Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 23 October 2021

M. Boutier
J. Chroboczek
IRIF, University of Paris
21 April 2021

Source-Specific Routing in Babel
draft-ietf-babel-source-specific-08

Abstract

Source-specific routing (also known as Source-Address Dependent Routing, SADR) is an extension to traditional next-hop routing where packets are forwarded according to both their destination and their source address. This document describes an extension for source-specific routing to the Babel routing protocol.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction and background

The Babel routing protocol [RFC8966] is a distance vector routing protocol for next-hop routing. In next-hop routing, each node maintains a forwarding table which maps destination prefixes to next hops. The forwarding decision is a per-packet operation which depends on the destination address of the packets and on the entries of the forwarding table. When a packet is about to be routed, its destination address is compared to the prefixes of the routing table: the entry with the most specific prefix containing the destination address of the packet is chosen, and the packet is forwarded to the associated next-hop. Next-hop routing is a simple, well understood paradigm that works satisfactorily in a large number of cases.

The use of next-hop routing limits the flexibility of the routing system in two ways. First, since the routing decision is local to each router, a router A can only select a route ABC...Z if its neighbouring router B has selected the route BC...Z. Second, the

only criterion used by a router to choose a route is the destination address: two packets with the same destination follow the same route. Yet, there are other data in the IP header that could conceivably be used to guide the routing decision -- the ToS octet and, of course, the source address.

Source-specific routing [SS-ROUTING], or Source Address Dependent Routing (SADR), is a modest extension to next-hop routing where the forwarding decision depends not only on the destination address but also on the source address of the packet being routed, which makes it possible for two packets with the same destination but different source addresses to be routed following different paths.

This document describes a source-specific routing extension for the Babel routing protocol [RFC8966]. This involves minor changes to the data structures, which must include a source prefix in addition to the destination prefix already present, and some changes to the Update, Route Request and Seqno Request TLVs, which are extended with a source prefix. The source prefix is encoded using a mandatory sub-TLV ([RFC8966] Section 4.4).

1.1. Application to multihoming

Multihoming is the practice of connecting a single network to two or more transit networks. The main application of source-specific routing is a form of multihoming known as "multihoming with multiple addresses".

Classical multihoming consists in assigning a provider-independent range of addresses to the multihomed network and announcing it to all transit providers. While classical multihoming works well for large networks, the cost of obtaining a provider-independent address range and announcing it globally in the Internet is prohibitive for small networks. Unfortunately, it is not possible to implement classical multihoming with ordinary provider-dependent addresses: in a network connected to two providers A and B, a packet with a source address allocated by A needs to be routed through the edge router connected to A. If it is routed through the edge router connected to B, it will most likely be filtered (dropped), in accordance with [BCP84].

In multihoming with multiple addresses, every host in the multihomed network is assigned multiple addresses, one for each transit provider. Additional mechanisms are needed in order (i) to choose, for each packet, a source address that is associated with a provider that is currently up, and (ii) to route each packet towards the router connected to the provider associated with its source address. One might argue that multihoming with multiple addresses splits the difficult problem of multihoming into two simpler sub-problems.

The issue of choosing a suitable source address is a decision local to the sending host, and an area of active research. The simplest solution is to use a traditional transport-layer protocol, such as TCP, and to probe all available source addresses at connection time, analogously to what is already done with destination addresses, either sequentially [RFC3484] or in parallel [RFC8305]. Since the transport-layer protocol is not aware of the multiple available addresses, flows are interrupted when the selected provider goes down (from the point of view of the user, all TCP connections are dropped when the network environment changes). A better user experience can be provided by making available all of the potential source and destination addresses to higher layer protocols, either at the transport layer [RFC8684] [RFC4960], or at the applicaton layer [RFC8445].

Source-specific routing solves the problem of routing a packet to the edge router indicated by its source address. Every edge router announces into the routing domain a default route specific to the prefix associated with the provider it is connected to. This route is propagated all the way to the routers on the access link, which are therefore able to route every packet to the correct router. Hosts simply send packets to their default router -- no host changes are necessary at the network layer.

1.2. Other applications

In addition to multihoming with multiple addresses, we are aware of two applications of source-specific routing. Tunnels and VPNs are packet encapsulation techniques that are commonly used in the Internet to establish a network-layer topology that is different from the physical topology. In some deployments, the default route points at the tunnel; this causes the network stack to attempt to send encapsulated packets through the tunnel, which causes it to break. Various solutions to this problem are possible, the most common of which is to point a host route at the tunnel endpoint.

When source-specific routing is available, it becomes possible to announce through the tunnel a default route that is specific to the prefix served by the tunnel. Since the encapsulated packets have a source address that is not within that prefix, they are not routed through the tunnel.

The third application of source-specific routing is controlled anycast. Anycast is a technique in which a single destination address is used to represent multiple network endpoints, collectively called an "anycast group". A packet destined to the anycast group is routed to an arbitrary member of the group, typically the one that is nearest according to the routing protocol.

In many applications of anycast, such as DNS root servers, the nondeterminism of anycast is acceptable; some applications, however, require finer control. For example, in some Content Distribution Networks (CDNs) every endpoint is expected to handle a well-defined subset of the client population. With source-specific routing, it is possible for each member of the anycast group to announce a route specific to its client population, a technique that is both simpler and more robust than manually tweaking the routing protocol's metric ("prepending" in BGP).

1.3. Specificity of prefix pairs

In ordinary next-hop routing, when multiple routing table entries match the destination of a packet, the "longest prefix rule" mandates that the most specific one applies. The reason why this rule makes sense is that the set of prefixes has the following "tree property":

> for any prefixes P and P', either P and P' are disjoint, or one is more specific than the other.

It would be a natural proposition to order pairs of prefixes pointwise: to define that (D,S) is more specific than (D',S') when D is more specific than D and S is more specific than S'. Unfortunately, the set of pairs of prefixes with the pointwise ordering doesn't satisfy the tree property. Indeed, consider the following two pairs:

```
   (2001:db8:0:1::/64, ::/0) and (::/0, 2001:db8:0:2::/64)
```

These two pairs are not disjoint (a packet with destination 2001:db8:0:1::1 and source 2001:db8:0:2::1 is matched by both), but neither is more specific than the other. The effect is that there is no natural unambiguous way to interpret a routing table such as the following:

```
          destination                source     next-hop
    2001:db8:0:1::/64                  ::/0            A
                 ::/0     2001:db8:0:2::/64            B
```

A more refined ordering over pairs of prefixes is required in order to avoid all ambiguities. There are two natural choices: the destination-first ordering, where (D,S) is more specific than (D',S') when

* D is strictly more specific than D'; or

* D = D' and S is more specific than S',

and, symmetrically, the source-first ordering, in which sources are compared first and destinations second.

Expedient as it would be to leave the choice to the implementation, this is not possible: all routers in a routing domain must use the same ordering, lest persistent routing loops occur. Indeed, consider the following topology:

```
   A --- B --- C --- D
```

Suppose that A announces a route for (::/0, 2001:db8:0:2::/64), while D announces a route for (2001:db8:0:1::/64, ::/0). Suppose further that B uses the destination-first ordering, while C uses the source-first ordering. Then a packet that matches both routes, say, with destination 2001:db8:0:1::1 and source 2001:db8:0:2::1, would be sent by B towards D and by C towards A, and would therefore loop indefinitely between B and C.

This document mandates (Section 4) that all routers use the destination-first ordering, which is generally believed to be more useful than the source-first ordering. Consider the following topology, where A is an edge router connected to the Internet and B is an internal router connected to an access network N:

```
  (::/0, S)             (D, ::/0)
   Internet --- A --- B --- N
```

A announces a source-specific default route with source S (::/0, S), while B announces a non-specific route to prefix D. Consider what happens to a packet with a desination in D and a source in S. With the destination-first ordering, the packet is routed towards the network N, which is the only way it can possibly reach its destination. With the source-first ordering, on the other hand, the packet is sent towards the Internet, with no hope to ever reach its destination in N.

2. Specification of Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Data Structures

A number of the conceptual data structures described in Section 3.2 of [RFC8966] contain a destination prefix. This specification extends these data structures with a source prefix. Data from the original protocol, which do not specify a source prefix, are stored with a zero length source prefix, which matches exactly the same set of packets as the original, non-source-specific data.

3.1. The Source Table

Every Babel node maintains a source table, as described in [RFC8966] Section 3.2.5. A source-specific Babel node extends this table with the following field:

* The source prefix (sprefix, splen) specifying the source address of packets to which this entry applies.

The source table is now indexed by 5-tuples of the form (prefix, plen, sprefix, splen, router-id).

Note that the route entry contains a source (see sections 2 and 3.2.5 of [RFC8966]) which itself contains both destination and source prefixes. These are two different concepts, and must not be confused.

3.2. The Route Table

Every Babel node maintains a route table, as described in [RFC8966] Section 3.2.6. Each route table entry contains, among other data, a source, which this specification extends with a source prefix as described above. The route table is now indexed by 5-tuples of the form (prefix, plen, sprefix, splen, neighbour), where the first four components are obtained from the source.

3.3. The Table of Pending Seqno Requests

Every Babel node maintains a table of pending seqno requests, as described in [RFC8966], Section 3.2.7. A source-specific Babel node extends this table with the following entry:

* The source prefix (sprefix, splen) being requested.

The table of pending seqno requests is now indexed by 5-tuples of the form (prefix, plen, sprefix, splen, router-id).

4. Data Forwarding

As noted in Section 1.3 above, source-specific tables can, in general, be ambiguous, and all routers in a routing domain must use the same algorithm for choosing applicable routes. An implementation of the extension described in this document MUST choose routing table entries by using the destination-first ordering, where a routing table entry R1 is preferred to a routing table entry R2 when either R1's destination prefix is more specific than R2's, or the destination prefixes are equal and R1's source prefix is more specific than R2's.

In practice, this means that a source-specific Babel implementation must take care that any lower layer that performs packet forwarding obey this semantics. More precisely:

* if the lower layers implement the destination-first ordering, then the Babel implementation SHOULD use them directly;

* if the lower layers can hold source-specific routes, but not with the right semantics, then the Babel implementation MUST either silently ignore any source-specific routes, or disambiguate the routing table by using a suitable disambiguation algorithm (see Section V.B of [SS-ROUTING] for such an algorithm);

* if the lower layers cannot hold source-specific routes, then a Babel implementation MUST silently ignore any source-specific routes.

5. Protocol Operation

This extension does not fundamentally change the operation of the Babel protocol, and we therefore only describe differences between the original protocol and the extended protocol.

In the original protocol, three TLVs carry a destination prefix: Updates, Route Requests and Seqno Requests. This specification extends these messages so that they may carry a Source Prefix sub-TLV, as described in Section 7 below. The sub-TLV is marked as mandatory, so that an unextended implementation will silently ignore the whole enclosing TLV. A node obeying this specification MUST NOT send a TLV with a zero-length source prefix: instead, it sends a TLV with no Source Prefix sub-TLV. Conversely, an extended implementation MUST interpret an unextended TLV as carrying a source prefix of zero length. Taken together, these properties ensure interoperability between the original and extended protocols (see Section 6 below).

5.1. Protocol Messages

This extension allows three TLVs of the original Babel protocol to carry a source prefix: Update TLVs, Route Request TLVs, and Seqno Request TLVs.

In order to advertise a route with a non-zero length source prefix, a node sends a source-specific Update, i.e., an Update with a Source Prefix sub-TLV. When a node receives a source-specific Update (prefix, source prefix, router-id, seqno, metric) from a neighbour neigh, it behaves as described in [RFC8966] Section 3.5.3, except that the entry under consideration is indexed by (prefix, plen, sprefix, splen, neigh) rather than just (prefix, plen, neigh).

Similarly, when a node needs to send a Request of either kind that applies to a route with a non-zero length source prefix, it sends a source-specific Request, i.e., a Request with a Source Prefix sub-TLV. When a node receives a source-specific Request, it behaves as described in Section 3.8 of [RFC8966], except that the request applies to the Route Table entry carrying the source prefix indicated by the Source Prefix sub-TLV.

5.2. Wildcard Messages

In the original protocol, the Address Encoding (AE) value 0 is used for wildcard messages: messages that apply to all routes, of any address family and with any destination prefix. Wildcard messages are allowed in two places in the protocol: wildcard retractions are used to retract all of the routes previously advertised by a node on a given interface, and wildcard Route Requests are used to request a full dump of the Route Table from a given node. Wildcard messages are intended to apply to all routes, including routes decorated with additional data and AE values to be defined by future extensions, and hence this specification extends wildcard operations to apply to all routes, whatever the value of the source prefix.

More precisely, a node receiving an Update with the AE field set to 0 and the Metric field set to infinity (a wildcard retraction) MUST apply the route acquisition procedure described in Section 3.5.3 of [RFC8966] to all of the routes that it has learned from the sending node, whatever the value of the source prefix. A node MUST NOT send a wildcard retraction with an attached source prefix, and a node that receives a wildcard retraction with a source prefix MUST ignore the retraction.

Similarly, a node that receives a route request with the AE field set to 0 (a wildcard route request) SHOULD send a full routing table dump, including routes with a non-zero length source prefix. A node

MUST NOT send a wildcard request that carries a source prefix, and a node receiving a wildcard request with a source prefix MUST ignore the request.

6. Compatibility with the base protocol

The protocol extension defined in this document is, to a great extent, interoperable with the base protocol defined in [RFC8966] (and all previously standardised extensions). More precisely, if non-source-specific routers and source-specific routers are mixed in a single routing domain, Babel's loop-avoidance properties are preserved, and, in particular, no persistent routing loops will occur.

However, this extension is encoded using mandatory sub-TLVs, introduced in [RFC8966], and therefore is not compatible with the older version of the Babel Routing Protocol [RFC6126] which does not support mandatory sub-TLVs. Consequently, this extension MUST NOT be used in a routing domain in which some routers implement RFC 6126, otherwise persistent routing loops may occur.

6.1. Starvation and Blackholes

In general, the discarding of source-specific routes by non-source-specific routers will cause route starvation. Intuitively, unless there are enough non-source-specific routes in the network, non-source-specific routers will suffer starvation, and discard packets for destinations that are only announced by source-specific routers.

In the common case where all source-specific routes are originated at one of a small set of edge routers, a simple yet sufficient condition for avoiding starvation is to build a connected source-specific backbone that includes all of the edge routers, and announce a non-source-specific default route towards the backbone.

7. Protocol Encoding

This extension defines a new sub-TLV used to carry a source prefix: the Source Prefix sub-TLV. It can be used within an Update, a Route Request or a Seqno Request TLV to match a source-specific entry of the Route Table, in conjunction with the destination prefix natively carried by these TLVs.

Since a source-specific routing entry is characterized by a single destination prefix and a single source prefix, a source-specific contains message exactly one Source Prefix sub-TLV. A node MUST NOT send more than one Source Prefix sub-TLV in a TLV, and a node receiving more than one Source Prefix sub-TLV in a single TLV MUST ignore the TLV. It MAY ignore the whole packet.

7.1. Source Prefix sub-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 = 128  |    Length     |  Source Plen  | Source Prefix...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Fields:

Type   Set to 128 to indicate a Source Prefix sub-TLV.

Length   The length of the body, in octets, exclusive of the Type and Length fields.

Source Plen The length of the advertised source prefix, in bits. This MUST NOT be 0.

Source Prefix The source prefix being advertised. This field's size is (Source Plen)/8 octets rounded upwards.

The length of the body TLV is normally of size 1+(Source Plen)/8 rounded upwards. If the Length field indicates a length smaller than that, then the sub-TLV is corrupt, and the whole enclosing TLV must be ignored; if the Length field indicates a length that is larger, then the extra octets contained in the sub-TLV MUST be silently ignored.

The contents of the Source Prefix sub-TLV are interpreted according to the AE of the enclosing TLV. If a TLV with AE equal to 0 contains a Source Prefix sub-TLV, then the whole enclosing TLV MUST be ignored. If a TLV contains multiple Source Prefix sub-TLVs, then the whole TLV MUST be ignored.

Note that this sub-TLV is a mandatory sub-TLV. Therefore, as described in Section 4.4 of [RFC8966], the whole TLV MUST be ignored if that sub-TLV is not understood (or malformed).

7.2. Source-specific Update

The source-specific Update is an Update TLV with a Source Prefix sub-TLV. It advertises or retracts source-specific routes in the same manner as routes with non-source-specific Updates (see [RFC8966]). A wildcard retraction (Update with AE equal to 0) MUST NOT carry a Source Prefix sub-TLV.

Babel uses a stateful compression scheme to reduce the size taken by destination prefixes in update TLVs (see Section 4.5 of [RFC8966]). The source prefix defined by this extension is not compressed. On the other hand, compression is allowed for the destination prefixes carried by source-specific updates. As described in Section 4.5 of [RFC8966], unextended implementations will correctly update their parser state while otherwise ignoring the whole TLV.

7.3. Source-specific Route Request

A source-specific Route Request is a Route Request TLV with a Source Prefix sub-TLV. It prompts the receiver to send an update for a given pair of destination and source prefixes, as described in Section 3.8.1.1 of [RFC8966]. A wildcard request (Route Request with AE equals to 0) MUST NOT carry a Source Prefix sub-TLV; if a wildcard request with a Source Prefix sub-TLV is received, then the request MUST be ignored.

7.4. Source-Specific Seqno Request

A source-specific Seqno Request is a Seqno Request TLV with a Source Prefix sub-TLV. It requests the receiving node to perform the procedure described in Section 3.8.1.2 of [RFC8966], but applied to a pair of a destination and source prefix.

8. IANA Considerations

IANA has allocated sub-TLV number 128 for the Source Prefix sub-TLV in the Babel sub-TLV types registry.

9. Security considerations

The extension defined in this document adds a new sub-TLV to three sub-TLVs already present in the original Babel protocol, and does not change the security properties of the protocol itself. However, the additional flexibility provided by source-specific routing might invalidate the assumptions made by some network administrators, which could conceivably lead to security issues.

For example, a network administrator might be tempted to abuse route filtering (Appendix C of [RFC8966]) as a security mechanism. Unless the filtering rules are designed to take source-specific routing into account, they might be bypassed by a source-specific route, which might cause traffic to reach a portion of a network that was thought to be protected. A network administrator might also assume that no route is more specific than a host route, and use a host route in order to direct traffic for a given destination through a security device (e.g., a firewall); source-specific routing invalidates this assumption, and in some topologies announcing a source-specific route might conceivably be used to bypass the security device.

10. Acknowledgments

The authors are indebted to Donald Eastlake, Joel Halpern, and Toke Hoiland-Jorgensen for their help with this document.

11. References

11.1. Normative References

[BCP84] Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks", BCP 84, RFC 3704, March 2004, <https://www.rfc-editor.org/rfc/rfc3704>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8966] Chroboczek, J. and D. Schinazi, "The Babel Routing Protocol", RFC 8966, DOI 10.17487/RFC8966, January 2021, <https://www.rfc-editor.org/info/rfc8966>.

11.2. Informative References

[RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, DOI 10.17487/RFC3484, February 2003, <https://www.rfc-editor.org/info/rfc3484>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <https://www.rfc-editor.org/info/rfc4960>.

[RFC6126] Chroboczek, J., "The Babel Routing Protocol", RFC 6126, DOI 10.17487/RFC6126, April 2011, <https://www.rfc-editor.org/info/rfc6126>.

[RFC8305] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency", RFC 8305, DOI 10.17487/RFC8305, December 2017, <https://www.rfc-editor.org/info/rfc8305>.

[RFC8445] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", RFC 8445, DOI 10.17487/RFC8445, July 2018, <https://www.rfc-editor.org/info/rfc8445>.

[RFC8684] Ford, A., Raiciu, C., Handley, M., Bonaventure, O., and C. Paasch, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 8684, DOI 10.17487/RFC8684, March 2020, <https://www.rfc-editor.org/info/rfc8684>.

[SS-ROUTING]
Boutier, M. and J. Chroboczek, "Source-Specific Routing", August 2014, <http://arxiv.org/pdf/1403.0445>. In Proc. IFIP Networking 2015.

Authors' Addresses

Matthieu Boutier
IRIF, University of Paris
Case 7014
75205 Paris Cedex 13
France

Email: boutier@irif.fr

Juliusz Chroboczek
IRIF, University of Paris
Case 7014
75205 Paris Cedex 13
France

Email: jch@irif.fr