Network Working Group   F.J. Baker
Internet-Draft   Cisco Systems
Intended status: Standards Track   February 17, 2013
Expires: August 21, 2013

IPv6 Source/Destination Routing using OSPFv3
draft-baker-ipv6-ospf-dst-src-routing-00

Abstract

This note describes the changes necessary for OSPFv3 to route classes of IPv6 traffic that are defined by a source prefix and a destination prefix. This implies not routing "to a destination", but "traffic matching a classification tuple". The obvious application is egress routing - routing traffic using a given prefix to an upstream network that will not drop traffic using that prefix using BCP 38 filters.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 2
1.1. Requirements Language 2
2. Theory of Routing 3
2.1. Dealing with ambiguity 3
3. Extensions necessary for IPv6 Source/Destination Routing in OSPFv3 4
3.1. IPv6 Source Prefix TLV 4
4. IANA Considerations 4
5. Security Considerations 5
6. Privacy Considerations 5
7. Acknowledgements 5
8. Change Log 5
9. References 5
9.1. Normative References 5
9.2. Informative References 5
Appendix A. Use case: Egress Routing 6
Appendix B. FIB Design 6
B.1. Linux Source-Address Forwarding 7
B.1.1. One FIB per source prefix 7
B.1.2. One FIB per source prefix plus a general FIB 7
B.2. PATRICIA 8
B.2.1. Virtual Bit String 8
B.2.2. Tree Construction 9
B.2.3. Tree Lookup 10
Author's Address 10

1. Introduction

This specification builds on the extensible LSAs defined in [I-D.baker-ipv6-ospf-extensible.txt]. It adds the option for an IPv6 Source Prefix, to define routes defined by a source and a destination prefix.

1.1. Requirements Language

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 [RFC2119].

2. Theory of Routing

Both IS-IS and OSPF perform their calculations by building a lattice of routers and routes from the router performing the calculation to each router, and then use those routes to get to destinations that those routes advertise connectivity to. Following the SPF algorithm, calculation starts by selecting a starting point (typically the router doing the calculation), and successively adding {link, router) pairs until one has calculated a route to every router in the network. As each router is added, including the original router, destinations that it is directly connected to are turned into routes in the route table: "to get to 2001:db8::/32, route traffic to {interface, list of next hop routers}". For immediate neighbors to the originating router, of course, there is no next hop router; traffic is handled locally.

2.1. Dealing with ambiguity

In any routing protocol, there is the possibility of ambiguity. An area border router might, for example, summarize the routes to other areas into a small set of relatively short prefixes, which have more specific routes within the area. Traditionally, we have dealt with that using a "longest match first" rule. If the same datagram matches more than one destination prefix advertised within the area, we follow the route to the longest matching prefix.

When routing a class of traffic, we follow an analogous "most specific match" rule; we follow the route for the most specific matching tuple. In cases of simple overlap, such as routing to 2001:db8::/32 or 2001:db8:1::/48, that is exactly analogous; we choose one of the two routes.

It is possible, however, to construct an ambiguous case in which neither class subsumes the other. For example, presume that

o A is a prefix,

o B is a more-specific prefix within A,

o C is a different prefix, and

o D is a more-specific prefix of C.

The two classes {A, D, *, *} and {B, C, *, *} are ambiguous: a datagram within {B, D, *, *} matches both classes, and it is not clear in the data plane what decision to make. Solving this requires the addition of a third route in the FIB corresponding to the class {B, D, *, *}, which is more-specific than either of the first two, and can be given routing guidance based on metrics or other policy in the usual way.

3. Extensions necessary for IPv6 Source/Destination Routing in OSPFv3

The several extensible LSAs defined in [I-D.baker-ipv6-ospf-extensible.txt] require one additional option to accomplish source/destination routing: the source prefix. This is defined here.

In addition, should (as one might expect is normal) destination-only intra-area-prefix, inter-area-prefix, and AS-external-prefix LSAs be encountered, we need a rule for interpretation. The rule is that they are treated exactly as the extensible version if the source prefix option is not specified or is specified to be ::/0 (any IPv6 address).

3.1. IPv6 Source Prefix TLV

The IPv6 Source Prefix TLV MAY be used with the IPv6 Destination Prefix TLV, but MUST NOT be used with the IPv4 Source Prefix TLV or the IPv4 Destination Prefix TLV.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Type     |     Length     |Prefix Length  |    Prefix
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Source Prefix TLV

Source Prefix Type: assigned by IANA

TLV Length: Length of the TLV in octets

Prefix Length: Length of the prefix in bits, in the range 0..128

Prefix: (source prefix length +7)/8 octets of prefix

4. IANA Considerations

This section will request an identifying value for the TLV defined. This is deferred to the -01 version of the draft.

5. Security Considerations

To be considered.

6. Privacy Considerations

To be considered.

7. Acknowledgements

8. Change Log

Initial Version: February 2013

9. References

9.1. Normative References

[ISO.10589.1992]
International Organization for Standardization, "Intermediate system to intermediate system intra-domain-routing routine information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode Network Service (ISO 8473)", ISO Standard 10589, 1992.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Informative References

[I-D.baker-ipv6-ospf-extensible.txt]
Baker, F., "Extensible OSPF LSAs", February 2013.

[PATRICIA]
Morrison, D.R., "Practical Algorithm to Retrieve Information Coded in Alphanumeric", Journal of the ACM 15(4) pp514-534, October 1968.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.

Appendix A. Use case: Egress Routing

Using this technology for egress routing is straightforward. Presume a multihomed edge (residential or enterprise) network with multiple egress points to the various ISPs. These ISPs allocate PA prefixes to the network. Due to BCP 38 [RFC2827], the network must presume that its upstream ISPs will filter out any traffic presented to them that does not use their PA prefix.

Within the network, presume that a /64 prefix from each of those PA prefixes is allocated on each LAN, and that hosts generate and use multiple addresses on each interface.

Within the network, we permit any host to communicate with any other. Hence, routing advertisements within the network use traditional destination routing, which is understood to be advertising the traffic class

    {destination, ::/0}.

From the egresses, the firewall or its neighboring router injects a default route for traffic "from" its PA prefix:

    {::/0, PA prefix}.

Routing is calculated as normal, with the exception that traffic following a default route will select that route based on the source address. Traffic will never be lost to BCP 38 filters, because by definition the only traffic sent to the ISP is using the PA prefix assigned by the ISP. In addition, while hosts can use spoofed addresses outside of their PA prefixes to attack each other, they cannot send traffic using spoofed addresses to their upstream networks; such traffic has no route.

Appendix B. FIB Design

While the design of the Forwarding Information Base is not a matter for standardization, as it only has to work correctly, not interoperate with something else, the design of a FIB for this type of lookup may differ from approaches used in destination routing. We describe one possible approach that is known to work, from the perspective of a proof of concept.

B.1. Linux Source-Address Forwarding

The University of Waikato has added to the Linux Advanced Routing & Traffic Control facility the ability to maintain multiple FIBS, one for each of a set of prefixes. Implementing source/destination routing using this mechanism is not difficult.

The router must know what source prefixes might be used in its domain. This may be by configuration or, at least in concept, learned from the routing protocols themselves. In whichever way that is done, one can imagine two fundamental FIB structures to serve N source prefixes; N FIBs, one per prefix, or N+1 FIBs, one per prefix plus one for destinations for which the source prefix is unspecified.

B.1.1. One FIB per source prefix

In an implementation with one FIB per source prefix, the routing algorithm has two possibilities.

- If it calculates a route to a prefix (such as a default route) associated with a given source prefix, it stores the route in the FIB for the relevant source prefix.

- If it calculates a route for which the source prefix is unspecified, it stores that route in all N FIBs.

When forwarding a datagram, the IP forwarder looks at the source address of the datagram to determine which FIB it should use. If it is from an address for which there is no FIB, the forwarder discards the datagram as containing a forged source address. If it is from an address within one of the relevant prefixes, it looks up the destination in the indicated FIB and forwards it in the usual way.

The argument for this approach is simplicity: there is one place to look in making a forwarding decision for any given datagram. The argument against it is memory space; it is likely that the FIBs will be similar, but every destination route not associated with a source prefix is duplicated in each FIB. In addition, since it automatically removes traffic whose source address is not among the configured list, it limits the possibility of user software using improper addresses.

B.1.2. One FIB per source prefix plus a general FIB

In an implementation with N+1 FIBs, the algorithm is slightly more complex.

- o If it calculates a route to a prefix (such as a default route) associated with a given source prefix, it stores the route in the FIB for the relevant source prefix.

- o If it calculates a route for which the source prefix is unspecified, it stores that route in the FIB that is not associated with a source prefix.

When forwarding a datagram, the IP forwarder looks at the source address of the datagram to determine which FIB it should use. If it is from one of the configured prefixes, it looks the destination up in the indicated FIB. In any event it also looks the destination up in the "unspecified source address" FIB. If the destination is found in only one of the two, the indicated route is followed. If the destination is found in both, the more specific route is followed.

The argument for this approach is memory space; if a large percentage of routes are only in the general FIB, such as when egress routing is used for the default route and all other routes are internal, the other FIBs are likely to be very small - perhaps only a single default route. The argument against this approach is complexity: most lookups if not all will be done in a prefix-specific FIB and in the general FIB.

B.2. PATRICIA

One approach is a [PATRICIA] Tree. This is a relative of a Trie, but unlike a Trie, need not use every bit in classification, and does not need the bits used to be contiguous. It depends on treating the bit string as a set of slices of some size, potentially of different sizes. Slice width is an implementation detail; since the algorithm is most easily described using a slice of a single bit, that will be presumed in this description.

B.2.1. Virtual Bit String

It is quite possible to view the fields in a datagram header incorporated into the classification tuple as a virtual bit string such as is shown in Figure 1. This bit string has various regions within it. Some vary and are therefore useful in a radix tree lookup. Some may be essentially constant - all global IPv6 addresses at this writing are within 2000::/3, for example, so while it must be tested to assure a match, incorporating it into the radix tree may not be very helpful in classification. Others are ignored; if the destination is a remote /64, we really don't care what the EID is. In addition, due to variation in prefix length and other details, the widths of those fields vary among themselves. The algorithm the FIB implements, therefore, must efficiently deal with the fact of a discontiguous lookup key.

```
+---------------------+---------------------+-----+-----------+
|Destination Prefix   |Source Prefix        |DSCP | Flow Label|
+------+------+-------+------+-------+-------+-----+-----------+
 Common|Varying|Ignored|Common|Varying|Ignored|Varying or ignored
```

Figure 1: Treating a traffic class as a virtual bit string

B.2.2. Tree Construction

The tree is constructed by recursive slice-wise decomposition. At each stage, the input is a set of classes to be classified. At each stage, the result is the addition of a lookup node in the tree that identifies the location of its slice in the virtual bit string (which might be a bit number), the width of the slice to be inspected, and an enumerated set of results. Each result is a similar set of classes, and is analyzed in a similar manner.

The analysis is performed by enumerating which bits that have not already been considered are best suited to classification. For a slice of N bits, one wants to select a slide that most evenly divides the set of classes into 2^N subsets. If one or more bits in the slice is ignored in some of the classes, those classes must be included in every subset, as the actual classification of them will depend on other bits.

```
Input:{2001:db8::/32, ::/0, *, *}
      {2001:db8:1::/48, ::/0, AF41, *}
      {2001:db8:1::/48, ::/0, AF42, *}
      {2001:db8:1::/48, ::/0, AF43, *}
Common parts: Destination prefix 2001:dba, source prefix, and label
Varying parts: DSCP and the third set of sixteen bits in the
               destination prefix
One possible decomposition:
(1) slice = DSCP
```

```
       enumerated cases:
   (a) { {2001:db8::/32, ::/0, *, *}, {2001:db8:1::/48, ::/0, AF41, *} }
   (b) { {2001:db8::/32, ::/0, *, *}, {2001:db8:1::/48, ::/0, AF42, *} }
   (c) { {2001:db8::/32, ::/0, *, *}, {2001:db8:1::/48, ::/0, AF43, *} }
   (2) slice = third sixteen bit field in destination
       This divides each enumerated case into those containing 0001 and
       "everything else", which would imply 2001:db8::/32
                             (1) DSCP
                  --------------------------
                 (1a)       (1b)         (1c)
                /    \     /    \       /    \
             /32   /48  /32   /48    /32   /48
```

Figure 2: Example PATRICIA Tree

B.2.3. **Tree Lookup**

To look something up in a PATRICIA Tree, one starts at the root of the tree and performs the indicated comparisons recursively walking down the tree until one reaches a terminal node. When the enumerated subset is empty or contains only a single class, classification stops. Either classification has failed (there was no matching class, or one has presumably found the indicated class. At that point, every bit in the virtual bit string must be compared to the classifier; classification is accepted on a perfect match.

In the example in Figure 2, if a packet {2001:db8:1:2:3:4:5:6, 2001:db8:2:3:4:5:6:7, AF41, 0} arrives, we start at the root. Since it is an AF41 packet, we deduce that case (1a) applies, and since the destination has 0001 in the third sixteen bit field of the destination address, we are comparing to {2001:db8:1::/48, ::/0, AF41, *}. Since the destination address is within 2001:db8:1::/48, classification as that succeeds.

Author's Address

Fred Baker
Cisco Systems
Santa Barbara, California 93117
USA

Email: fred@cisco.com