Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 21, 2013

F.J. Baker
Cisco Systems

February 17, 2013

Using OSPFv3 with Role-Based Access Control
draft-baker-ipv6-ospf-dst-flowlabel-routing-00

Abstract

This note describes the changes necessary for OSPFv3 to route classes of IPv6 traffic that are defined by an IPv6 Flow Label and a destination prefix. This implies not routing "to a destination", but "traffic matching a classification tuple". The obvious application is data center inter-tenant routing using a form of role-based access control. If the sender doesn't know the value to insert in the flow label (the receiver's tenant ID), he in effect has no route to that destination.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 21, 2013.

Table of Contents

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 Flow Label, to define routes defined by a destination prefix plus a flow label.

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

- A is a prefix,
- B is a more-specific prefix within A,
- C is a specific flow label value

The two classes "routes to A using flow label C" and "routes to B using any flow label" are ambiguous: a datagram to B using the flow label C 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 for routes to B using flow label C, 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 OSPFv3

The several extensible LSAs defined in [I-D.baker-ipv6-ospf-extensible.txt] require one additional option to accomplish source/destination routing: the flow label in use by the destination. 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 flow label TLV is omitted, which is to say, that any flow label value is accepted.

3.1. On Flow Labels and security

According to section 6 of [RFC2460], a Flow Label is a 20 bit number which

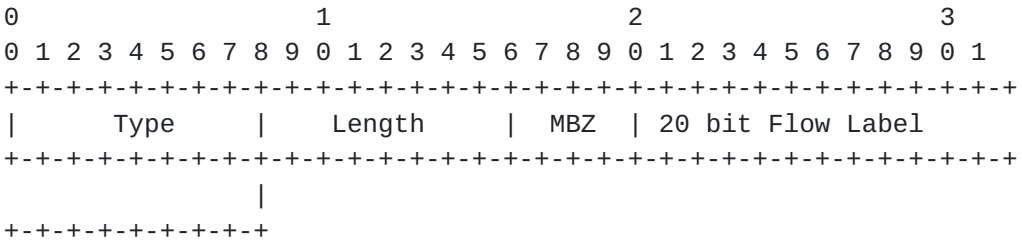
> "may be used by a source to label sequences of packets for which it requests special handling by the IPv6 routers".

The possible use case mentioned in an appendix is egress routing. Other RFCs suggest other possible use cases.

In this model, the flow label is used to prove that the datagram's sender has specific knowledge of its intended receiver. No proof is requested; this is left for higher layer exchanges such as IPSec or TLS. However, if the information is distributed privately, such as through DHCP/DHCPv6, the network can presume that a system that marks traffic with the right flow label has a good chance of being authorized to communicate with its peer.

The key consideration, in this context, is that the flow label is a 20 bit number. As such, an advertised route requiring a given flow label value is calling for an exact match of all 20 bits of the label value.

3.2. Flow Label TLV



Flow Label TLV

Flow Label Type: assigned by IANA

TLV Length: Length of the TLV in octets

Flow Label: 20 bits of Flow Label value

MBZ: unused, MUST be zero when generated, ignored on receipt.

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

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

[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.

[RFC2460] Deering, S.E. and R.M. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

9.2. Informative References

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

Appendix A. Use case: Data Center Role-based Access Control

Consider a data center in which IPv6 is deployed throughout using internet routing technologies instead of tunnels, and the Flow Label is used to identify tenants, as discussed in Section 3.1. Hosts are required, by configuration if necessary, to know their own tenant number and the numbers of any tenants they are authorized to communicate with. When they originate a datagram, they send it to their peer's destination address and label it with their peer's tenant id. They, or their router on their behalf, advertise their own addresses as traffic classes

{destination prefix, Tenant Flow Label }

The net effect is that traffic is routed among tenants that are authorized to communicate, but not among tenants that are not authorized to communicate - there is no route. This is done without tunnels, access lists, or other data plane overhead; the overhead is in the control plane, equipping authorized parties to communicate.

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 two possible approaches from the perspective of a proof of concept. These are a staged lookup and a single FIB.

B.1. Staged Lookup

A FIB can be designed as a staged lookup. Given that it is unlikely that any given destination would support very many tenants, a simple list or small hash may be sufficient; one looks up the destination, and having found it, validates the flow label used. In such a design, it is necessary to have the option of "any" flow label in addition to the set of specified flow labels, as it is legal and correct to advertise routes that do not have flow labels.

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.

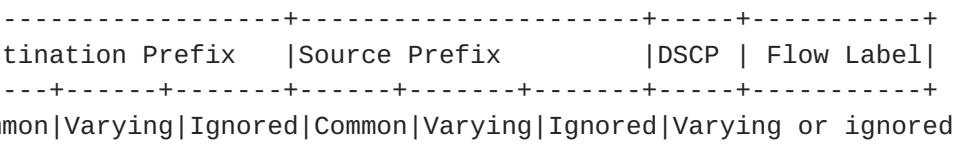


Figure 1: Treating a traffic class as a virtual bit string

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
```

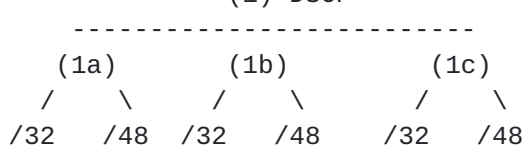


Figure 2: Example PATRICIA Tree

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