        Default Address Selection for Internet Protocol version 6 (IPv6)

Status of this Memo

    This document specifies an Internet standards track protocol for the
    Internet community, and requests discussion and suggestions for
    improvements. Please refer to the current edition of the "Internet
    Official Protocol Standards" (STD 1) for the standardization state
    and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

    This document describes two algorithms, for source address selection
    and for destination address selection. The algorithms specify
    default behavior for all Internet Protocol version 6 (IPv6)
    implementations. They do not override choices made by applications
    or upper-layer protocols, nor do they preclude the development of
    more advanced mechanisms for address selection. The two algorithms
    share a common context, including an optional mechanism for allowing
    administrators to provide policy that can override the default
    behavior. In dual stack implementations, the destination address
    selection algorithm can consider both IPv4 and IPv6 addresses -
    depending on the available source addresses, the algorithm might
    prefer IPv6 addresses over IPv4 addresses, or vice-versa.

    All IPv6 nodes, including both hosts and routers, must implement
    default address selection as defined in this specification.

Table of Contents

1. Introduction

   The IPv6 addressing architecture [1] allows multiple unicast
   addresses to be assigned to interfaces. These addresses may have
   different reachability scopes (link-local, site-local, or global).
   These addresses may also be "preferred" or "deprecated" [2]. Privacy
   considerations have introduced the concepts of "public addresses" and
   "temporary addresses" [3]. The mobility architecture introduces
   "home addresses" and "care-of addresses" [8]. In addition, multi-
   homing situations will result in more addresses per node. For
   example, a node may have multiple interfaces, some of them tunnels or
   virtual interfaces, or a site may have multiple ISP attachments with
   a global prefix per ISP.

   The end result is that IPv6 implementations will very often be faced
   with multiple possible source and destination addresses when
   initiating communication. It is desirable to have default

algorithms, common across all implementations, for selecting source
and destination addresses so that developers and administrators can
reason about and predict the behavior of their systems.

Furthermore, dual or hybrid stack implementations, which support both
IPv6 and IPv4, will very often need to choose between IPv6 and IPv4
when initiating communication. For example, when DNS name resolution
yields both IPv6 and IPv4 addresses and the network protocol stack
has available both IPv6 and IPv4 source addresses. In such cases, a
simple policy to always prefer IPv6 or always prefer IPv4 can produce
poor behavior. As one example, suppose a DNS name resolves to a
global IPv6 address and a global IPv4 address. If the node has
assigned a global IPv6 address and a 169.254/16 auto-configured IPv4
address [9], then IPv6 is the best choice for communication. But if
the node has assigned only a link-local IPv6 address and a global
IPv4 address, then IPv4 is the best choice for communication. The
destination address selection algorithm solves this with a unified
procedure for choosing among both IPv6 and IPv4 addresses.

The algorithms in this document are specified as a set of rules that
define a partial ordering on the set of addresses that are available
for use. In the case of source address selection, a node typically
has multiple addresses assigned to its interfaces, and the source
address ordering rules in section 5 define which address is the
"best" one to use. In the case of destination address selection, the
DNS may return a set of addresses for a given name, and an
application needs to decide which one to use first, and in what order
to try others should the first one not be reachable. The destination
address ordering rules in section 6, when applied to the set of
addresses returned by the DNS, provide such a recommended ordering.

This document specifies source address selection and destination
address selection separately, but using a common context so that
together the two algorithms yield useful results. The algorithms
attempt to choose source and destination addresses of appropriate
scope and configuration status (preferred or deprecated in the RFC
2462 sense). Furthermore, this document suggests a preferred method,
longest matching prefix, for choosing among otherwise equivalent
addresses in the absence of better information.

This document also specifies policy hooks to allow administrative
override of the default behavior. For example, using these hooks an
administrator can specify a preferred source prefix for use with a
destination prefix, or prefer destination addresses with one prefix
over addresses with another prefix. These hooks give an
administrator flexibility in dealing with some multi-homing and
transition scenarios, but they are certainly not a panacea.

The selection rules specified in this document MUST NOT be construed
to override an application or upper-layer's explicit choice of a
legal destination or source address.

1.1. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
document are to be interpreted as described in BCP 14, RFC 2119 [4].

2. Context in Which the Algorithms Operate

Our context for address selection derives from the most common
implementation architecture, which separates the choice of
destination address from the choice of source address. Consequently,
we have two separate algorithms for these tasks. The algorithms are
designed to work well together and they share a mechanism for
administrative policy override.

In this implementation architecture, applications use APIs [10] like
getaddrinfo() that return a list of addresses to the application.
This list might contain both IPv6 and IPv4 addresses (sometimes
represented as IPv4-mapped addresses). The application then passes a
destination address to the network stack with connect() or sendto().
The application would then typically try the first address in the
list, looping over the list of addresses until it finds a working
address. In any case, the network layer is never in a situation
where it needs to choose a destination address from several
alternatives. The application might also specify a source address
with bind(), but often the source address is left unspecified.
Therefore the network layer does often choose a source address from
several alternatives.

As a consequence, we intend that implementations of getaddrinfo()
will use the destination address selection algorithm specified here
to sort the list of IPv6 and IPv4 addresses that they return.
Separately, the IPv6 network layer will use the source address
selection algorithm when an application or upper-layer has not
specified a source address. Application of this specification to
source address selection in an IPv4 network layer may be possible but
this is not explored further here.

Well-behaved applications SHOULD iterate through the list of
addresses returned from getaddrinfo() until they find a working
address.

The algorithms use several criteria in making their decisions. The
combined effect is to prefer destination/source address pairs for
which the two addresses are of equal scope or type, prefer smaller
scopes over larger scopes for the destination address, prefer non-
deprecated source addresses, avoid the use of transitional addresses
when native addresses are available, and all else being equal prefer
address pairs having the longest possible common prefix. For source
address selection, public addresses [3] are preferred over temporary
addresses. In mobile situations [8], home addresses are preferred
over care-of addresses. If an address is simultaneously a home
address and a care-of address (indicating the mobile node is "at
home" for that address), then the home/care-of address is preferred
over addresses that are solely a home address or solely a care-of
address.

This specification optionally allows for the possibility of
administrative configuration of policy that can override the default
behavior of the algorithms. The policy override takes the form of a
configurable table that specifies precedence values and preferred
source prefixes for destination prefixes. If an implementation is
not configurable, or if an implementation has not been configured,
then the default policy table specified in this document SHOULD be
used.

2.1. Policy Table

The policy table is a longest-matching-prefix lookup table, much like
a routing table. Given an address A, a lookup in the policy table
produces two values: a precedence value Precedence(A) and a
classification or label Label(A).

The precedence value Precedence(A) is used for sorting destination
addresses. If Precedence(A) > Precedence(B), we say that address A
has higher precedence than address B, meaning that our algorithm will
prefer to sort destination address A before destination address B.

The label value Label(A) allows for policies that prefer a particular
source address prefix for use with a destination address prefix. The
algorithms prefer to use a source address S with a destination
address D if Label(S) = Label(D).

IPv6 implementations SHOULD support configurable address selection
via a mechanism at least as powerful as the policy tables defined
here. Note that at the time of this writing there is only limited
experience with the use of policies that select from a set of
possible IPv6 addresses. As more experience is gained, the
recommended default policies may change. Consequently it is
important that implementations provide a way to change the default

policies as more experience is gained. Sections 10.3 and 10.4
provide examples of the kind of changes that might be needed.

If an implementation is not configurable or has not been configured,
then it SHOULD operate according to the algorithms specified here in
conjunction with the following default policy table:

Prefix	Precedence	Label
::1/128	50	0
::/0	40	1
2002::/16	30	2
::/96	20	3
::ffff:0:0/96	10	4

One effect of the default policy table is to prefer using native
source addresses with native destination addresses, 6to4 [5] source
addresses with 6to4 destination addresses, and v4-compatible [1]
source addresses with v4-compatible destination addresses. Another
effect of the default policy table is to prefer communication using
IPv6 addresses to communication using IPv4 addresses, if matching
source addresses are available.

Policy table entries for scoped address prefixes MAY be qualified
with an optional zone index. If so, a prefix table entry only
matches against an address during a lookup if the zone index also
matches the address's zone index.

2.2. Common Prefix Length

We define the common prefix length CommonPrefixLen(A, B) of two
addresses A and B as the length of the longest prefix (looking at the
most significant, or leftmost, bits) that the two addresses have in
common. It ranges from 0 to 128.

3. Address Properties

In the rules given in later sections, addresses of different types
(e.g., IPv4, IPv6, multicast and unicast) are compared against each
other. Some of these address types have properties that aren't
directly comparable to each other. For example, IPv6 unicast
addresses can be "preferred" or "deprecated" [2], while IPv4
addresses have no such notion. To compare such addresses using the
ordering rules (e.g., to use "preferred" addresses in preference to
"deprecated" addresses), the following mappings are defined.

3.1. Scope Comparisons

Multicast destination addresses have a 4-bit scope field that controls the propagation of the multicast packet. The IPv6 addressing architecture defines scope field values for interface-local (0x1), link-local (0x2), subnet-local (0x3), admin-local (0x4), site-local (0x5), organization-local (0x8), and global (0xE) scopes [11].

Use of the source address selection algorithm in the presence of multicast destination addresses requires the comparison of a unicast address scope with a multicast address scope. We map unicast link-local to multicast link-local, unicast site-local to multicast site-local, and unicast global scope to multicast global scope. For example, unicast site-local is equal to multicast site-local, which is smaller than multicast organization-local, which is smaller than unicast global, which is equal to multicast global.

We write Scope(A) to mean the scope of address A. For example, if A is a link-local unicast address and B is a site-local multicast address, then Scope(A) < Scope(B).

This mapping implicitly conflates unicast site boundaries and multicast site boundaries [11].

3.2. IPv4 Addresses and IPv4-Mapped Addresses

The destination address selection algorithm operates on both IPv6 and IPv4 addresses. For this purpose, IPv4 addresses should be represented as IPv4-mapped addresses [1]. For example, to lookup the precedence or other attributes of an IPv4 address in the policy table, lookup the corresponding IPv4-mapped IPv6 address.

IPv4 addresses are assigned scopes as follows. IPv4 auto-configuration addresses [9], which have the prefix 169.254/16, are assigned link-local scope. IPv4 private addresses [12], which have the prefixes 10/8, 172.16/12, and 192.168/16, are assigned site-local scope. IPv4 loopback addresses [12, section 4.2.2.11], which have the prefix 127/8, are assigned link-local scope (analogously to the treatment of the IPv6 loopback address [11, section 4]). Other IPv4 addresses are assigned global scope.

IPv4 addresses should be treated as having "preferred" (in the RFC 2462 sense) configuration status.

3.3. Other IPv6 Addresses with Embedded IPv4 Addresses

IPv4-compatible addresses [1], IPv4-mapped [1], IPv4-translatable [6] and 6to4 addresses [5] contain an embedded IPv4 address. For the purposes of this document, these addresses should be treated as having global scope.

IPv4-compatible, IPv4-mapped, and IPv4-translatable addresses should be treated as having "preferred" (in the RFC 2462 sense) configuration status.

3.4. IPv6 Loopback Address and Other Format Prefixes

The loopback address should be treated as having link-local scope [11, section 4] and "preferred" (in the RFC 2462 sense) configuration status.

NSAP addresses and other addresses with as-yet-undefined format prefixes should be treated as having global scope and "preferred" (in the RFC 2462) configuration status. Later standards may supersede this treatment.

3.5. Mobility Addresses

Some nodes may support mobility using the concepts of a home address and a care-of address (for example see [8]). Conceptually, a home address is an IP address assigned to a mobile node and used as the permanent address of the mobile node. A care-of address is an IP address associated with a mobile node while visiting a foreign link. When a mobile node is on its home link, it may have an address that is simultaneously a home address and a care-of address.

For the purposes of this document, it is sufficient to know whether or not one's own addresses are designated as home addresses or care-of addresses. Whether or not an address should be designated a home address or care-of address is outside the scope of this document.

4. Candidate Source Addresses

The source address selection algorithm uses the concept of a "candidate set" of potential source addresses for a given destination address. The candidate set is the set of all addresses that could be used as a source address; the source address selection algorithm will pick an address out of that set. We write CandidateSource(A) to denote the candidate set for the address A.

It is RECOMMENDED that the candidate source addresses be the set of
unicast addresses assigned to the interface that will be used to send
to the destination. (The "outgoing" interface.) On routers, the
candidate set MAY include unicast addresses assigned to any interface
that forwards packets, subject to the restrictions described below.

   Discussion: The Neighbor Discovery Redirect mechanism [14]
   requires that routers verify that the source address of a packet
   identifies a neighbor before generating a Redirect, so it is
   advantageous for hosts to choose source addresses assigned to the
   outgoing interface. Implementations that wish to support the use
   of global source addresses assigned to a loopback interface should
   behave as if the loopback interface originates and forwards the
   packet.

In some cases the destination address may be qualified with a zone
index or other information that will constrain the candidate set.

For multicast and link-local destination addresses, the set of
candidate source addresses MUST only include addresses assigned to
interfaces belonging to the same link as the outgoing interface.

   Discussion: The restriction for multicast destination addresses
   is necessary because currently-deployed multicast forwarding
   algorithms use Reverse Path Forwarding (RPF) checks.

For site-local destination addresses, the set of candidate source
addresses MUST only include addresses assigned to interfaces
belonging to the same site as the outgoing interface.

In any case, anycast addresses, multicast addresses, and the
unspecified address MUST NOT be included in a candidate set.

If an application or upper-layer specifies a source address that is
not in the candidate set for the destination, then the network layer
MUST treat this as an error. The specified source address may
influence the candidate set, by affecting the choice of outgoing
interface. If the application or upper-layer specifies a source
address that is in the candidate set for the destination, then the
network layer MUST respect that choice. If the application or
upper-layer does not specify a source address, then the network layer
uses the source address selection algorithm specified in the next
section.

On IPv6-only nodes that support SIIT [6, especially section 5], if
the destination address is an IPv4-mapped address then the candidate
set MUST contain only IPv4-translatable addresses. If the

destination address is not an IPv4-mapped address, then the candidate
set MUST NOT contain IPv4-translatable addresses.

5. Source Address Selection

The source address selection algorithm produces as output a single
source address for use with a given destination address. This
algorithm only applies to IPv6 destination addresses, not IPv4
addresses.

The algorithm is specified here in terms of a list of pair-wise
comparison rules that (for a given destination address D) imposes a
"greater than" ordering on the addresses in the candidate set
CandidateSource(D). The address at the front of the list after the
algorithm completes is the one the algorithm selects.

Note that conceptually, a sort of the candidate set is being
performed, where a set of rules define the ordering among addresses.
But because the output of the algorithm is a single source address,
an implementation need not actually sort the set; it need only
identify the "maximum" value that ends up at the front of the sorted
list.

The ordering of the addresses in the candidate set is defined by a
list of eight pair-wise comparison rules, with each rule placing a
"greater than," "less than" or "equal to" ordering on two source
addresses with respect to each other (and that rule). In the case
that a given rule produces a tie, i.e., provides an "equal to" result
for the two addresses, the remaining rules are applied (in order) to
just those addresses that are tied to break the tie. Note that if a
rule produces a single clear "winner" (or set of "winners" in the
case of ties), those addresses not in the winning set can be
discarded from further consideration, with subsequent rules applied
only to the remaining addresses. If the eight rules fail to choose a
single address, some unspecified tie-breaker should be used.

When comparing two addresses SA and SB from the candidate set, we say
"prefer SA" to mean that SA is "greater than" SB, and similarly we
say "prefer SB" to mean that SA is "less than" SB.

Rule 1: Prefer same address.
If SA = D, then prefer SA. Similarly, if SB = D, then prefer SB.

Rule 2: Prefer appropriate scope.
If Scope(SA) < Scope(SB): If Scope(SA) < Scope(D), then prefer SB
and otherwise prefer SA. Similarly, if Scope(SB) < Scope(SA): If
Scope(SB) < Scope(D), then prefer SA and otherwise prefer SB.

Rule 3: Avoid deprecated addresses.
The addresses SA and SB have the same scope. If one of the two
source addresses is "preferred" and one of them is "deprecated" (in
the RFC 2462 sense), then prefer the one that is "preferred."

Rule 4: Prefer home addresses.
If SA is simultaneously a home address and care-of address and SB is
not, then prefer SA. Similarly, if SB is simultaneously a home
address and care-of address and SA is not, then prefer SB.
If SA is just a home address and SB is just a care-of address, then
prefer SA. Similarly, if SB is just a home address and SA is just a
care-of address, then prefer SB.

Implementations should provide a mechanism allowing an application to
reverse the sense of this preference and prefer care-of addresses
over home addresses (e.g., via appropriate API extensions). Use of
the mechanism should only affect the selection rules for the invoking
application.

Rule 5: Prefer outgoing interface.
If SA is assigned to the interface that will be used to send to D
and SB is assigned to a different interface, then prefer SA.
Similarly, if SB is assigned to the interface that will be used to
send to D and SA is assigned to a different interface, then prefer
SB.

Rule 6: Prefer matching label.
If Label(SA) = Label(D) and Label(SB) <> Label(D), then prefer SA.
Similarly, if Label(SB) = Label(D) and Label(SA) <> Label(D), then
prefer SB.

Rule 7: Prefer public addresses.
If SA is a public address and SB is a temporary address, then prefer
SA. Similarly, if SB is a public address and SA is a temporary
address, then prefer SB.

Implementations MUST provide a mechanism allowing an application to
reverse the sense of this preference and prefer temporary addresses
over public addresses (e.g., via appropriate API extensions). Use of
the mechanism should only affect the selection rules for the invoking
application. This rule avoids applications potentially failing due to
the relatively short lifetime of temporary addresses or due to the
possibility of the reverse lookup of a temporary address either
failing or returning a randomized name. Implementations for which
privacy considerations outweigh these application compatibility
concerns MAY reverse the sense of this rule and by default prefer
temporary addresses over public addresses.

Rule 8: Use longest matching prefix.
If CommonPrefixLen(SA, D) > CommonPrefixLen(SB, D), then prefer SA.
Similarly, if CommonPrefixLen(SB, D) > CommonPrefixLen(SA, D), then
prefer SB.

Rule 8 may be superseded if the implementation has other means of
choosing among source addresses. For example, if the implementation
somehow knows which source address will result in the "best"
communications performance.

Rule 2 (prefer appropriate scope) MUST be implemented and given high
priority because it can affect interoperability.

6. Destination Address Selection

The destination address selection algorithm takes a list of
destination addresses and sorts the addresses to produce a new list.
It is specified here in terms of the pair-wise comparison of
addresses DA and DB, where DA appears before DB in the original list.

The algorithm sorts together both IPv6 and IPv4 addresses. To find
the attributes of an IPv4 address in the policy table, the IPv4
address should be represented as an IPv4-mapped address.

We write Source(D) to indicate the selected source address for a
destination D. For IPv6 addresses, the previous section specifies
the source address selection algorithm. Source address selection for
IPv4 addresses is not specified in this document.

We say that Source(D) is undefined if there is no source address
available for destination D. For IPv6 addresses, this is only the
case if CandidateSource(D) is the empty set.

The pair-wise comparison of destination addresses consists of ten
rules, which should be applied in order. If a rule determines a
result, then the remaining rules are not relevant and should be
ignored. Subsequent rules act as tie-breakers for earlier rules.
See the previous section for a lengthier description of how pair-wise
comparison tie-breaker rules can be used to sort a list.

Rule 1: Avoid unusable destinations.
If DB is known to be unreachable or if Source(DB) is undefined, then
prefer DA. Similarly, if DA is known to be unreachable or if
Source(DA) is undefined, then prefer DB.

   Discussion: An implementation may know that a particular
   destination is unreachable in several ways. For example, the
   destination may be reached through a network interface that is

currently unplugged. For example, the implementation may retain
for some period of time information from Neighbor Unreachability
Detection [14]. In any case, the determination of unreachability
for the purposes of this rule is implementation-dependent.

Rule 2: Prefer matching scope.
If Scope(DA) = Scope(Source(DA)) and Scope(DB) <> Scope(Source(DB)),
then prefer DA. Similarly, if Scope(DA) <> Scope(Source(DA)) and
Scope(DB) = Scope(Source(DB)), then prefer DB.

Rule 3: Avoid deprecated addresses.
If Source(DA) is deprecated and Source(DB) is not, then prefer DB.
Similarly, if Source(DA) is not deprecated and Source(DB) is
deprecated, then prefer DA.

Rule 4: Prefer home addresses.
If Source(DA) is simultaneously a home address and care-of address
and Source(DB) is not, then prefer DA. Similarly, if Source(DB) is
simultaneously a home address and care-of address and Source(DA) is
not, then prefer DB.

If Source(DA) is just a home address and Source(DB) is just a care-of
address, then prefer DA. Similarly, if Source(DA) is just a care-of
address and Source(DB) is just a home address, then prefer DB.

Rule 5: Prefer matching label.
If Label(Source(DA)) = Label(DA) and Label(Source(DB)) <> Label(DB),
then prefer DA. Similarly, if Label(Source(DA)) <> Label(DA) and
Label(Source(DB)) = Label(DB), then prefer DB.

Rule 6: Prefer higher precedence.
If Precedence(DA) > Precedence(DB), then prefer DA. Similarly, if
Precedence(DA) < Precedence(DB), then prefer DB.

Rule 7: Prefer native transport.
If DA is reached via an encapsulating transition mechanism (e.g.,
IPv6 in IPv4) and DB is not, then prefer DB. Similarly, if DB
is reached via encapsulation and DA is not, then prefer DA.

   Discussion: 6-over-4 [15], ISATAP [16], and configured tunnels
   [17] are examples of encapsulating transition mechanisms for which
   the destination address does not have a specific prefix and hence
   can not be assigned a lower precedence in the policy table. An
   implementation MAY generalize this rule by using a concept of
   interface preference, and giving virtual interfaces (like the
   IPv6-in-IPv4 encapsulating interfaces) a lower preference than
   native interfaces (like ethernet interfaces).

Rule 8: Prefer smaller scope.
If Scope(DA) < Scope(DB), then prefer DA. Similarly, if Scope(DA) >
Scope(DB), then prefer DB.

Rule 9: Use longest matching prefix.
When DA and DB belong to the same address family (both are IPv6 or
both are IPv4): If CommonPrefixLen(DA, Source(DA)) >
CommonPrefixLen(DB, Source(DB)), then prefer DA. Similarly, if
CommonPrefixLen(DA, Source(DA)) < CommonPrefixLen(DB, Source(DB)),
then prefer DB.

Rule 10: Otherwise, leave the order unchanged.
If DA preceded DB in the original list, prefer DA. Otherwise prefer
DB.

Rules 9 and 10 may be superseded if the implementation has other
means of sorting destination addresses. For example, if the
implementation somehow knows which destination addresses will result
in the "best" communications performance.

7. Interactions with Routing

This specification of source address selection assumes that routing
(more precisely, selecting an outgoing interface on a node with
multiple interfaces) is done before source address selection.
However, implementations may use source address considerations as a
tiebreaker when choosing among otherwise equivalent routes.

For example, suppose a node has interfaces on two different links,
with both links having a working default router. Both of the
interfaces have preferred (in the RFC 2462 sense) global addresses.
When sending to a global destination address, if there's no routing
reason to prefer one interface over the other, then an implementation
may preferentially choose the outgoing interface that will allow it
to use the source address that shares a longer common prefix with the
destination.

Implementations may also use the choice of router to influence the
choice of source address. For example, suppose a host is on a link
with two routers. One router is advertising a global prefix A and
the other router is advertising global prefix B. Then when sending
via the first router, the host may prefer source addresses with
prefix A and when sending via the second router, prefer source
addresses with prefix B.

8. Implementation Considerations

   The destination address selection algorithm needs information about
   potential source addresses. One possible implementation strategy is
   for getaddrinfo() to call down to the network layer with a list of
   destination addresses, sort the list in the network layer with full
   current knowledge of available source addresses, and return the
   sorted list to getaddrinfo(). This is simple and gives the best
   results but it introduces the overhead of another system call. One
   way to reduce this overhead is to cache the sorted address list in
   the resolver, so that subsequent calls for the same name do not need
   to resort the list.

   Another implementation strategy is to call down to the network layer
   to retrieve source address information and then sort the list of
   addresses directly in the context of getaddrinfo(). To reduce
   overhead in this approach, the source address information can be
   cached, amortizing the overhead of retrieving it across multiple
   calls to getaddrinfo(). In this approach, the implementation may not
   have knowledge of the outgoing interface for each destination, so it
   MAY use a looser definition of the candidate set during destination
   address ordering.

   In any case, if the implementation uses cached and possibly stale
   information in its implementation of destination address selection,
   or if the ordering of a cached list of destination addresses is
   possibly stale, then it should ensure that the destination address
   ordering returned to the application is no more than one second out
   of date. For example, an implementation might make a system call to
   check if any routing table entries or source address assignments that
   might affect these algorithms have changed. Another strategy is to
   use an invalidation counter that is incremented whenever any
   underlying state is changed. By caching the current invalidation
   counter value with derived state and then later comparing against the
   current value, the implementation could detect if the derived state
   is potentially stale.

9. Security Considerations

   This document has no direct impact on Internet infrastructure
   security.

   Note that most source address selection algorithms, including the one
   specified in this document, expose a potential privacy concern. An
   unfriendly node can infer correlations among a target node's
   addresses by probing the target node with request packets that force
   the target host to choose its source address for the reply packets.
   (Perhaps because the request packets are sent to an anycast or

multicast address, or perhaps the upper-layer protocol chosen for the
attack does not specify a particular source address for its reply
packets.) By using different addresses for itself, the unfriendly
node can cause the target node to expose the target's own addresses.

10. Examples

This section contains a number of examples, first of default behavior
and then demonstrating the utility of policy table configuration.
These examples are provided for illustrative purposes; they should
not be construed as normative.

10.1. Default Source Address Selection

The source address selection rules, in conjunction with the default
policy table, produce the following behavior:

Destination: 2001::1
Candidate Source Addresses: 3ffe::1 or fe80::1
Result: 3ffe::1 (prefer appropriate scope)

Destination: 2001::1
Candidate Source Addresses: fe80::1 or fec0::1
Result: fec0::1 (prefer appropriate scope)

Destination: fec0::1
Candidate Source Addresses: fe80::1 or 2001::1
Result: 2001::1 (prefer appropriate scope)

Destination: ff05::1
Candidate Source Addresses: fe80::1 or fec0::1 or 2001::1
Result: fec0::1 (prefer appropriate scope)

Destination: 2001::1
Candidate Source Addresses: 2001::1 (deprecated) or 2002::1
Result: 2001::1 (prefer same address)

Destination: fec0::1
Candidate Source Addresses: fec0::2 (deprecated) or 2001::1
Result: fec0::2 (prefer appropriate scope)

Destination: 2001::1
Candidate Source Addresses: 2001::2 or 3ffe::2
Result: 2001::2 (longest-matching-prefix)

    Destination: 2001::1
    Candidate Source Addresses: 2001::2 (care-of address) or 3ffe::2
    (home address)
    Result: 3ffe::2 (prefer home address)

    Destination: 2002:836b:2179::1
    Candidate Source Addresses: 2002:836b:2179::d5e3:7953:13eb:22e8
    (temporary) or 2001::2
    Result: 2002:836b:2179::d5e3:7953:13eb:22e8 (prefer matching label)

    Destination: 2001::d5e3:0:0:1
    Candidate Source Addresses: 2001::2 or 2001::d5e3:7953:13eb:22e8
    (temporary)
    Result: 2001::2 (prefer public address)

10.2. Default Destination Address Selection

    The destination address selection rules, in conjunction with the
    default policy table and the source address selection rules, produce
    the following behavior:

    Candidate Source Addresses: 2001::2 or fe80::1 or 169.254.13.78
    Destination Address List: 2001::1 or 131.107.65.121
    Result: 2001::1 (src 2001::2) then 131.107.65.121 (src
    169.254.13.78) (prefer matching scope)

    Candidate Source Addresses: fe80::1 or 131.107.65.117
    Destination Address List: 2001::1 or 131.107.65.121
    Result: 131.107.65.121 (src 131.107.65.117) then 2001::1 (src
    fe80::1) (prefer matching scope)

    Candidate Source Addresses: 2001::2 or fe80::1 or 10.1.2.4
    Destination Address List: 2001::1 or 10.1.2.3
    Result: 2001::1 (src 2001::2) then 10.1.2.3 (src 10.1.2.4) (prefer
    higher precedence)

    Candidate Source Addresses: 2001::2 or fec0::2 or fe80::2
    Destination Address List: 2001::1 or fec0::1 or fe80::1
    Result: fe80::1 (src fe80::2) then fec0::1 (src fec0::2) then
    2001::1 (src 2001::2) (prefer smaller scope)

    Candidate Source Addresses: 2001::2 (care-of address) or 3ffe::1
    (home address) or fec0::2 (care-of address) or fe80::2 (care-of
    address)
    Destination Address List: 2001::1 or fec0::1
    Result: 2001:1 (src 3ffe::1) then fec0::1 (src fec0::2) (prefer home
    address)

Candidate Source Addresses: 2001::2 or fec0::2 (deprecated) or
fe80::2
Destination Address List: 2001::1 or fec0::1
Result: 2001::1 (src 2001::2) then fec0::1 (src fec0::2) (avoid
deprecated addresses)

Candidate Source Addresses: 2001::2 or 3f44::2 or fe80::2
Destination Address List: 2001::1 or 3ffe::1
Result: 2001::1 (src 2001::2) then 3ffe::1 (src 3f44::2) (longest
matching prefix)

Candidate Source Addresses: 2002:836b:4179::2 or fe80::2
Destination Address List: 2002:836b:4179::1 or 2001::1
Result: 2002:836b:4179::1 (src 2002:836b:4179::2) then 2001::1 (src
2002:836b:4179::2) (prefer matching label)

Candidate Source Addresses: 2002:836b:4179::2 or 2001::2 or fe80::2
Destination Address List: 2002:836b:4179::1 or 2001::1
Result: 2001::1 (src 2001::2) then 2002:836b:4179::1 (src
2002:836b:4179::2) (prefer higher precedence)

10.3. Configuring Preference for IPv6 or IPv4

The default policy table gives IPv6 addresses higher precedence than
IPv4 addresses. This means that applications will use IPv6 in
preference to IPv4 when the two are equally suitable. An
administrator can change the policy table to prefer IPv4 addresses by
giving the ::ffff:0.0.0.0/96 prefix a higher precedence:

Prefix	Precedence	Label
::1/128	50	0
::/0	40	1
2002::/16	30	2
::/96	20	3
::ffff:0:0/96	100	4

This change to the default policy table produces the following
behavior:

Candidate Source Addresses: 2001::2 or fe80::1 or 169.254.13.78
Destination Address List: 2001::1 or 131.107.65.121
Unchanged Result: 2001::1 (src 2001::2) then 131.107.65.121 (src
169.254.13.78) (prefer matching scope)

Candidate Source Addresses: fe80::1 or 131.107.65.117
Destination Address List: 2001::1 or 131.107.65.121
Unchanged Result: 131.107.65.121 (src 131.107.65.117) then 2001::1
(src fe80::1) (prefer matching scope)

Candidate Source Addresses: 2001::2 or fe80::1 or 10.1.2.4
Destination Address List: 2001::1 or 10.1.2.3
New Result: 10.1.2.3 (src 10.1.2.4) then 2001::1 (src 2001::2)
(prefer higher precedence)

10.4. Configuring Preference for Scoped Addresses

The destination address selection rules give preference to
destinations of smaller scope. For example, a site-local destination
will be sorted before a global scope destination when the two are
otherwise equally suitable. An administrator can change the policy
table to reverse this preference and sort global destinations before
site-local destinations, and site-local destinations before link-
local destinations:

Prefix	Precedence	Label
::1/128	50	0
::/0	40	1
fec0::/10	37	1
fe80::/10	33	1
2002::/16	30	2
::/96	20	3
::ffff:0:0/96	10	4

This change to the default policy table produces the following
behavior:

Candidate Source Addresses: 2001::2 or fec0::2 or fe80::2
Destination Address List: 2001::1 or fec0::1 or fe80::1
New Result: 2001::1 (src 2001::2) then fec0::1 (src fec0::2) then
fe80::1 (src fe80::2) (prefer higher precedence)

Candidate Source Addresses: 2001::2 (deprecated) or fec0::2 or
fe80::2
Destination Address List: 2001::1 or fec0::1
Unchanged Result: fec0::1 (src fec0::2) then 2001::1 (src 2001::2)
(avoid deprecated addresses)

10.5. Configuring a Multi-Homed Site

Consider a site A that has a business-critical relationship with
another site B. To support their business needs, the two sites have
contracted for service with a special high-performance ISP. This is
in addition to the normal Internet connection that both sites have
with different ISPs. The high-performance ISP is expensive and the
two sites wish to use it only for their business-critical traffic
with each other.

Each site has two global prefixes, one from the high-performance ISP
and one from their normal ISP. Site A has prefix 2001:aaaa:aaaa::/48
from the high-performance ISP and prefix 2007:0:aaaa::/48 from its
normal ISP. Site B has prefix 2001:bbbb:bbbb::/48 from the high-
performance ISP and prefix 2007:0:bbbb::/48 from its normal ISP. All
hosts in both sites register two addresses in the DNS.

The routing within both sites directs most traffic to the egress to
the normal ISP, but the routing directs traffic sent to the other
site's 2001 prefix to the egress to the high-performance ISP. To
prevent unintended use of their high-performance ISP connection, the
two sites implement ingress filtering to discard traffic entering
from the high-performance ISP that is not from the other site.

The default policy table and address selection rules produce the
following behavior:

Candidate Source Addresses: 2001:aaaa:aaaa::a or 2007:0:aaaa::a or
fe80::a
Destination Address List: 2001:bbbb:bbbb::b or 2007:0:bbbb::b
Result: 2007:0:bbbb::b (src 2007:0:aaaa::a) then 2001:bbbb:bbbb::b
(src 2001:aaaa:aaaa::a) (longest matching prefix)

In other words, when a host in site A initiates a connection to a
host in site B, the traffic does not take advantage of their
connections to the high-performance ISP. This is not their desired
behavior.

Candidate Source Addresses: 2001:aaaa:aaaa::a or 2007:0:aaaa::a or
fe80::a
Destination Address List: 2001:cccc:cccc::c or 2006:cccc:cccc::c
Result: 2001:cccc:cccc::c (src 2001:aaaa:aaaa::a) then
2006:cccc:cccc::c (src 2007:0:aaaa::a) (longest matching prefix)

In other words, when a host in site A initiates a connection to a
host in some other site C, the reverse traffic may come back through
the high-performance ISP. Again, this is not their desired behavior.

This predicament demonstrates the limitations of the longest-
matching-prefix heuristic in multi-homed situations.

However, the administrators of sites A and B can achieve their
desired behavior via policy table configuration. For example, they
can use the following policy table:

Prefix	Precedence	Label
::1	50	0
2001:aaaa:aaaa::/48	45	5
2001:bbbb:bbbb::/48	45	5
::/0	40	1
2002::/16	30	2
::/96	20	3
::ffff:0:0/96	10	4

This policy table produces the following behavior:

Candidate Source Addresses: 2001:aaaa:aaaa::a or 2007:0:aaaa::a or
fe80::a
Destination Address List: 2001:bbbb:bbbb::b or 2007:0:bbbb::b
New Result: 2001:bbbb:bbbb::b (src 2001:aaaa:aaaa::a) then
2007:0:bbbb::b (src 2007:0:aaaa::a) (prefer higher precedence)

In other words, when a host in site A initiates a connection to a
host in site B, the traffic uses the high-performance ISP as desired.

Candidate Source Addresses: 2001:aaaa:aaaa::a or 2007:0:aaaa::a or
fe80::a
Destination Address List: 2001:cccc:cccc::c or 2006:cccc:cccc::c
New Result: 2006:cccc:cccc::c (src 2007:0:aaaa::a) then
2001:cccc:cccc::c (src 2007:0:aaaa::a) (longest matching prefix)

In other words, when a host in site A initiates a connection to a
host in some other site C, the traffic uses the normal ISP as
desired.

Normative References

   [1] Hinden, R. and S. Deering, "IP Version 6 Addressing
        Architecture", RFC 2373, July 1998.

   [2] Thompson, S. and T. Narten, "IPv6 Stateless Address
        Autoconfiguration", RFC 2462 , December 1998.

   [3] Narten, T. and R. Draves, "Privacy Extensions for Stateless
        Address Autoconfiguration in IPv6", RFC 3041, January 2001.

   [4] Bradner, S., "Key words for use in RFCs to Indicate Requirement
        Levels", BCP 14, RFC 2119, March 1997.

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

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

Informative References

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

   [8] Johnson, D. and C. Perkins, "Mobility Support in IPv6", Work in
        Progress.

   [9] S. Cheshire, B. Aboba, "Dynamic Configuration of IPv4 Link-local
        Addresses", Work in Progress.

   [10] Gilligan, R., Thomson, S., Bound, J. and W. Stevens, "Basic
        Socket Interface Extensions for IPv6", RFC 2553, March 1999.

   [11] S. Deering et. al, "IP Version 6 Scoped Address Architecture",
        Work in Progress.

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

   [13] Baker, F, "Requirements for IP Version 4 Routers", RFC 1812,
        June 1995.

   [14] Narten, T. and E. Nordmark, and W. Simpson, "Neighbor Discovery
        for IP Version 6", RFC 2461, December 1998.

   [15] Carpenter, B. and C. Jung, "Transmission of IPv6 over IPv4
        Domains without Explicit Tunnels", RFC 2529, March 1999.

   [16] F. Templin et. al, "Intra-Site Automatic Tunnel Addressing
        Protocol (ISATAP)", Work in Progress.

   [17] Gilligan, R. and E. Nordmark, "Transition Mechanisms for IPv6
        Hosts and Routers", RFC 1933, April 1996.

Author's Address

   Richard Draves
   Microsoft Research
   One Microsoft Way
   Redmond, WA 98052

   Phone: +1 425 706 2268
   EMail: richdr@microsoft.com

Full Copyright Statement

Acknowledgement