Network Working Group
Internet-Draft
Intended status: Informational
Expires: December 6, 2007

B. Carpenter
IBM

June 4, 2007

General Identifier-Locator Mapping Considerations
draft-carpenter-idloc-map-cons-01

Status of this Memo

This Internet-Draft will expire on December 6, 2007.

Copyright Notice

Abstract

This document presents various general considerations about the mapping between identifiers and locators at the network and routing level in the Internet.

Table of Contents

1. Introduction

In the discussions following the IAB Routing and Addressing workshop [I-D.iab-raws-report], a common observation is that a key issue in today's Internet is the overlapping semantics of IP addresses used as 'locators' and as 'identifiers.' A common conclusion from this is that the architectural solution is a 'identifier-locator split' (sometimes abbreviated as 'id-loc'). This conclusion is discussed below. However, if we accept that locators and identifiers have different (even if possibly overlapping) semantics, the question immediately arises of how they can be mapped onto one another. Given a locator, which identifier(s) refer to the same entity? Given an identifier, which locator(s) refer to the same enity? The main part of this document discusses the considerations raised by these questions.

Suggested discussion list: ram@iab.org.

1.1. Definition of terms

IP Address: an IPv4 or IPv6 address, viewed as an opaque 32 or 128 bit quantity.

Stack: An active instantiation of the TCP/IP model; one participant or the process on one side of an end-to-end communication. That participant may move and may be represented by multiple hosts. Quoting from an unpublished document produced within the former Namespace Research Group [NSRG]:

------------Start Quotation------------

Today, a host may represent multiple entities. This happens when a service provider hosts many web sites on one server. Similarly, a single entity may be represented by multiple hosts. Replicated web servers are just such an example. These entities are "protocol stacks" or simply "stacks", instantiations of the TCP/IP model, be they across one or many hosts. A stack is defined as one participant or the process on one side of an end-to-end communication. That participant may move and may be represented by multiple hosts.

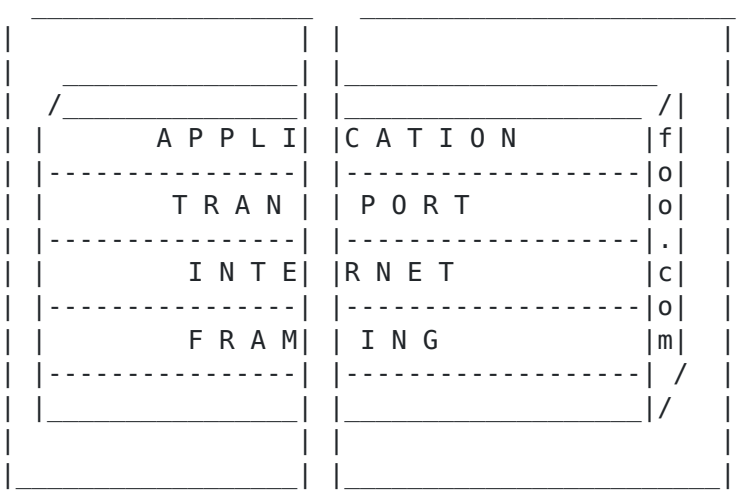


Figure 4: Another application: single stack represented by multiple hosts

Each instance of a stack has a name, a "stack name". At an architectural level the Name Space Research Group debated the value of such names, and their associated costs. Forms of this name are used in numerous places today. SSH uses public/private key pairs to identify end points. An HTTP cookie anonymously identifies one end of a communication, in such a manner that both the connection and the IP address of the other end point may change many times. Stack names are intended to identify mobile nodes, devices behind NATs, and participants in a content delivery or overlay network.

------------End Quotation------------

Locator: A binary quantity (not necessarily an IP Address) that can be used by a routing or forwarding device to decide where to send a packet.

Identifier: A binary quantity (not necessarily an IP Address) that can be used by a Stack "A" to uniquely identify another Stack "B"

both for bilateral communication and for informing a third Stack "C" that it should communicate with Stack "B". (Note that there is an assumption in this definition that a Stack is the entity we require to identify; in this era of virtualized servers with failover capabilities, and of mobile clients, this seems to be a reasonable assumption.)

Namespace: a set of natural numbers, each of which is referred to as a name. Since it is a set, by definition each name is unique and thus the namespace is unambiguous. Locators and Identifiers must belong to specific namespaces.

Namespace Context: the context within which a given namespace retains its uniqueness property. (For example, the Namespace Context of the Namespace created by [RFC1918] is a single Internet site.)

1.2. Related Work

[I-D.nikander-ram-ilse] discusses the design space for Identifier-Locator Separation and contains excellent background references. [I-D.farinacci-lisp], [I-D.wang-ietf-efit], and [I-D.templin-ipvlx] discuss solutions along the encapsulation axis. SHIM6 [I-D.ietf-shim6-proto] is a host-based solution along the dynamic translation axis. [GSE] was a router-based solution along the dynamic translation axis. HIP [RFC4423] is a real new namespace.

Early thinking on this whole topic should be credited to Noel Chiappa [ENDPOINTS], who in turn cites related work back to 1978.

2. Identifier and Locator Behaviour Today

[RFC2101] discussed "IPv4 Address Behaviour Today" (where "today" was February 1997). It focused on the then new issues caused by private addresses [RFC1918], dynamic address allocation, and network address translation. Its fundamental observations remain true:

------------Start Quotation------------

> Due to dynamic address allocation and increasingly frequent
> network renumbering, temporal uniqueness of IPv4 addresses is no
> longer globally guaranteed, which puts their use as identifiers
> into severe question. Due to the proliferation of Intranets,
> spatial uniqueness is also no longer guaranteed across routing
> realms; interconnecting routing realms could be accomplished via
> either ALGs or NATs. In principle such interconnection will have
> less functionality than if those Intranets were directly
> connected. In practice the difference in functionality may or may
> not matter, depending on individual circumstances.
> ...
> As far as temporal uniqueness (identifier-like behaviour) is
> concerned, the IPv6 model [RFC 1884] is very similar to the current
> state of the IPv4 model, only more so. IPv6 will provide mechanisms
> to autoconfigure IPv6 addresses on IPv6 hosts. Prefix changes,
> requiring the global IPv6 addresses of all hosts under a given prefix
> to change, are to be expected. Thus, IPv6 will amplify the existing
> problem of finding stable identifiers to be used for end-to-end
> security and for session bindings such as TCP state.
>
> The IAB feels that this is unfortunate, and that the transition to
> IPv6 would be an ideal occasion to provide upper layer end-to-end
> protocols with temporally unique identifiers. The exact nature of
> these identifiers requires further study.
>
> ------------End Quotation------------
>
> How is the discrepancy between identifier and locator namespaces
> handled today (2007)?
>
> The discrepancy can be summarised as follows: the progressive
> shortage of IPv4 addresses, coupled with the lack of economic
> incentive to deploy IPv6, has led to generalised use of private
> addresses within enterprise and home networks. Thus, we have
> disjoint Locator Namespaces - roughly speaking, one very large
> Locator Namespace commonly referred to as the Internet, and a
> separate Locator Namespace for every network "behind" a NAT.
> However, much software, and many protocols, have been designed on the
> assumption that we have a single Identifier Namespace, which is
> furthermore assumed to be identical to the Internet Locator
> Namespace. Neither of these assumptions is true.
>
> We should also note that with IPv6 partially deployed, we already
> have two distinct Internet Locator Namespaces. Mathematically, we
> could consider them as a single Locator Namespace (with the
> unambiguous IPv4 space being mapped as a subset of IPv6 space), but
> little existing software is prepared to treat IPv4 and IPv6 as a
> single Identifier Namespace, even though IPv6 does not suffer from

IPv4's problem of ambiguous addresses.

To work around these false assumptions, several measures have been taken, none of them as a result of architectural design:

- Network Address Translation. Effectively, a NAT creates a dynamic mapping between two Locator Namespace Contexts (sometimes using a port number to tag the mapping). The focus of a NAT is to enable packet delivery despite the namespace being fragmented; thus it acts as a context boundary in the Identifier Namespace. NATs and their associated ALGs attempt to hide this by (for example) recomputing TCP checksums. However, it is a fundamental fact that the NATs and ALGs cannot know how the two ends of a communication use the Identifier Namespace, and therefore it is logically impossible for them to fix up the Identifier Namespace in general. Furthermore, NATs do not publish the Identifier/Locator mappings they have created, so the affected Stacks have no sure way to discover it.
- Discovery by probing. For a specific application or set of applications, a way round the namespace discrepancy can be constructed, essentially by deducing from external evidence the Identifier/Locator mapping that a NAT has created, and signalling that mapping to all interested Stacks. The best known examples are STUN [RFC3489] and ICE [I-D.ietf-mmusic-ice].
- Build your own namespace. Fortunately for people wishing to construct novel Internet applications, there is a very convenient way to ignore the whole problem: make use of the fact that every NAT has an associated ALG that translates HTTP/TCP packets appropriately, and incidentally that HTTP passes through most security checks. Similarly, an HTTP proxy can hide an IPv4/IPv6 boundary. Thus, despite the strong recommendation to the contrary in [RFC3205], many application suites have been built to run over HTTP, basing the roots of their own Identifier Namespaces in some way in the DNS. An interesting example is the massive Web Services suite, whose XML-based namespace derives its uniqueness from the uniqueness of DNS names [<http://www.w3.org/TR/ws-addr-core/#namespaces>]. It is clear that if this technique were not available, we would have confronted the network's Identifier Namespace problem years ago, or all innovation would have stopped.

A separate observation about today's situation is that we can loosely divide upper layer implementations (transport layer and above) into two classes:

1. Those that use a socket API, or the equivalent, on the traditional assumption that an IP Address is simultaneously a unique Locator and a unique Identifier. These are automatically subject to unpredictable problems when they encounter Identifier/Locator ambiguity.

2. Those that avoid this assumption, for example by creating their own namespace or relying completely on the DNS.

Confusion can nevertheless occur, for example in an application that believes that using a URL-based namespace protects it from ambiguity, but encounters a URL containing a literal Net 10 address. One can easily imagine http://10.1.1.1:80/ referring to different entities at the two ends of a session.

A related issue is the way network elements are monitored and managed. Network operations staff know that the IP Address of a network element (router, switch, bridge, server or user device) both identifies and locates it. In a network of any size, the IP Addresses of network elements are embedded in network configuration and monitoring systems in many ways. The assumption generally made by operations staff is that there is a single namespace, but this may be false. An illustration of this assumption is that many MIBs include IP Addresses; the contents of such MIBs cannot readily be used outside their original Namespace Context. In enterprise networks that contain internal NATs, this is a known source of operational difficulty. It is in fact the motivator for one of the largest reported IPv6 deployment projects [<http://www.nanog.org/mtg-0606/pdf/alain-durand.pdf>].

3. Is a Split the Solution?

Given that our problem is overlapping namespaces, it's tempting to conclude immediately that splitting them is the solution. In a split solution, sites and hosts would have global network-level identities independent of the routing system and of individual service providers. The routing system would be free to assign and manipulate locators so as to assist route aggregation, support multihoming, and implement path selection policies. But what would a split mean, and is it practical?

As implied by the previous section, there are cases where a split would increase rather than decrease confusion. Enterprise network operations would be strongly affected - would the normal way to identify a misbehaving box be its locator or its identifier? Network management systems and configuration databases would have to be upgraded to support both. Firewalls would have to deal with fundamental change, since identifiers rather than locators would be the focus of many attacks. The way the DNS and the socket API are related to each other might change. Any upper layer implementation that believes that an IP Address is both a Locator and an Identifier would be in even greater trouble than today. And of course the fragile superstructure built on NATs and ALGs would be severely

challenged, especially during the transitional phase.

This is, no doubt, why the IAB's suggestion in [RFC2101], quoted above, has borne no fruit, except for HIP [RFC4423], which still remains R&D. But if we are to respond to the currently perceived routing and addressing problem, a choice must be made.

Two different approaches seem to be possible:
1. Completely disjoint namespaces. Identifiers can never be used as Locators, and vice versa.
2. Partially disjoint namespaces. In some Namespace Contexts, Identifiers can be used as Locators, and vice versa. In other Namespace Contexts, they cannot.

Since the choice between these two approaches will have significant impact on the mapping mechanism, they are now discussed in more detail.

3.1. Completely Disjoint Namespaces

In this class of solution, there is no overlap or possible ambiguity between an Identifier and a Locator. They are defined and managed orthogonally, apart from the mapping function. They never need to be disentangled by use of context information. They will not be equivalent when used as parameters of an API.

Conceptually, in this document, Identifiers identify stacks, not end-systems or application instances. Thus, http://www.example.com cannot be a stack Identifier. The full process of sending the first packet of a session would consist of these conceptual stages:
1. Map 'www.example.com' to an Identifier (i.e. map the application Namespace into the network stack Namespace).
2. Construct the first packet to be sent to this stack at this Identifier.
3. Map the Identifier into a Locator.
4. Forward the packet towards that Locator.
5. At or near the receiving stack, restore the original Identifier if it has not been trasmitted in the packet. This requires a reverse map from Locator to Identifier.

We may assume for present purposes that step 1 will involve the DNS. Step 3 will use the mapping service that is the object of this document. It may take place in the sending host, or it may take place in a nearby router. But since the two namespaces are disjoint, it will happen exactly once. Step 5 is the converse, if needed.

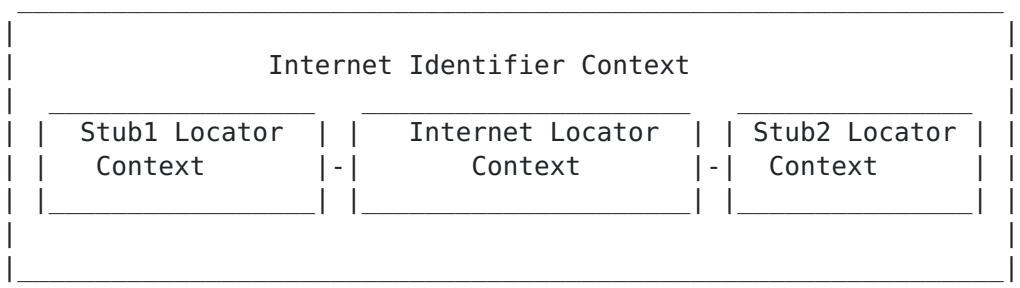
It seems evident that this model of completely disjoint namespaces is 'correct' from a logical (computer science) point of view. It is in

fact a classical layering solution reminiscent of, but not mapping exactly onto, the OSI model. ("Stacks" are a new layer, above the network endpoint layer and below the transport layer.) It definitely requires a namespace mapping mechanism acting at the layer boundary. If fully executed, this would clean up many of the glitches caused by IP address fragmentation today. From a practical point of view, the technical community will have to decide if there is an operationally and economically viable path to deployment of such a solution.

3.2. Partially Disjoint Namespaces

In this class of solution, we decide in advance that the syntax of an Identifier and that of a Locator are the same. Thus we can have overlapping scopes. A given token could simultaneously be a member of two Namespaces, distinguished only by context.

One simple model for this is illustrated as follows. Identifiers are defined to be all global in scope - there is exactly one Internet Identifier Namespace. Locators are defined to have either 'stub scope' or 'Internet scope'. A 'stub' is what is sometimes called a site network or an enterprise network; its main distinguishing characteristic is that it is not providing transit.



In this simple model, and assuming the same syntax for Identifiers and Locators, we could for example decide that a Stack inside Stub1 has Identifier=Locator in that context, but has different Locators in the Internet and Stub2 contexts. A Stack inside Stub2 has Identifier=Locator in that context, but different Locators in the Internet and Stub1 contexts. It is this ability to equate Identifier and Locator in the local stub network that principally distinguishes the partially disjoint model from the completely disjoint model. As far out from the sending host as that equality is valid, the Identifier/Locator behaves exactly like a classical IP Address.

To be clear, there is no reason why all Locators could not be

globally unique - today's ambiguous IPv4 addresses are a side-effect of the limited size of IPv4 addresses. If that constraint was applied, the three Locator Namespaces shown above could be strict subsets of the Internet Identifier Namespace, as a design choice.

It should be noted that there are echoes of this model in both SHIM6 [I-D.ietf-shim6-proto] and LISP [I-D.farinacci-lisp], but they both have rather more complexity than described above.

An operational advantage of this class of model is that it allows upper layers and management systems to treat the Identifier and Locator Namespaces as identical within the local (stub) context. Network management systems, as long as they stay within a single Locator Namespace Context, will be essentially unaffected. On the other hand, management across a Locator Namespace boundary would be just as awkward as management across a NAT today, unless the management system was updated to address network elements by Identifier.

Another practical advantage of this class of model is that if Identifiers and Locators, which share syntax, are in the same format as today's IPv6 addresses, upper layer software can to a large extent continue to use today's APIs.

A corresponding disadvantage is that, unlike the completely disjoint model, there is no 'a priori' distinction between an Identifier and a Locator. Network elements and Stacks would need to be able to treat tokens that look like an IP Address much as they do today, but with a clear definition of the context within which Identifiers and Locators are the same thing, and of the contexts in which they are not. As has been discovered for IPv6 scoped addresses [RFC4007], there is significant complexity in attaching scope rules to what is otherwise an architecturally opaque address value. Any design based on a partially disjoint model needs to specify how context and scope are managed, and this may add to the requirements for a mapping service and for APIs.

3.3. Common Requirements

Consider three Stacks A, B and C in three adminstratively separate sites. What Locator and Identitfier properties do they require?

Within its site, A must have an Identifier and Locator that are conveniently mapped for the operational staff (if not equal), and do not vary in a way unexpected by those operational staff. The same is true for B and C respectively.

When communicating with B, A must provide an Identifier that B can

rely on, and conversely. To send its packets, A must either know a Locator for B that is valid in the context of site A, or send the packet to a router that knows such a Locator (given B's Identifier). The same is true in the reverse direction.

It is irrelevant to A and B if the Locator used by A (or A's router) is only valid locally at site A, i.e. if the locator changes once or several times along the way. All that matters is that the packet be delivered to B, still carrying A's Identifier.

A must also be able to communicate with C on the same basis, and furthermore tell C that it may communicate with B as part of the same application. Since we have admitted that the Locator that A (or A's router) knows for B may only be valid locally, it is required that A only needs to communicate B's Identifier to C.

However, we must be careful not to read too much into this. By communicating an Identifier to C, A does not undertake to tell C how to reach B. Finding a Locator for B is out-of-band, and should be subject to B's security policies and whatever routing arrangements are in place between C and B.

We conclude that:

- On-site, Identifiers and Locators may or may not be split, depending on whether the completely or partially disjoint model is chosen.
- To go off-site, it is necessary to know which Locator leads towards a given Identifier. Either the sending Stack or its router could know this.
- The Identifier of the sending Stack must be delivered unchanged to the receiving Stack.
- It must be possible to refer third party Stacks to a remote Stack's Identifier.

Another likely common requirement is that whatever plays the part of the socket API should provide sufficient features for the upper layer protocol that it can deal with transitional situations, where some Stacks support the id-loc split and others don't. In the case of partially disjoint namespaces, the API must provide features to allow the upper layer protocol to discriminate between Locators and Identifiers if necessary.

4. Mapping Goals

With the above as background we can consider the goals of a mapping mechanism between Identifier Namespace and Locator Namespaces (note the plural).

It is assumed that application Namespaces, and in particular domain names, will be mapped to Identifiers independently, for example using existing or new DNS RR types.

4.1. Many Locator Namespaces

Every site using [RFC1918] has its own Locator Namespace. The IPv4 Internet constitutes another Locator Namespace, and IPv6 constitutes a Locator Namespace. Furthermore, sites using IPv6 ULAs [RFC4193] will have their own Locator Namespaces. We cannot exclude the existence of additional (possibly secret) Locator Namespaces.

4.2. One Identifier Namespace

Going forward, we should aim to have exactly one Identifier Namespace at network level. In different contexts, it will be mapped to different Locator Namespaces.

4.3. Reversible Mapping Issues

Clearly mapping from an Identifier to a set of Locators must be possible. Since we require every packet to be delivered carrying its original Identifier, reverse mapping would be needed if packets do not carry their original Identifier en route. This has important implications. If reverse mapping is not required, it would be quite possible for the Locator used for transport across the Internet to be highly multiplexed - in the limit, all packets sent to a given site could be sent to the same Locator. (For an analogy, compare the way that all IPv6 packets for a 6to4 site are sent to the same IPv4 address [RFC3056]). Such a simpification could have a dramatic effect on the size and frequency of mapping updates.

4.4. Two-Faced Maps?

Since the Locator Namespace within a site is in the general case different from the Internet Locator Namespace, there could in theory be a need for two maps at a given site: Identifier Namespace to the site's Locator Namespace, and Identifier Namespace to the Internet Locator Namespace (plus some complications due to IPv4 and IPv6 being different Internet Locator Namespaces). Two-faced maps can be avoided only if we choose the partially disjoint option and deem that within the site, Identifier equals Locator.

4.5. Is the Map Isotropic?

The short answer is: highly unlikely. It is not true by construction that if the best Locator for B when sending from A has value L(A,B), and the best Locator for B when sending from C has value L(C,B), then

L(A,B)=L(C,B). In fact, it is not true by construction that L(A,B) is necessarily a valid locator when sending from C to B. A trivial example is when L(A,B) is 10.1.1.1/32. It may be true that B has some Locators that are globally valid, but this cannot be assumed unless the system is designed that way.

The map will therefore almost certainly have some of the properties of a routing system - the entries for a given Identifier will be different at different points in the topology.

4.6. Scale

We want no arbitrary scaling limits. However, proposed scaling targets are 10 to 100 billion Stacks (which scales the Identifier Namespace), and 10 million sites. Although the latter does not directly scale the Internet's Locator Namespace, it indicates the worst-case granularity of the routing table for that Locator Namespace. If we don't do better than random allocation of address blocks to sites, we will end up with 10 million routing table entries.

4.7. What Does an Identifier Look Like?

This is of course a question that HIP [RFC4423] has looked at, although focussed on "hosts" rather than Stacks. In practice, the difference is probably not so important - a Stack can be viewed as a virtual host - and HIP in fact settled on using a HIT (Host Identity Tag) as a 128-bit representation for a Host Identity in actual packets. HIP further posits that "a HIT should be unique in the whole IP universe as long as it is being used." Whether or not we accept the HIP/HIT model, in the partially disjoint model, choosing near-128 bit opaque binary objects as the baseline for Stack Identifiers seems like the obvious conclusion, and can easily be rendered compatible with IPv6 or embedded IPv4 addresses. In the fully disjoint model, the format of an Identifier is currently unconstrained.

4.8. Do Identifiers have Useful Properties?

Thus far we assumed that Identifiers are opaque binary objects. Is this a correct assumption? We may want them to be designed for efficient lookup, and we may discover cryptographic requirements. Both architectural and design decisions need to be taken on this point.

If they are truly opaque (e.g., HITs) they have no structure, cannot be aggregated in any sort of hierarchy, and cannot be used for any kind of structured lookup. If used on-site as Locators, HITs force

use of flat routing. If Identifiers are (for example) equated locally to Locators, and specifically to IPv6 Unique Local Addresses (ULA) [RFC4193], they can be organized to match a site's subnet infrastructure and be used in a convenient way in on-site routing and in reverse DNS. However, for mobile systems, this argument can only apply on the home site; once the system roams to another site, its ULA can no longer be used as a Locator even it remains as the Identifier. Therefore, we must be cautious about relying on any address-like properties of Identifiers.

4.9. What Does a Locator Look Like?

Unless we plan to destroy the Internet and build a new one, it seems probable that it will look like an IPv4 or IPv6 prefix. Note that it doesn't need to be a full address: routing an IPv4 packet to the site border router handling a given /24 may be necessary and sufficient, if that border router knows how to resolve the Identifier to a local Stack Locator. So it is suggested that Locators in the map look like:

Address type:Prefix/MaskLength

and of course it is legitimate to use /32 or /128 masks.

4.10. Who Needs the Map?

The minimal model is that the map is needed at every border router that connects a customer site (or a local ISP) to one or more transit providers. In that case, Stack A will simply send a packet to Stack Identifier B in care of its default router, and the border router will invoke the map to forward the packet to a Locator for B's site. B's border router will remap, if necessary, to B's local Locator. (However, that remapping will be a no-operation if, in the partially disjoint model, Identifier equals stub Locator.)

In more complex scenarios, several Locator remappings could occur in transit. In other words, in addition to A's and B's border routers, which both sit at a point where the Locator Namespace Context changes, the packet flows through another router at which such a change occurs. Although this would represent a discontinuity in the Internet Locator Namespace, it would not break the Identifier/Locator mapping model. In fact, a Locator remapping can be regarded as a property of the routing system and quite distinct from the Identifier/Locator map. It's an open question whether a general Locator/Locator mapping system is needed.

Note that if sites A and B were actually interconnected by an IP-level VPN rather than by the Internet, this could readily be represented in the map held by the two border routers. It's really

just a different way of representing a specific route.

Finally, the map could be held by the sending hosts. In a sense this is what SHIM6 [I-D.ietf-shim6-proto] proposes, except that SHIM6 derives its map session-by-session without directly involving the routing system. But SHIM6 could certainly work equally well if it acquired the map globally from some source, and moving SHIM6 functionality from the host to a proxy is being studied.

4.11. What is the lifetime of a mapping?

At a given instant in time, an Identifier will correspond to one or more Locators. The mapping between a given Identifier and a given Locator must have some valid lifetime, which is unlikely to be infinite. For example, imagine a server with a permanent Identifier. Assume it has four corresponding Locators, the IPv6 and IPv4 addresses of the site's border routers, which are connected to two different ISPs. When the site decides to cancel one of its ISPs, the two corresponding locators become invalid.

Lifetime could be much shorter for client machines configured dynamically each time they connect to a site network, using DHCP or IPv6 Neighbor Discovery. When such a machine boots up, it might acquire a new Identifier, possibly equal to its Locator in the site context. This will need to be mapped to all external Locators valid for the site. That mapping will be valid only until the client machine is removed from the site network. Even if there is a way to give a client machine a permanent Identifier, its Locator mappings will be created at reboot.

How quickly will such changes in the map be propagated through the Internet? Will old mappings be deleted from cache when a lifetime expires, or will there be an explicit delete?

4.12. How Does the Map Relate to Mobility?

Is it correct, and sufficient, to say that whatever Locator a mobile system is currently using belongs to some site or other, and the Identifier/Locator mapping for that site applies? In other words, can we deem that id-loc mapping is orthogonal to mobility? Or must we ask how a site discovers the appropriate mapping when a visiting mobile node appears? Alternatively, should we consider that dynamic id-loc mapping is itself a mobility mechanism? What lessons can we learn from Mobile IP? In fact, is Mobile IP anything other than an id-loc mapping system, where the Home Agent fulfils the role of an id-loc mapper?

4.13. Who chooses the Locator?

Because of multihoming, it's likely that the map will contain more than one Locator for a given Identifier. An important consideration is that the choice of locator should be under policy control, in order that traffic should flow along the paths preferred by site or ISP managers. The map needs to provide for some elementary policy constructs such as precedence.

4.14. Push, Pull, Push/Pull

In a naive push model, the originator of a mapping entry pushes it out to all those who may need it, which in the limiting case is every node in the Internet. In a naive pull model, a node that needs a mapping entry (to map the first packet of a session) pulls the entry when it needs it. Clearly, as described, both of these models have major issues.

The naive push model would need to flood the entire Internet whenever a mapping entry changes. Even if we can design to have one mapping entry per site, that makes a target of flooding the Internet with updates from 10 million sites. The rate is unknown, but presumably many times today's BGP4 update rate, and presumably unsustainable.

The naive pull model would insert a substantial lookup delay (probably measured in tens or hundreds of milliseconds) at the beginning of every applications session that could not make use of a locally cached mapping entry. This would be a major problem, particularly for massive server farms serving tens of thousands of effectively random Internet users, where the chance of having a suitable cached mapping entry would be low. Also, it would cause a risk of unacceptable glitches in real time sessions, if dynamic remapping became necessary during a session.

The answer probably lies in some intermediate Push/Pull model where mapping changes are opportunistically flooded to caching map servers around the network, and opportunistically discovered by mapping nodes, with a fallback to a pull model when needed. While the DNS may serve as a general model for such a solution, it is far from obvious that the DNS is the right tool as it stands today.

5. Conclusion

This document doesn't draw a conclusion as to whether the completely disjoint or partially disjoint namespaces model is better, but the community needs to make a clear decision. Neither does it analyze map distribution mechanisms in detail, where again a community choice

of mechanism is needed. It is hoped the foregoing discussion will provide background for the choices to be made.

6. Security Considerations

An important decision must be made whether the mapping mechanism will exist only in boxes deemed to be intrinsically trusted (i.e. routers accessible exclusively by trusted personnel) or whether it will also exist in boxes liable to general attack threats (i.e. hosts accessible to a wide variety of users and not necessarily maintained professionally). The threat analysis for a solution will be significantly different in the two cases.

This document does not attempt a threat analysis in a vacuum. It is clear that if Internet routing comes to depend on an Identifier to Locator mapping service, that service could become an attack vector for either packet diversion or denial of service unless adequately protected. Thus, it seems very likely that the mapping elements must be cryptographicaly authenticated to prevent tampering, and possible DoS attacks must be anticipated. The threat analyses for MULTI6 [RFC4218], SHIM6 (see Security Considerations in [I-D.ietf-shim6-proto]) and LISP [I-D.bagnulo-lisp-threat] are illustrative.

A major purpose of a Stack Identifier is to give assurance to the other end of a communication that it's talking to the right entity. There are various cases in current-day TCP/IP where the IP Address is used for that, on the theory that we know the packets we send are being directed there; we should be looking for something stronger. This implies a possible need for Identifiers to be authenticated, regardless of mapping issues.

Any id-loc mapping scheme will have an impact on privacy, e.g. facilitating or hindering the tracking of an individual's or host's activities over time. Whatever id-loc schemes are proposed should be specific about their impact in this area. Also, is there any relation to topology hiding [RFC4864]?

7. IANA Considerations

This document requests no action by the IANA.

8. Acknowledgements

There are years of previous thinking and writing on this topic in the

Internet technical community. No claim to originality is made, and overlooked references will be gladly added.

Contributions and comments by Eliot Lear, Russ White, Stephen Farrell, Noel Chiappa, John Leslie, Alvaro Vives and others are gratefully acknowledged.

This document was produced using the xml2rfc tool [RFC2629].

9. Change log [RFC Editor: please remove this section]

draft-carpenter-idloc-map-cons-01: significant reorganization, especially distinguishing disjoint and overlapping namespace models; included feedback comments, 2007-06-04

draft-carpenter-idloc-map-cons-00: original version, 2007-04-17

10. Informative References

[ENDPOINTS] Chiappa, N., "Endpoints and Endpoint Names: A Proposed Enhancement to the Internet Architecture [unpublished, see http://ana.lcs.mit.edu/~jnc//tech/endpoints.txt]", June 1995.

[GSE] O'Dell, M., "GSE - An Alternate Addressing Architecture for IPv6 [unpublished]", February 1997.

[I-D.bagnulo-lisp-threat] Bagnulo, M., "Preliminary LISP Threat Analysis", draft-bagnulo-lisp-threat-00 (work in progress), March 2007.

[I-D.farinacci-lisp] Farinacci, D., "Locator/ID Separation Protocol (LISP)", draft-farinacci-lisp-00 (work in progress), January 2007.

[I-D.iab-raws-report] Meyers, D., "Report from the IAB Workshop on Routing and Addressing", draft-iab-raws-report-02 (work in progress), April 2007.

[I-D.ietf-mmusic-ice] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Offer/Answer Protocols",

draft-ietf-mmusic-ice-15 (work in progress), March 2007.

[I-D.ietf-shim6-proto]
Bagnulo, M. and E. Nordmark, "Shim6: Level 3 Multihoming Shim Protocol for IPv6", draft-ietf-shim6-proto-08 (work in progress), April 2007.

[I-D.nikander-ram-ilse]
Nikander, P., "Identifier / Locator Separation: Exploration of the Design Space (ILSE)", draft-nikander-ram-ilse-00 (work in progress), February 2007.

[I-D.templin-ipvlx]
Templin, F., "The IPvLX Architecture", draft-templin-ipvlx-08 (work in progress), May 2007.

[I-D.wang-ietf-efit]
Massey, D., "A Proposal for Scalable Internet Routing & Addressing", draft-wang-ietf-efit-00 (work in progress), February 2007.

[NSRG] Lear, E. and R. Droms, "What's In A Name: Thoughts from the NSRG [unpublished]", September 2003.

[RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC2101] Carpenter, B., Crowcroft, J., and Y. Rekhter, "IPv4 Address Behaviour Today", RFC 2101, February 1997.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC3056] Carpenter, B. and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds", RFC 3056, February 2001.

[RFC3205] Moore, K., "On the use of HTTP as a Substrate", BCP 56, RFC 3205, February 2002.

[RFC3489] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003.

[RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and B. Zill, "IPv6 Scoped Address Architecture", RFC 4007,

March 2005.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.

[RFC4218] Nordmark, E. and T. Li, "Threats Relating to IPv6 Multihoming Solutions", RFC 4218, October 2005.

[RFC4423] Moskowitz, R. and P. Nikander, "Host Identity Protocol (HIP) Architecture", RFC 4423, May 2006.

[RFC4864] Van de Velde, G., Hain, T., Droms, R., Carpenter, B., and E. Klein, "Local Network Protection for IPv6", RFC 4864, May 2007.

Author's Address

Brian Carpenter
IBM
8 Chemin de Blandonnet
1214 Vernier,
Switzerland

Email: brian.e.carpenter@gmail.com

Full Copyright Statement

Intellectual Property

Acknowledgment

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