Network Working Group
Internet-Draft
Obsoletes: 2765 (if approved)
Intended status: Standards Track
Expires: January 8, 2010

D. Thaler, Ed.
Microsoft

July 7, 2009

IPv6 Addressing of IPv6/IPv4 Translators
draft-thaler-behave-translator-addressing-00.txt

Status of this Memo

Copyright Notice

Abstract

This document discusses how an individual IPv6 address can be algorithmically translated to a corresponding IPv4 address, and vice

versa, using only statically configured information. This technique is used in IPv6/IPv4 translators, as well as other types of proxies and gateways (e.g., for DNS) used in IPv6/IPv4 scenarios.

Table of Contents

1. Introduction

This document is part of a series of IPv6/IPv4 translation documents. A framework for IPv6/IPv4 translation is discussed in [I-D.baker-behave-v4v6-framework], including a taxonomy of scenarios that will be used in this document. Other documents specify the behavior of various types of translators and gateways, including mechanisms for translating between IP headers and other types of messages that include IP addresses. This document specifies how an individual IPv6 address is translated to a corresponding IPv4 address, and vice versa, in cases where an algorithmic mapping is used. While specific types of devices are used herein as examples, it is the responsibility of the specification of such devices to reference this document for algorithmic mapping of the addresses themselves.

In this document, an "IPv6/IPv4 translator" is an entity that translates IPv6 packets to IPv4 packets, and vice versa. It may do "stateless" translation, meaning that there is no per-flow state required, or "stateful" translation where per-flow state is created when the first packet in a flow is received.

In this document, an "address translator" is any entity that has to derive an IPv6 address from an IPv4 address or vice versa. This applies not only to devices that do IPv6/IPv4 packet translation, but also to other entities that manipulate addresses, such as name resolution proxies (e.g., DNS64 [I-D.bagnulo-behave-dns64]) and possibly other types of Application Layer Gateways (ALGs).

[OPEN ISSUE: addressing for encapsulation mechanisms such as Dual-Stack Lite, including use of port ranges, is currently treated as out of scope for this document. Should such addressing be in scope?]

In choosing a mapping between IPv6 and IPv4 addresses for a given scenario, there are two separate choices to make: choosing an IPv6 prefix, and choosing an algorithm for constructing the remainder of the IPv6 address. These two topics are covered in Section 2 and Section 3, respectively.

Algorithmic derivation of an IPv6 address from an IPv4 address, or vice versa, is used with two different classes of IPv6 addresses, discussed in Section 1.1 and Section 1.2.

1.1. IPv6 Addresses Assigned to IPv6 Hosts

In an IPv6 network, IPv6 addresses are assigned to IPv6 hosts, and these IPv6 addresses need to be translated to IPv4 addresses that can be used by IPv4 hosts. Such IPv4 addresses are not assigned to a

host, but packets destined to such addresses are routed to an appropriate IPv6/IPv4 translator that handles a set of such IPv4 addresses. Thus stateless address translation mechanisms typically put constraints on what IPv6 addresses can be assigned to IPv6 hosts that want to communicate with IPv4 destinations using an algorithmic mapping (although such hosts may have other IPv6 addresses used for other purposes). Without such constraints, stateful translation on such addresses must be done instead, which typically only supports initiation from the IPv6 side, and does not result in stable addresses that can be used in DNS and other protocols and applications that do not deal well with highly dynamic addresses.

IPv6 addresses assigned to IPv6 hosts for use with stateless translation are referred to as "IPv4-translatable" IPv6 addresses in [RFC2765] although that term is also used to refer to a specific address format (defined in [RFC2765] section 2.1) and hence we avoid the use of this term herein except when referring to the specific address format.

1.2. IPv6 Addresses Used For IPv4 Hosts

In an IPv4 network, IPv4 addresses are assigned to IPv4 hosts, and these IPv4 addresses need to be translated to IPv6 addresses that can be used by IPv6 hosts. Such IPv6 addresses are not assigned to a host, but packets destined to such addresses are routed to an appropriate IPv6/IPv4 translator that handles a set of such IPv6 addresses. Typically there are no additional constraints put on the IPv4 addresses. Unlike the addresses discussed in Section 1.1, algorithmic mapping of IPv6 addresses used for IPv4 hosts is used with both stateful and stateless translation.

IPv6 addresses used for IPv4 hosts are referred to as "IPv4-mapped" IPv6 addresses in [RFC2765] although that term is also used to refer to a specific address format (defined in [RFC2765] section 2.1) and hence we avoid the use of this term herein except when referring to the specific address format.

2. Prefix Selection

This section discusses the choice of IPv6 prefixes for use with the two classes of addresses outlined above.

2.1. Requirements

There are a number of important requirements to be considered.

2.1.1. IPv6 Routing System Scalability

One critical issue to consider when understanding the impact of the prefix used is related to the scalability of the IPv6 routing system. Since a goal of translation is to enable IPv4 only-hosts and IPv6-only nodes to communicate, the IPv6 addresses used for IPv4 hosts need to be routable within the IPv6 network served by the IPv6/IPv4 translator. This implies that a prefix covering such addresses must be covered by one or more routes in the IPv6 network. In some scenarios a single IPv6 route may suffice, whereas other scenarios may require multiple more specific routes. It is important, however, to avoid injecting an IPv6 route for every IPv4 route in the Internet.

2.1.2. Native Connectivity Preference for Dual-Stack Nodes

When dual stack nodes are involved in communication, a potential issue arises if they prefer translated IPv6/IPv4 connectivity over native IPv4 or IPv6 connectivity.

For communication initiated from an IPv6-only node towards a dual-stack node, there are two possibilities: communicating to the native IPv6 address of the destination, or communicating via an IPv6/IPv4 translator to the IPv4 address of the destination (by using an IPv6 address that is translated to the IPv4 address). In some cases this may be solved by having the IPv6-only node only learn the native IPv6 address and not the algorithmically derived one (e.g., by using a normal DNS resolver), but this is not possible in general due to the variety of mechanisms for learning the destination addresses (e.g., referrals). To cover those cases, an IPv6-only node SHOULD be able to distinguish a native IPv6 address from an IPv6 address used for an IPv4 host.

For communication initiated from a dual-stack node toward an IPv4-only node, there are two possibilities: communicating to the native IPv4 address of the destination, or communicating via an IPv6/IPv4 translator to the IPv4 address of the destination (by using an IPv6 address that is translated to the IPv4 address). In some cases this may be solved by having the dual-stack node only learn the native IPv4 address and not the algorithmically derived one, but this is not possible in general due to the variety of mechanisms for learning the destination addresses. Normally it is desirable for a dual-stack node to prefer an IPv6 destination address over an IPv4 destination address, but in this case the IPv6 destination address is worse because it will be translated. Hence in general, a dual-stack node SHOULD be able to distinguish a native IPv6 address from an IPv6 address used for an IPv4 host, so that the former can be preferred over an IPv4 destination address but not the latter.

[RFC3484] today provides the ability for IPv6-capable hosts to be configured with prefixes used for address sorting sufficient to solve the native connectivity preference issue. However the gap today is that this requires manual configuration of all such hosts, which is not practical.

2.1.3. Referral Support

A referral operation is when a host A passes the IP address of a Host B to a third Host C as application data. The Host C may then initiate communication towards Host B using the IP address received.

At this point in time, there are several widely-available protocols that operate on the IPv4 Internet and perform referrals, including HTTP, DNS, SIP, and BitTorrent. The analysis in [I-D.wing-behave-nat64-referrals] of SIP (which does referrals between IPv4 and IPv6) shows that SIP needs to refer IPv4 addresses, not IPv6 addresses. Thus, it doesn't matter what IPv6 prefix is used because an IPv6 address isn't referred.

[EDITOR'S NOTE: The wing document discusses BitTorrent but the text pulled above from draft-xli-behave-v4v6prefix section 5.1.2.1 only summarizes SIP. Furthermore, it doesn't cover other widely-deployed protocols that do referrals such as HTTP or DNS and hence is inconclusive. Finally, it's unclear whether it applies to IPv6 addresses of IPv4 hosts or of IPv6 hosts, or both. Hence, referral support is still considered an OPEN ISSUE. Another OPEN ISSUE is whether to incorporate text from draft-wing-behave-nat64-referrals into this document.]

2.1.4. Support for Multiple IPv6/IPv4 Translators

For IPv6 addresses assigned to IPv6 hosts, the choice of translator used in the IPv4-to-IPv6 direction is determined by the IPv4 address it is mapped to, rather than by the choice of IPv6 prefix.

For IPv6 addresses used for IPv4 hosts, the choice of translator used in the IPv6-to-IPv4 direction is determined by the IPv6 address, but is somewhat orthogonal to the choice of prefix. In general, it is possible to use a single IPv6 prefix for multiple IPv6/IPv4 translators, or different prefixes for different IPv6/IPv4 translators. Using different prefixes can be achieved by inserting additional subnet bits after the a common prefix such that each IPv6/IPv4 translator uses a separate range of subnets. Often only the common prefix needs to be configured (as opposed to each more-specific prefix) in other types of address translators such as DNS64 devices.

2.1.5. Appropriate Prefix Length

This section discusses several issues related to prefix length selection.

2.1.5.1. Routing Policy

[EDITOR'S NOTE: The text below needs to be rewritten somewhat. It's currently hard to follow, and is missing justification for its statements.]

The major issue for selecting the prefix length is the routing policy. If IPv4/IPv6 translation is implemented within a subnet, then a /96 should be fine. However, if IPv4/IPv6 translation is implemented in an ISP's backbone, then the minimum prefix should be /64 and in some cases should be /48.

2.1.5.2. Forwarding Efficiency

According to current specifications [EDITOR'S NOTE: need reference], routers must handle routes containing prefixes of any valid length, from 0 to 128. However, some users have reported that routers exhibit worse performance when routing using prefixes longer than 80 bits. This implies that using routes with prefixes of 80 bits or fewer would result in better performance in some cases.

2.1.5.3. EUI-64 Format

The use of a prefix length longer than 64 bits may affect the Interface Identifier format. Specifically, [RFC4291] section 2.5.1 requires that for all unicast addresses, except those that start with the binary value 000, Interface IDs are required to be 64 bits long and to be constructed in Modified EUI-64 format. While any method can be used to construct a Modified EUI-64, the format requires the 71st bit (the universal/local bit) to be set with specific meaning, and hence it is not available for use by an address format unless the prefix starts with binary 000.

Furthermore, for IPv6 addresses assigned to IPv6 hosts, the prefix must be 64 bits or less in order to work with stateless address autoconfiguration [RFC4862] on most links.

2.1.6. Uniqueness

An IPv6 address used for an IPv4 host must be unique (i.e., not ambiguously map to multiple possible IPv4 hosts) within the IPv6 network that can use that address. For example, since private IPv4 addresses can be reused within multiple IPv4 networks, an IPv6

network that connects to multiple such IPv4 networks cannot rely on the IPv4 address itself providing uniqueness.

2.2. Types of Prefixes

A prefix may be intended for use in IPv6 addresses assigned to IPv6 hosts, or for use in IPv6 addresses used for IPv4 hosts. In either case, there are two types of prefixes that can be used.

2.2.1. Network-Specific Prefix

IPv6 prefixes are assigned to a network operator by its regional internet registrar (RIR). From an IPv6 prefix assigned to the operator, the operator chooses a longer prefix for use by the operator's translator(s). Hence a given IPv4 address would have different IPv6 representations in different networks that use different prefixes. A network-specific prefix is also known as a Local Internet Registry (LIR) prefix.

If the network operator is an ISP that has been allocated a /32 or shorter, then it may be possible for the ISP to allocate a fairly short prefix such as a /40. However, if the network operator is an end site with a /48, then the prefixes for the translator would be much longer, such as a /56. Using a /56 prefix still leaves 24 bits that can be used (without routing on prefixes longer than 80 bits) for routes via different translators. However, since a prefix that originates from an RIR will not start with binary 000, the use of the 71st bit cannot be used by any format with a network-specific prefix.

2.2.2. Well-Known Prefix

A well-known prefix is an IPv6 prefix assigned by IANA for use in an algorithmic mapping. Hence a given IPv4 address would have the same IPv6 representation in all networks that use the same well-known prefix.

Rather than requiring manual configuration of the IPv6 prefixes in each device in a network (and for each network to which a given device can connect), a well-known prefix can be implemented by default until there exists a protocol to learn the policies. Using a network-specific prefix allows no such factory defaults.

Another benefit of a well-known prefix over a network-specific prefix is that a short prefix length can be used, allowing greater flexibility in the choice of format and support for multiple translators, while not requiring routing on prefixes longer than 80 bits.

Finally, a well-known prefix can start with binary 000, allowing the use of all subsequent bits.

Two examples of well-known prefixes, as specified in [RFC2765] section 2.1, are:

- IPv4-mapped: This prefix is 0:0:0:0:0:ffff::/96, and addresses in this prefix are used for IPv4 hosts.
- IPv4-translatable: This prefix is 0:0:0:0:ffff:0::/96, and addresses in this prefix are assigned to IPv6 hosts. However, since this prefix is longer than /64, this prefix does not work with stateless address autoconfiguration. Hence some other mechanism for configuring IPv6 hosts must be used with this prefix.

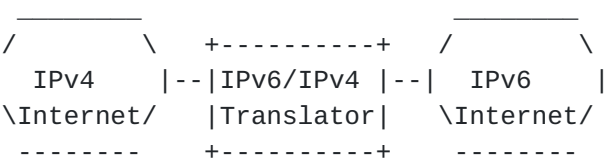
Note that both of the above prefixes start with binary 000, and hence there is no issue with the 71st bit.

2.3. Scenario-Specific Discussion and Recommendations

In this section we discuss four topologies in which IPv6/IPv4 translation is interesting. In each topology, initiating communication can be attempted from either the IPv6 side or the IPv4 side, though it may or may not be supported.

2.3.1. Connecting the IPv6 Internet to the IPv4 Internet

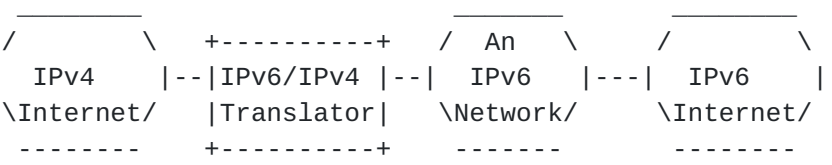
This scenario need not be supported for communication initiated from either side.



2.3.2. Connecting an IPv6 network to the IPv4 Internet

[EDITOR'S NOTE: draft-xli-behave-v4v6-prefix-00 section 5.1.1.2 was hard to follow, and contained some statements that got significant pushback on the list. This section tries to take these into account, but there may still be some points missing.]

The figure below shows an IPv6 network connected to the IPv6 Internet, and also to the IPv4 network via an IPv6/IPv4 translator.



For the IPv6 prefix used for IPv4 hosts, all IPv6-capable devices served by the translator SHOULD be configurable with preference policies consistent with [RFC3484], and SHOULD default to having the Well-Known Mapped Prefix defined in Section 5 in this table, with a lower preference than either native IPv4 or native IPv6 addresses.

Similarly, all address translators MUST be configurable with the IPv6 prefix to use for IPv4 hosts, and SHOULD use the Well-Known Mapped Prefix unless configured with a network-specific prefix.

When any well-known prefix is used by a given network, it MUST NOT be advertised into the IPv6 Internet, to prevent pollution of the global IPv6 routing table by elements of the IPv4 routing table. Therefore, a site which also has a native IPv6 connection MUST NOT advertise a well-known prefix on that connection, and native IPv6 network operators MUST filter out and discard any prefix routing advertisements for the well-known prefix.

Furthermore, to avoid being used as transit, the IPv6 network should not advertise into the IPv6 Internet the IPv6 prefix used for IPv4 hosts, regardless of whether it is network-specific or well-known. This is easy to ensure when the IPv6 prefix used for IPv4 hosts is disjoint from the IPv6 prefix used for IPv6 hosts.

For the IPv6 prefix used to assign addresses to IPv6 hosts, the use differs between stateless and stateful translation.

2.3.2.1. Stateful Translation

When stateful translation is used, the choice of IPv6 prefix for addresses assigned to IPv6 hosts is unaffected, since algorithmic mapping is not used for these addresses.

2.3.2.2. Stateless Translation

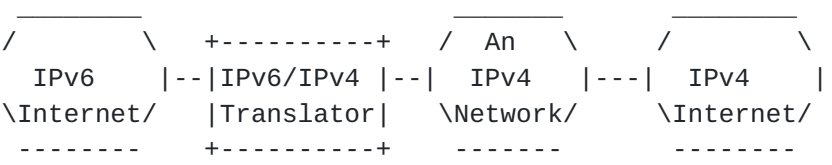
"An IPv6 network" will advertise to the IPv6/IPv4 translator the prefix for IPv6 addresses assigned to IPv6 hosts (and similarly the IPv6/IPv4 translator will advertise into "an IPv6 network" the IPv6 prefix used for IPv4 hosts).

On the other side, towards the IPv6 Internet, the IPv6 network advertises into the IPv6 Internet an IPv6 prefix for IPv6 addresses

assigned to IPv6 hosts. This prefix, used to be reachable from the IPv6 Internet, may or may not be the same as the prefix used to assign to hosts IPv6 addresses that are reachable from the IPv4 Internet, but it seems simplest to only require one prefix (and hence one global IPv6 address per host).

2.3.3. Connecting an IPv4 network to the IPv6 Internet

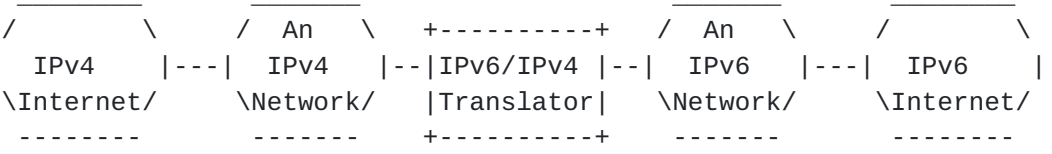
For this scenario, shown in the following figure, only stateful translation can be used in general, and an algorithmic mapping is not relevant for IPv6 addresses assigned to IPv6 hosts since they are not under the control of the same organization as the translator. (Note that algorithmic mapping can be used if communication with only a small subset of the IPv6 Internet is supported. That scenario is covered in Section 2.3.4.)



For IPv6 addresses used for the IPv4 hosts, the following considerations apply. If the IPv4 network uses private IPv4 addresses, a well-known prefix will not work since there is no distinction among IPv4 networks using the same private IPv4 address block. Therefore, a network-specific prefix MUST be used. If the IPv4 network uses public IPv4 addresses, it will inject a route into the IPv6 routing table pointing to its translator(s). Hence the routing scalability requirement requires that an IPv6 network-specific prefix again MUST be used.

2.3.4. Connecting between an IPv4 network and an IPv6 network

TO BE FILLED IN



3. IPv6 Address Format and Translation Algorithms

There are multiple mechanisms used today to algorithmically map between an IPv6 address and an IPv4 address, and more may be defined over time. In this section, we first present a set of requirements

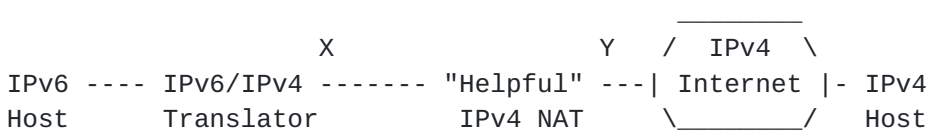
for such mechanisms, and then evaluate a number of existing mechanisms.

3.1. Requirements

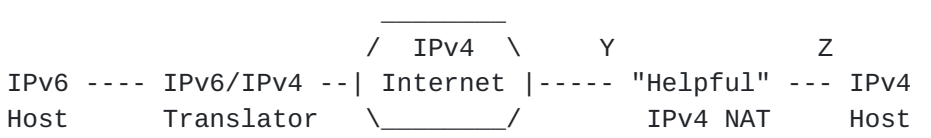
1. **An algorithm MUST be one-to-one and reversible.**
2. Unless the prefix starts with binary 000, an address format MUST NOT use the 71st bit for any purpose other than indicating universal/local as specified in [RFC4291],
3. An address format SHOULD provide support for multiple IPv6/IPv4 translators using different routes advertised into the IPv6 network (and different routes advertised into the IPv4 network).
4. An address format intended to be used with a stateless translator SHOULD be checksum-neutral. That is, the IPv6 address and its corresponding IPv4 address should result in the same one's complement checksum to avoid having to parse or modify the transport header. Simply relying on an administrator to choose a checksum-neutral prefix is tricky and hence error-prone. An algorithm that automatically compensates no matter what the administrator types is less harmful that one that does not. Note that checksum-neutral translation only benefits stateless translators that maintain a one-to-one mapping between an IPv4 address and an IPv6 address, since otherwise it has to have transport-specific behavior anyway.
5. For ease of readability and debugging, an address format that is not designed to intentionally hide the IPv4 address SHOULD allow accepting and/or displaying an embedded IPv4 address in dotted-decimal form. This is done today for a variety of address formats (e.g., IPv4-mapped and IPv4-translatable addresses as discussed below, IPv4-compatible addresses which were deprecated by [RFC4291] Section 2.5.5.1, and ISATAP [RFC5214] addresses). Per [RFC4291] Section 2.2, this can only be done when the embedded IPv4 address appears in the low-order 32 bits. It is not done when an IPv4 address is embedded elsewhere in the address (as in a 6to4 address). Displaying an address using dotted-decimal can only be done when some other portion of the IPv6 address is used to indicate displaying the low-order 32 bits in dotted-decimal form.
6. When the IPv4 network is a private network for which the topology is considered sensitive information, the algorithm SHOULD provide a way to hide the details of the internal IPv4 subnetting scheme. Note that there may be other mechanisms of discovering the topology beyond merely inspecting addresses, so while this is not sufficient in itself, it is a necessary component of any larger solution. Also note that providing this capability conflicts with requirement 3.
7. An algorithm MAY provide the ability to hide an IPv4 address from "helpful" IPv4 NATs. Consider the scenarios depicted below with IPv4 NATs that attempt to be "helpful" by looking for the NAT's

public IPv4 address in inbound application payloads and then translating it to a private IPv4 address, and similarly translating a private IPv4 address to the NAT's public IPv4 address in outbound payloads. While this can break applications since the same bytes that appear in the IPv4 address can appear normally in the payload purely by coincidence, the fact remains that many NATs have been observed to do this in an attempt to make other protocols work.

In the first scenario (an IPv6 address assigned to an IPv6 host), an IPv4 NAT has IPv4 public address Y, and an address translator uses IPv4 address X to map to an IPv6 address assigned to the IPv6 host. If the IPv6 host sends an application payload that includes an IPv6 address that directly embeds X (e.g., ::ffff:0:X) then this may be translated by the IPv4 NAT to ::ffff:0:Y, and similarly if the IPv4 host sends an application payload that includes ::ffff:0:Y then this may be translated by the IPv4 NAT to ::ffff:0:X. This may or may not be desirable.



In the second scenario (an IPv6 address used for an IPv4 host), the IPv4 NAT has public address Y, and the IPv4 host behind it has address Z. If the IPv6 host sends an application payload that includes an IPv6 address that directly embeds Y (e.g., ::ffff:Y) then this may be translated by the IPv4 NAT to ::ffff:Z, and similarly if the IPv4 host sends an application payload that includes ::ffff:Z then this may be translated by the IPv4 NAT to ::ffff:Y. This may or may not be desirable.



As a result, protocols such as Teredo [RFC4380] and STUN [RFC5389] today avoid such problems by obscuring the IPv4 address using XOR.

Note that providing this capability conflicts with requirement 3, but anything that meets requirement 4 will also meet this requirement.

3.2. Mechanisms

In this section we discuss a number of mechanisms for algorithmic mapping. [EDITOR'S NOTE: currently the subsections are ordered from most specific to most general. Would the opposite order be more or less readable?]

3.2.1. IPv4-Mapped IPv6 Addresses

IPv4-mapped IPv6 addresses are defined in [RFC4291] Section 2.5.5.2 as IPv6 addresses used for IPv4 hosts, using the format ::ffff: a.b.c.d, where a.b.c.d is the corresponding IPv4 address. IPv4-mapped IPv6 addresses are used both on the wire ([RFC2765]) when translation is done in the network, as well as within hosts (e.g., [RFC3493]) when translation is done in the end system. This format was designed to be checksum-neutral, and obviously uses a well-known prefix. It is widely deployed in host operating systems today.

3.2.2. IPv4-Translatable Addresses

IPv4-translatable addresses (also known as IPv4-translated addresses) are defined in [RFC2765] Section 2.1 as addresses assigned to IPv6 hosts, using the format ::ffff:0:a.b.c.d, where a.b.c.d is a corresponding IPv4 address used by a translator. IPv4-translatable addresses are used on the wire ([RFC2765]) when translation is done in the network. Hypothetically they could be used within hosts when translation is done in the end system, but there is no specification of this at present. This format was also designed to be checksum-neutral, and obviously uses a well-known-prefix. It is not known to be widely deployed today.

OPEN ISSUE: Should IPv4-translatable addresses be deprecated by this document, or should it continue to be used as the well-known default prefix for this purpose? For now, we assume it will continue to be used as the well-known default prefix for this purpose.

3.2.3. Zero-Pad And Embed

In this mechanism, the IPv4 address is placed in the last 32-bits, after a 96-bit configured prefix. That is, a well-known or network-specific prefix is zero-padded to 96 bits. This is referred to as PREFIX::/96 in the deprecated [RFC2766], resulting in addresses of the form PREFIX::a.b.c.d, where a.b.c.d is the corresponding IPv4 address. Note that typically the dotted-decimal form can only be used for input and in documentation, but not display since display would require the PREFIX to be known to all displaying systems to indicate the use of dotted-decimal.

This format is not checksum-neutral unless the PREFIX is checksum-neutral. Hence, a well-known prefix can ensure checksum neutrality, but using this format with network-specific prefixes in general cannot.

The universal/local bit in the Modified EUI-64 occurs in the prefix and MUST be set to zero unless the IPv4 address is known to be global (but can be set to zero even if it is known to be global).

Note that IPv4-mapped and IPv4-translatable addresses are a special case of this mechanism, where the PREFIX is well-known.

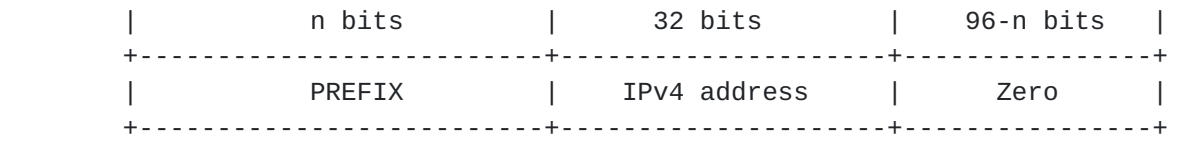
3.2.4. Compensation-Pad And Embed

This potential mechanism is the same as Zero-Pad And Embed (Section 3.2.3), except that it provides checksum neutrality and hence benefits stateless IPv6/IPv4 translators. A configured well-known or network-specific PREFIX::/80 is followed by 16 bits that result in the first 96 bits being checksum-neutral.

The universal/local bit in the Modified EUI-64 occurs in the prefix and MUST be set to zero unless the IPv4 address is known to be global (but can be set to zero even if it is known to be global).

3.2.5. Embed And Zero-Pad

In this mechanism (often referred to as "IVI"), the IPv4 address is embedded immediately after a routable prefix, and then zero-padded at the end.



[EDITOR'S NOTE: draft-baker-behave-v4v6-framework disallowed PREFIX being 64-95 bits long, without explanation. IPv6 addresses used for IPv4 hosts should be fine to use prefixes of other lengths. Hence

suggested clarifying text below. For IPv6 addresses assigned to IPv6 hosts, kept the restriction though I do not understand why this is needed and hence still consider this an OPEN ISSUE.]

The IPv4 address is intended to straddle the boundary between the prefix used in routable tables and the bits in the host portion. For IPv6 addresses assigned to IPv6 hosts, this would require a PREFIX of length 32..63 bits. For IPv6 addresses used for IPv4 hosts, any PREFIX length is sufficient.

However, if the PREFIX is a network-specific prefix, rather than a well-known prefix, this mechanism requires the prefix to be less than 38 bits (so that the IPv4 address would end prior to the universal/local bit), or at least 71 bits (so that the IPv4 address would begin after the universal/local bit).

Note that Zero-Pad and Embed can be considered to be a special case of this mechanism, where the PREFIX is 96 bits and the SUFFIX is 0 bits.

3.2.6. Preconfigured Mapping Table

In this, the most general, mechanism, an IPv6/IPv4 address translator is preconfigured with a mapping table including all legal pairs. Any IPv6 and IPv4 addresses can be used. This mechanism is not checksum-neutral. Since the IPv4 address is not embedded in any way, it need not reveal any details of the IPv4 topology, and minimizes issues with "helpful" IPv4 NATs.

3.3. Scenario-Specific Discussion and Recommendations

In this section we discuss four topologies in which IPv6/IPv4 translation is interesting. In each topology, initiating communication can be attempted from either the IPv6 side or the IPv4 side.

3.3.1. Connecting the IPv6 Internet to the IPv4 Internet

This scenario need not be supported.

3.3.2. Connecting an IPv6 network to the IPv4 Internet

Since the IPv4 network is the Internet, there is negligible value in trying to hide the topology details of the IPv4 network and hence requirement 4 does not apply. The other requirements all apply normally.

TO BE FILLED IN

3.3.3. Connecting an IPv4 network to the IPv6 Internet

In this scenario, only stateful translation can be used and hence requirement 2 does not apply. The other requirements all apply normally.

TO BE FILLED IN

3.3.4. Connecting between an IPv4 network and an IPv6 network

Typically the IPv4 network and the IPv6 network are managed by the same organization in this scenario, and hence it is not necessary to hide the topology details of the IPv4 network and requirement 4 does not apply in this case. The other requirements all apply normally.

TO BE FILLED IN

4. Security Considerations

The prefix and format need to be the same among multiple devices in the same network (e.g., hosts that need to prefer native over translated addresses, DNS gateways, and IPv6/IPv4 translators). As such, the means by which they are learned/configured must be secure. Specifying a default prefix and/or format in implementations provides one way to configure them securely. Any alternative means of configuration is responsible for specifying how to do so securely.

5. IANA Considerations

A future version of this memo will request an IPv6 prefix assignment as a Well-Known Mapped Prefix, that is used for IPv4 hosts, and which must start with binary 000.

[EDITOR'S NOTE: 0/8 is reserved by the IETF (and not allocated by IANA), so all that is needed is to specify the prefix herein since it is an allocation from IETF not from IANA.]

OPEN ISSUE: The prefix length of this block has not yet been determined. Some possibilities are /16, /32, /48 or /96.

6. Acknowledgements

Many people in the Behave WG have contributed to the discussion that led to this document, including Andrew Sullivan, Andrew Yourtchenko, Brian Carpenter, Congxiao Bao, Dan Wing, Ed Jankiewicz, Fred Baker,

Hiroshi Miyata, Iljitsch van Beijnum, John Schnizlein, Keith Moore, Kevin Yin, Magnus Westerlund, Marcelo Bagnulo Braun, Margaret Wasserman, Masahito Endo, Phil Roberts, Philip Matthews, Remi Denis-Courmont, Remi Despres, William Waites and Xing Li.

7. Contributors

The following individuals co-authored drafts from which text has been incorporated, and are listed in alphabetical order.

Congxiao Bao
CERNET Center/Tsinghua University
Room 225, Main Building, Tsinghua University
Beijing, 100084
China
Phone: +86 62785983
Email: congxiao@cernet.edu.cn

Fred Baker
Cisco Systems
Santa Barbara, California 93117
USA
Phone: +1-408-526-4257
Fax: +1-413-473-2403
Email: fred@cisco.com

Hiroshi Miyata
Yokogawa Electric Corporation
2-9-32 Nakacho
Musashino-shi, Tokyo 180-8750
JAPAN
Email: h.miyata@jp.yokogawa.com

Marcelo Bagnulo
Universidad Carlos III de Madrid
Av. Universidad 30
Leganes, Madrid 28911
ESPANA
Email: marcelo@it.uc3m.es

Xing Li
CERNET Center/Tsinghua University
Room 225, Main Building, Tsinghua University
Beijing, 100084
China
Phone: +86 62785983
Email: xing@cernet.edu.cn

8. References

8.1. Normative References

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

8.2. Informative References

[I-D.bagnulo-behave-dns64] Bagnulo, M., Sullivan, A., Matthews, P., Beijnum, I., and M. Endo, "DNS64: DNS extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", draft-bagnulo-behave-dns64-02 (work in progress), March 2009.

[I-D.baker-behave-v4v6-framework] Baker, F., Li, X., and C. Bao, "Framework for IPv4/IPv6 Translation", draft-baker-behave-v4v6-framework-02 (work in progress), February 2009.

[I-D.wing-behave-nat64-referrals] Wing, D., "Referrals Across a NAT64", draft-wing-behave-nat64-referrals-00 (work in progress), March 2009.

[RFC2765] Nordmark, E., "Stateless IP/ICMP Translation Algorithm (SIIT)", RFC 2765, February 2000.

[RFC2766] Tsirtsis, G. and P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)", RFC 2766, February 2000.

[RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.

[RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, February 2006.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless

Address Autoconfiguration", RFC 4862, September 2007.

[RFC5214] Templin, F., Gleeson, T., and D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", RFC 5214, March 2008.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

Author's Address

Dave Thaler (editor)
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
USA

Phone: +1 425 703 8835
Email: dthaler@microsoft.com