IPv6 Site Multihoming Using a Host-based Shim Layer

Pekka Savola
CSC - Scientific Computing Ltd
PL405, 02100 Espoo
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 protection against failures. Commonly deployed IPv4 site multihoming mechanisms, BGP and NAT have not been available in IPv6. Thus new mechanisms are needed, and the Internet Engineering Task Force (IETF) has been working on this problem for some time now. A prevailing approach, at least for smaller sites, is to deploy multiple provider-assigned IP address prefixes from multiple ISPs on the sites.**

We analyze the implications of having multiple addresses from multiple ISPs on a host, and describe and analyze the IPv6 site multihoming solution called "shim6". The design is still in progress so we note and discuss many open issues. The biggest constraint of the protocol appears to be the inflexibility of so-called Hash Based Addresses, which are used to provide the security to the session survivability.

While the site multihoming solution is being developed, there is also active discussion on various address allocation fora on relaxing the globally routable address space criteria; the outcome will certainly be a factor in determining the practical applicability and deployment of shim6.

I. Introduction

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 protection against failures.

Site multihoming with IPv4 is typically achieved by routing (with Border Gateway Protocol, BGP) or using Network Address Translator -based mechanisms [1]. 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 [2], [1].

IPv6 address allocation policies and commonly-deployed prefix length filters for BGP advertisements 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, and to maintain future scalability.

At the same time, IETF IPv6 Site Multihoming (multi6) working group [3] has explored the requirements [4], threats [5] and architectures at depth. In late 2004, consensus emerged that a particular (beginnings of a) solution, called shim6, should be developed in a new dedicated working group.

The reader is assumed to be knowledgeable of identifiers, locators, and the identifier/locator split in general. [6]

In this paper, we begin by describing the generic issues with hosts having multiple addresses from different providers. Then we describe the shim6 proposal which provides a number of scalable site multihoming benefits. The focus of this paper is providing a concatenated overview of the proposal, and discussing and analyzing the most interesting and relevant design choices.

II. 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, it would still be desirable:

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

We discuss these below.

A. 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. This implies source-address based policy routing (with a very simple policy) at the border routers. [7], [1]

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. This happens because there is no working connectivity (with would pass the ingress filtering) for that prefix from the site to the ISP advertising the aggregate. This problem can be worked around by using tunnels to build backup connectivity to each ISP [8], which also obviates the address selection and session survivability issues.

B. Address Selection

For the 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.

For the 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. [9]

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 III-E for more.)

Destination address selection [10] 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 [11].

So, the specific requirements are:

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

Unfortunately, the fallback to the next address is not necessarily quick nor 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 so it would be counter-productive; and an 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 [12]. 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.

There are proposals to improve source address selection to retry [9], but such a mechanism will need to deal with the same situation as destination address selection that there may not be any feedback from the network on failing attempts.

Another mitigation technique 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 it is not clear how to obtain that reliably especially without new mechanisms; and updating DNS dynamically for temporary failures might be very impractical and the old data would still persist for in the DNS caches for the lifetime of the record's previous TTL.

C. Session Survivability

Session survivability is a more difficult problem because the existing protocols such as TCP and UDP cannot 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. [1]

Once switching a session to use a new address, address selection must also be performed. 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.

We will next look at the security issues of session survivability.

1) Security of the Session Survivability: Being able to redirect a session between two addresses to use different addresses has significant security threats [5].

Because a global trust infrastructure does not exist, the designs have had to cope with a different trust model. The analysis has studied how plaintext communications may be disrupted as of today. The current level of security is that if the 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 likely redirect communications. The security must not be worse than that; therefore the designs find the risk of permanent on-the-path attackers acceptable.

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 [5]:

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; there must be an upper limit how long on-the-path verification is valid, and
3) The attacker on a slow link could subscribe a large transmission from A to himself, then redirect it 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.

Additionally, 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.

D. Conclusions

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

- If tunneling [8] can be used, session survivability and address selection work; only ingress filtering 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 which also work when there is no feedback from the network [13]. 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 avoid most 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 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.

III. 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). When a failure occurs, the IP addresses used by the applications (ULIDs) stay the same, while the shim translates the packets to use

[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.

different addresses at egress and rewrites them back at ingress (see Figure 2; the mapping is therefore reversible. [14]

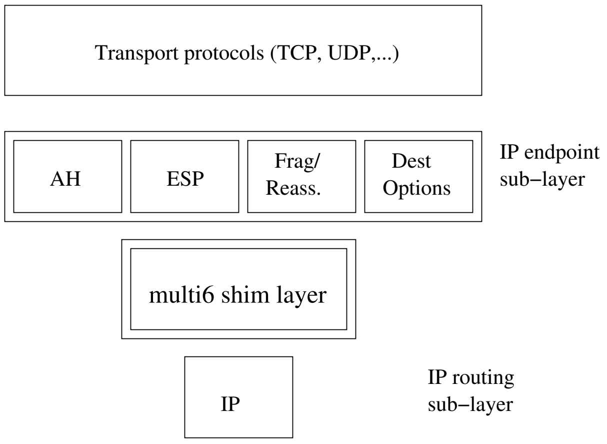

Fig. 1. The placement of the Shim [14]

Our analysis is that Shim6 only provides session survivability, while making address selection simpler. The solution does not provide provider independence (and consequently, does not eliminate the renumbering) or traffic engineering [1]. This is discussed at more length in Section IV-C.

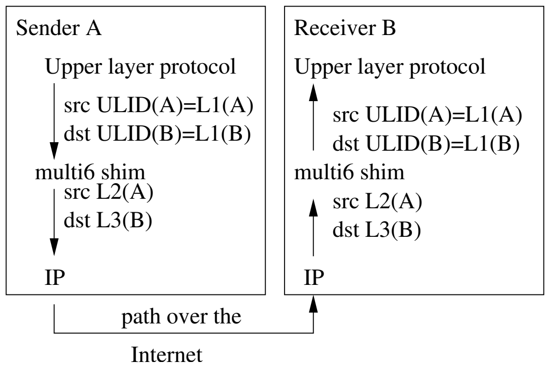

Fig. 2. The shim mapping with changed locators [14]

A. 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). [15]

There are multiple choices on what the applications could use as identifiers of a session [15]:

- 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.

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 general.

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 IV-D); there may be other alternatives [15].

A relatively small drawback of using existing locators as ULIDs is issues with mobility and renumbering, as described in Section IV-A.

B. 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.

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 to be protected against 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 III-D.

C. 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 [16].

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),
- 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 III-D.

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)? – We think that the host should not tell all the locators if the list is long, so they should probably all be exchanged at once; this also avoids 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 no good either.
- Are there any considerations for adding new locators on the fly? – We currently lack the security mechanisms to do this properly but otherwise it would seem doable (see Section IV-E).
- 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 dismantled. However, closing the state explicitly might be useful if the previous locator ends up being given to someone else, and that host would try to establish communications with the peer.

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

TABLE I
SUMMARY OF MULTIHOMING STATE

State	Size	Section
List of host's locators	at least 32 bytes	III-C
Security data structure	at least 100 bytes	III-D
HBA Signature	at least 40-50 bytes	III-D
Address pair pref. (if needed)	1 bytes	III-E
Context tag (if needed)	4-6 bytes?	III-F
Flow label to be used[2]	4 bytes	III-F.1

D. Secure Locator Exchange

In Section II-C.1 we described generic threats with multiple locators and why just using return routability is not sufficient to obtain security of locator changes.

Hash Based Addresses (HBAs) [17], an extension of Cryptographically Generated Addresses (CGAs) come to the rescue. The list of prefixes (along with a public key or a random number) is cryptographically encoded in the

64-bit interface identifier of the IPv6 address, so when a prefix is added[3], all the addresses need to be changed.

The strength of 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 include the hijacked and target prefixes. This would allow redirection of an IP address to a (random) HBA address in the target prefix. [17]

It is assumed that HBAs are sufficiently strong; while we assume that Sec=0 would be typically used initially, if the users become aware of attack attempts, higher values would start to be used. In the following analysis, we assume that HBAs are sufficiently strong.

HBA does not prevent Man-in-the-Middle (MITM) attacks, but requires 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). [17]

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 a (random) HBA address of prefix B [17]. Addressing third party bombing requires either a return routability check before the locator is used for communications [17], or requiring that the sender of the locator update shows a certificate (that the recipient can verify) that the sender "owns" the prefix. As the latter requires a significant trust infrastructure, using return routability is likely going to be a simpler choice.

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.

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 potential for an IPR issue: there have been patents 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.

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

E. Network Failure Detection and Reaction

As we described in Section II-B, depending 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 [18] 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. Would the address selection data, if transferred in the shim6 exchange, be somehow used here as to prefer a particular address pair; that is, is it necessary for the peer to be able to tell which addresses it would prefer to be used? 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 [18].

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 [18]:

- 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). This needs to be taken into account when designing the failure detection protocol. The case when there exists only unidirectionally operational address pairs is also theoretically possible, but in that environment one could not even establish new TCP sessions, so it may be situation the shim6 protocol might have to be able to use for session survivability, but not in the stable conditions. [18]

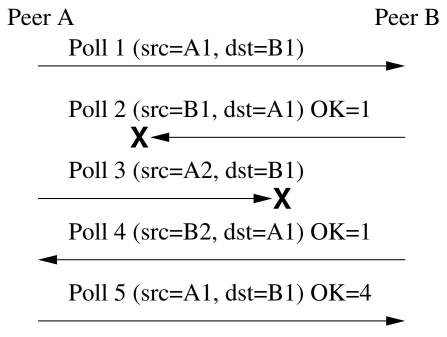
Peer A
Peer B
Poll 1 (src=A1, dst=B1)
Poll 2 (src=B1, dst=A1) OK=1
X
Poll 3 (src=A2, dst=B1)
X
Poll 4 (src=B2, dst=A1) OK=1
Poll 5 (src=A1, dst=B1) OK=4

Fig. 3. Detecting unidirectional failures [18]

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.
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 II-B), ingress filtering, or an address no longer being locally operational (e.g., link down).

F. 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. [14]

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 it must be avoided. [14]

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. [14]

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 includes (1) sufficient security information, and (2) carries the multihoming state update (see Section IV-E).

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 a help 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 signle 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 $\{$src, dst, 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 overload the flow label field; if flow label would need to be used for some other important application, it would still be possible.

For Flow Labels, the 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, but even then, it would be possible to reserve the flows per locator set so that a host could have at most a million concurrent sessions between each host. That should be enough, at least.

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 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 probably be architecturally a bad idea.

IV. 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.

A. *Multihoming vs Mobility*

The multihoming and IP mobility 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.

We analyze the main differences between multihoming and mobility to be:

- Mobile nodes cannot know when they will move; sites will know beforehand if/when they will need to change the providers and have time to prepare. Therefore when an unexpected move occurs, it's already too late to start quick preparation as the mobile node has lost its connectivity to the old IP address.
- 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. Another solution for securing against the attacker being briefly on the path would need to be designed.
- Shim6 uses locators as ULIDs. These change rapidly, and applications keep using them even after the IP address is 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 could not 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. However, it might still be a good idea to figure out an alternative to HBAs if HBAs turn out to be impractical (e.g., the IPR concerns or too frequent renumbering).

B. *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?

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 20-bit Flow Label. The flow label field could be used in one of several ways (see Section III-F) 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 [19].
- 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 roughly 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 [19]; 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.

C. *Independence and Traffic Engineering*

Shim6 does not solve the 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 [1]. In other words, the sites want provider independent (PI) addresses.

Large, 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 [1], but there are also disagreements over whether that's 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.

D. *Reverse and Forward DNS for Locator Search*

Section III-A 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. [15]

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 applications.

E. Adding a New Locator

Adding new locators afterwards is relatively simple. However, HBAs make that a bit more difficult, as the whole HBA set would need to be regenerated at such an event. As it is, adding locators to the existing set is not possible without alternative security mechanisms to verify the address. One possibility would be using CGA-only addresses, but then the public key would need to be verified, which might not always be possible.

However, let us assume a solution for this could be found. Let us consider the case where a host would like to immediately send from a new address, without prior coordination, piggybacking a locator update. This would seem to require that when the packet is first processed by the host, the destination option or extension header is processed to find the locator update (and other multihoming state). Only then the packet can be injected to the shim, as otherwise the state would not be up-to-date.

V. 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.

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. The most urgently work seems to be needed in securing the locator exchange, in particular, figuring out an acceptable way to add locators (e.g., add a prefix) without disturbing the existing sessions; an alternative for HBAs might be useful for other reasons as well.

It seems to be widely accepted that the focus is on IPv6, but there is still debate about to which degree one should look at mobility as well as multihoming.

References

[1] P. Savola and T. Chown, “A Survey of IPv6 Site Multihoming Proposals,” Accepted for publication for IEEE 8th Conference on Telecommunications (ConTEL 2005).

[2] P. Savola, “Examining Site Multihoming in Finnish Networks,” Master’s thesis, Helsinki University of Technology, Finland, 2003. [Online]. Available: http://staff.csc.fi/psavola/di.ps

[3] IETF. Site Multihoming in IPv6 (multi6) charter. [Online]. Available: http://www.ietf.org/html.charters/multi6-charter.html

[4] J. Abley, B. Black, and V. Gill, “Goals for IPv6 Site-Multihoming Architectures,” RFC 3582, Aug. 2003.

[5] E. Nordmark and T. Li, “Threats Relating to IPv6 Multihoming Solutions,” draft-ietf-multi6-multihoming-threats-03.txt, Jan. 2005, work in progress.

[6] P. Nikander. Implications of Identifier / Locator Split. [Online]. Available: http://www.cs.hut.fi/ pmrg/NETS1a/IdLocSplit-04-10-28.pdf

[7] C. Huitema, R. Draves, and M. Bagnulo, “Ingress filtering compatibility for IPv6 multihomed sites,” draft-huitema-multi6-ingress-filtering-00.txt, Oct. 2004, work in progress.

[8] J. Hagino and H. Snyder, “IPv6 Multihoming Support at Site Exit Routers,” RFC 3178, Oct. 2001.

[9] C. Huitema, R. Draves, and M. Bagnulo, “Address selection in multihomed environments,” draft-huitema-multi6-addr-selection-00.txt, Oct. 2004, work in progress.

[10] R. Draves, “Default Address Selection for IPv6,” RFC 3484, Feb. 2003.

[11] M.-K. Shin, Y.-G. Hong, J. Hagino, P. Savola, and E. Castro, “Application Aspects of IPv6 Transition,” RFC 4038, Mar. 2005.

[12] F. Gont, “TCP’s Reaction to Soft Error,” draft-gont-tcpm-tcp-soft-errors-01.txt, Oct. 2004, work in progress.

[13] C. Kenjiro and et al. IPv6 Fix: TCP Connection Establishment. [Online]. Available: http://www.v6fix.net/docs/wide-draft-v6fix.en

[14] E. Nordmark and M. Bagnulo, “Multihoming L3 Shim Approach,” draft-ietf-multi6-l3shim-00.txt, Jan. 2005, work in progress.

[15] E. Nordmark, “Multi6 Application Referral Issues,” draft-ietf-multi6-app-refer-00.txt, Jan. 2005, work in progress.

[16] M. Bagnulo and J. Arkko, “Functional decomposition of the M6 protocol,” draft-ietf-multi6-functional-dec-00.txt, Dec. 2004, work in progress.

[17] M. Bagnulo, “Hash Based Addresses (HBA),” draft-ietf-multi6-hba-00.txt, Dec. 2004, work in progress.

[18] J. Arkko, “Failure Detection and Locator Selection in Multi6,” draft-ietf-multi6-failure-detection-00.txt, Jan. 2005, work in progress.

[19] A. Medina, M. Allman, and S. Floyd, “Measuring the Evolution of Transport Protocols in the Internet,” Computer Communications Review, Apr. 2005.