removed from the host. Using a separate name space would be better for mobility. This seems to break (1) referral/callback lookup mechanisms (at least forward+reverse DNS no longer works), and (2) connectivity if the application would want to talk to a new host which has been given the same IP address as the already used ULID.
- Shim6 can be designed so that enabling session survivability requires the nodes to signal the IP addresses before the addresses are used; such design has a number of benefits for demultiplexing.

We conclude that these differences seem to be sufficiently constraining not to overload the timing-critical multihoming with the mobility problem as well. This topic deserves research of its own. However, it might still be a good idea to figure out an alternative to HBAs if HBAs turn out to be impractical (e.g., due to the IPR concerns).

5.2. IPv6 vs IPv4

As the site multihoming issues apply to both IPv6 and IPv4, the question is sometimes raised why not design a solution for both IPv4 and IPv6?

[4] We note that there are relatively complex optimizations that allow even a million sessions per peer host if necessary. For example, after the context has been exchanged with peer A with addresses A_1 and A_2, with certain assumptions new sessions to other peers could reuse the flow label as long as the new peer's IP address would not be A_1, A_2 or any of the other known addresses.

[5] We use the term "IP mobility" in a generic sense, not just restricting to Mobile IPv6.

The explicit choice has been to avoid having to make tradeoffs for keeping IPv4 compatibility. The specific reasons have never been documented, but we believe focusing on IPv6 is reasonable for the following reasons:

- IPv6 has more bits in the address. This allows creating designs which are impossible or would have to be done differently with IPv4. For example, HBAs use the 64-bit interface identifier for obtaining sufficient cryptographic strength.
- IPv6 has a 20-bit Flow Label. The flow label field could be used in one of several ways (see Section 4.6) as a multihoming context tag, requiring no packet overhead; adding packet overhead complicates fragmentation/reassembly and Path MTU Discovery, and these do not work very well in IPv4 as it is [17].
- Small IPv4 sites can multihome using NATs, reducing their need for a multihoming solution; as IPv6 does not have NATs, these people have no corresponding IPv6 multihoming solution though their IPv4 needs have been at least partially satisfied.
- IPv4 has a lot more legacy; for example, 70% of web sites are not accessible if a new IP option is added to the packet [17]; this would constrain the design.
- IPv4 has NATs and they would need to be traversed and the state kept alive. The design would likely be quite a bit different.

All of these would be resolvable (if a sufficient alternative to HBAs can be found) with a more complex design that does not optimize where IPv6 could be optimized. We still conclude that it makes sense to focus on the IPv6 designs only, and possibly later create an IPv4 adaptation of the protocol if deemed appropriate.

5.3. Independence and Traffic Engineering

Shim6 does not solve sites' desire to be independent of their ISPs; especially larger sites want to avoid service provider lock-in, and want to be able to switch providers without having to renumber their network [24]. In other words, the sites want provider independent (PI) addresses.

Larger, especially multi-national sites also have desires to engineer the (incoming) traffic flows. This is required especially if they have a single address assignment – so they would like to advertise subprefixes from different geographical locations; if they have no PI addresses, this should be no problem. Small sites may want to load-balance the traffic over several links, but this could be achievable.

We have observed that the pressure has been building in 2005 in Regional Internet Registries, at least ARIN and RIPE, to allow PI allocations to a wider audience (e.g., all the member organizations or any site at all). This has not been considered scalable as all of these need to be routed globally [24], but there are also disagreements over whether that's actually the case or not.

As unfortunate as it may be for better technical development, we assume that sooner or later the allocation policies will get relaxed, and some sites will get PI addresses. However, hopefully these come with a sufficient cost to discourage those that don't really need them. We expect that small and medium-sized sites could very well use shim6 and provider-based addressing, and for SME and SOHO enterprises shim6 would be an ideal and architecturally sound solution. These small-to-medium sites should not get provider independent globally routable addresses as it would discourage them from using shim6.

An interesting consideration is the "convergence speed" of the shim6-based session survivability compared to the convergence speed of BGP-based multihoming. The most pessimistic view is that BGP convergence throughout the Internet could take even a dozen minutes, though closer in topology the speed is certainly much faster, less than a minute [16]. If the use of BGP would not be suitably fast to maintain session survivability, the potential shim6 userbase could be much larger if it would provide faster convergence.

5.4. Reverse and Forward DNS for Locator Search

Section 4.1 quickly described applications that do referrals and callbacks. Especially for these applications it is important to be able to find the other locators, given just one address.

One proposed way to do so is to look up the PTR record in the reverse DNS for the address, and look up the addresses from the forward DNS name the pointer refers to. [20]

This assumes that forward and reverse DNS trees are managed sufficiently well, so that all the addresses have a reverse record that points to a name which lists all the addresses of a node.

The critical assumptions are therefore:

- The ISPs allow the sites to manage the reverse DNS entries of the addresses they use (this may be a stretch for home and similar users), and
- The sites control a provider-independent domain name under which they can record the hostnames and addresses (this may be a stretch for home users).

We conclude that reverse and forward lookups for searching the locators has a chance of succeeding in well-managed sites, but we fear that sites most interested in

shim6 may not be sufficiently well-managed. Luckily enough, the support for locator search is not needed with "classical" client-server applications.

5.5. Deployment Considerations

We analyze that the key benefits of shim6 from the deployment perspective are:

- 100% interoperation (but without multihoming benefits) with legacy hosts, i.e., feasible incremental deployment,
- APIs and IP packets do not need to be changed, especially for simple applications,
- multihoming state does not need to be established immediately, saving in latency and bandwidth,
- does not require any rendezvous servers or other services from the network; fully compatible with the end-to-end paradigm, and
- shim6 "correspondent node" functionality can be deployed and enabled transparently, implying an easy deployment path for vendors.

6. Conclusions

While the debates in various address allocation fora about provider independent address allocations for sites still rage on, shim6 is being designed to provide redundancy in a scalable manner. It is expected that shim6 will be of most interest for small and middle-sized sites, but the outcome is likely linked with the decisions to be made about address assignments; if getting PI addressing is easier and cheaper than deploying and maintaining a shim6-based site multihoming solution, the sites are going to go for their own addresses even if that would have serious implications on interdomain routing scalability.

We described and analyzed the shim6 proposal. In general, the design so far seems to be reasonable. There are obviously many areas which still need work and decisions how to move forward. The most pressing issues are likely going to be on address selection and failure detection, the lack of independence and traffic engineering, and possibly HBA/CGA IPR issues.

7. Acknowledgements

Marcelo Bagnulo and the anonymous reviewers are thanked for their review and suggestions for improvement.

References

[1] J. Abley, B. Black, and V. Gill. Goals for IPv6 Site-Multihoming Architectures. RFC 3582, Aug. 2003.
[2] J. Arkko. Failure Detection and Locator Selection in Multi6. draft-ietf-shim6-failure-detection-00.txt, Jan. 2005. Work in progress.
[3] T. Aura. Cryptographically Generated Addresses (CGA). RFC 3972, Mar. 2005.
[4] M. Bagnulo. Hash Based Addresses (HBA). draft-ietf-shim6-hba-00.txt, July 2005. Work in progress.
[5] M. Bagnulo and J. Arkko. Functional decomposition of the M6 protocol. draft-ietf-shim6-functional-dec-00.txt, July 2005. Work in progress.
[6] F. Baker and P. Savola. Ingress Filtering for Multihomed Networks. RFC 3704, Mar. 2004.
[7] J. N. Chiappa. Endpoints and Endpoint Names: A Proposed Enhancement to the Internet Architecture, 1999.
[8] R. Draves. Default Address Selection for IPv6. RFC 3484, Feb. 2003.
[9] F. Gont. TCP's Reaction to Soft Error. draft-gont-tcpm-tcp-soft-errors-01.txt, Oct. 2004. Work in progress.
[10] J. Hagino and H. Snyder. IPv6 Multihoming Support at Site Exit Routers. RFC 3178, Oct. 2001.
[11] C. Huitema, R. Draves, and M. Bagnulo. Address selection in multihomed environments. draft-huitema-shim6-ingress-filtering-00.txt, Oct. 2005. Work in progress.
[12] C. Huitema, R. Draves, and M. Bagnulo. Ingress filtering compatibility for IPv6 multihomed sites. draft-huitema-shim6-ingress-filtering-00.txt, Oct. 2005. Work in progress.
[13] IETF. Better-than-nothing Security (btns) charter.
[14] IETF. Site Multihoming in IPv6 (multi6) charter.
[15] C. Kenjiro and et al. IPv6 Fix: TCP Connection Establishment.
[16] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed internet routing convergence. In *SIGCOMM*, pages 175–187, 2000.
[17] A. Medina, M. Allman, and S. Floyd. Measuring the Evolution of Transport Protocols in the Internet. Computer Communications Review, Apr. 2005.
[18] R. Moskowitz and P. Nikander. Host Identity Protocol Architecture. draft-ietf-hip-arch-03.txt, Aug. 2005. Work in progress.
[19] P. Nikander. Implications of Identifier / Locator Split.
[20] E. Nordmark. Multi6 Application Referral Issues. draft-ietf-shim6-app-refer-00.txt, July 2005. Work in progress.
[21] E. Nordmark and M. Bagnulo. Multihoming L3 Shim Approach. draft-ietf-shim6-l3shim-00.txt, July 2005. Work in progress.
[22] E. Nordmark and T. Li. Threats Relating to IPv6 Multihoming Solutions. draft-ietf-multi6-multihoming-threats-03.txt, Jan. 2005. Work in progress.
[23] P. Savola. Examining Site Multihoming in Finnish Networks. Master's thesis, Helsinki University of Technology, Finland, 2003.
[24] P. Savola and T. Chown. A Survey of IPv6 Site Multihoming Proposals. In *Proceedings of the 8th International Conference of Telecommunications (ConTEL 2005)*. IEEE, 2005.
[25] M.-K. Shin, Y.-G. Hong, J. Hagino, P. Savola, and E. Castro. Application Aspects of IPv6 Transition. RFC 4038, Mar. 2005.