Network Working Group   M. Boutier
Internet-Draft   J. Chroboczek
Intended status: Informational   PPS, University of Paris-Diderot
Expires: January 4, 2014   July 3, 2013

Source-specific Routing
draft-boutier-homenet-source-specific-routing-00

Abstract

Source-specific routing is a generalisation of next-hop routing in which the routing decision is made depending on a packet's source address in addition to the destination. We describe the motivation for source-specific routing and our experiences with an experimental extension of the Babel routing protocol that implements source-specific routing.

described in the Simplified BSD License.

Table of Contents

1. Introduction

The main routing paradigm deployed on the Global Internet is next-hop routing. In next-hop routing, routing decisions are performed per-packet, and consist in examining a packet's destination address only, and mapping it to a next-hop router.

The use of next-hop routing restricts the flexibility of the routing system in two ways. First, since a router only controls the next hop, a route can only be selected by the network if it has a selected route as its suffix, which makes some forms of global optimisation difficult or impossible. Other routing paradigms, such as circuit switching, label switching and source routing, do not have this limitation. (Source-routing, in particular, has been proposed multiple times as a suitable routing paradigm for the Global Internet [CLARK]), but has been forbidden due to claimed security reasons [RFC5095].

Second, the only decision criterion used by a router is the destination address. This implies that two packets with the same destination are routed identically, which is not always desirable. There are other data in the IP header that can be reasonably used for making a routing decision -- the TOS octet, the flow-id, and, of course, the source address.

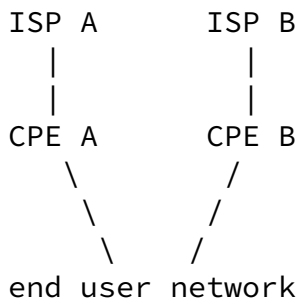
2. Source-specific routing

Source-specific routing is a modest extension of next-hop routing. In source-specific routing, just like in next-hop routing, a router's role is limited to computing a next hop. Unlike in next-hop routing, however, it can use both the destination and the source address in order to perform this computation. In effect, source-specific routing gives a modest amount of routing control to the sending host, which can choose among potentially many source addresses, while leaving routing decisions firmly in the control of the routers.

2.1. Usage scenarios

2.1.1. Simple multihoming

Consider a multihomed network connected to two (or more) providers, for example a home network with two ADSL lines, or one ADSL line and one cellular connection. We assume no cooperation between the two providers, so that there are two edge routers ("CPEs"), one for each provider. We further assume that one or both ISPs might be hostile to multihoming, so that solutions requiring changes to the on-the-wire packet format are not applicable.



Each provider grants the network a range of addresses that can be assigned to nodes. A node can choose to configure with an address from one of the two ranges, or to configure two addresses, one from each range; it will then need to choose, for each packet being sent, an address to use as the source address.

Since providers hopefully implement source address filtering [BCP38], the network must choose the edge router through which to route a packet depending on its source.

2.1.2. Tunnels and VPNs

Tunnels and VPNs are commonly used to establish a network-layer topology that is different from the physical topology, notably for security reasons. In many tunnel or VPN deployments, the end network uses its native default route, and only routes some set of prefixes through the tunnel or VPN.

In some deployments, however, the default route points at the tunnel. If this is done naively, the network stack attempts to route the

encapsulated packets through the tunnel itself, which causes the tunnel to break. Many workarounds are possible, the simplest being to point a host route towards the tunnel endpoint through the native interface.

Source-specific routing provides a clean solution to that problem. The native default route is kept unchanged, while a source-specific default route is installed through the tunnel. The source-specific route being more specific than the native default route, packets from the user network are routed through the tunnel, while the encapsulated packets sourced at the edge router follow the native, non-specific route.

2.2. Routing tables

In classical next-hop routing, every router maintains a routing table, a set of pairs (D, NH), where D is a destination prefix and NH the corresponding next-hop router. When a packet with destination address d is routed, an entry (D, NH) such that d is in D is

selected, and the packet is sent to the corresponding NH.

In a source-specific router, the routing table is a set of triples of the form (D, S, NH), where D is a destination prefix, S a source prefix, and NH the next-hop router. When a packet with destination d and source s is routed, an entry (D, S, NH) is selected such that d is in D and s is in S, and the packet is sent to the corresponding NH.

2.2.1. Ambiguity

The two procedures described above omit an important detail: in general, there are multiple routing table entries that match a given packet. A router must therefore choose one among these entries in order to determine a next hop.

In next-hop routing, if two routing table entries overlap, then one is necessarily more specific than the other; the "longest prefix rule" specifies that the most specific applicable routing table entry is chosen. In source-specific routing, this is no longer the case: there might, in general, be multiple applicable entries with none being included in the others.

Consider the following fragment of a routing table:

```
(2001:DB8:0:1::/64, ::/0, A)
(::/0, 2001:DB8:0:2::/64, B)
```

This specifies that all packets with destination in 2001:DB8:0:1::/64 are to be routed through A, while packets with a source in 2001:DB8:0:2::/64 are to be routed through B. A packet with source 2001:DB8:0:2::42 and destination 2001:DB8:0:1::57 matches both rules, although neither is more specific than the other.

We say that a routing table such as the above is ambiguous. Most practical routing tables with source-specific routes turn out to be ambiguous.

2.2.2. Resolving ambiguity

In the presence of ambiguity, routing tables should be considered by destination first; intuitively, "the destination wins". (We are indebted to Fred Baker, who explained that to us.)

Consider the following network topology:

```
::/0 --- A --- B --- C --- 2001:DB8:0:1::/64
```


Suppose that the routing table at B contains a source-specific default route through A and a non-specific route towards 2001:DB8:0:1::/64 through C. The correct behaviour is clearly to send a packet destined to 2001:DB8:0:1::/64 through C -- this is the only choice that has a chance of getting the packet to the right destination.

It is important to note that all routers in the same routing domain must have the exact same behaviour in the presence of ambiguity, lest persistent routing loops occur. Indeed, consider again the example above; if router C implements a "source first" disambiguation behaviour, then it will send B's packets back to B, which in turn will send it back to B, etc.

2.2.3. Disambiguation routes

Ideally, we would like the lower layers of the system (the OS kernel, the line cards, etc.) to implement source-specific routing tables out of the box, with the right disambiguation behaviour already present. In practice, however, we have found that such lower-layer support either doesn't exist, doesn't work, or has a behaviour different from the one desired.

In order to work with the limitations of the lower layers, we introduce disambiguation routes. A disambiguation route is a route that covers the intersection of two ambiguous routes, and therefore specifies the behaviour of packets that match both. Disambiguation routes do not appear on the wire, and in our implementation are not even inserted into the RIB; they are computed and inserted into the FIB on the fly, at route selection time. From the point of view of the routing protocol, disambiguation routes are a lower level implementation detail.

Interestingly enough, we have found that we do not need to maintain a list of disambiguation routes that we have installed: when removing a route from the FIB, the set of disambiguation routes that need to be removed can be computed on the fly, similarly to what happens during route insertion.

3. Implementation

We have implemented a source-specific variant of the Babel routing protocol [BABEL] for the Linux kernel. We first attempted to use the source-specific API provided by Linux; it turns out that this API is specific to IPv6, and only works in a very restricted case, insufficient for our needs.

We have therefore chosen to use the "rule" API, which allows a

routing daemon to use multiple routing tables that are combined using a fairly flexible set of "rules". We use a dynamically allocated set of routing tables, and manage routing rules on the fly. The use of disambiguation routes is essential to obtaining the right behaviour.

4. Interoperability issues

4.1. Interoperability with next-hop routing

In many networks, only some routers will need to perform source-specific routing decisions. For example, in a typical multihomed network the two specific default routes will match in most of the network, and only be distinguished near the edge routers. Our implementation allows running a base version of Babel within most of the network, and only run a source-specific daemon where the specific routes are distinct.

Source-specific routes are encoded within the protocol as a new TLV type, in accordance with the Babel extension mechanism [BABEL-EXT]. This new TLV will be silently ignored by base Babel routers, which will therefore route packets following non-specific routes only.

Hybrid networks consisting of base and source-specific routers do not cause persistent routing loops. However, since non-specific routers do not see source-specific routes, they might drop packets unless they have enough non-specific routes; distributing a non-specific default route throughout the network solves this particular issue in all cases.

Additionally, since non-specific routers do not propagate specific routes, packets may end up routed to the wrong destination unless there are enough specific routers to propagate all the specific routes throughout the network. A simple solution is to ensure that the specific routers form a connected subgraph, which, at worst, can be achieved by using tunnels. Intuitively, such a network consists of a source-specific backbone together with a set of non-specific leaf networks.

4.2. Other forms of specific routing

The technology described in this document is fully general, and applies equally well to other forms of specific routing (say, TOS-specific or flow-id-specific routing). In the presence of multiple forms of specific routing, a natural question to ask is whether they can interoperate in a single routing domain.

In general, such interoperability is possible assuming that the

preference rules of all the implementations are subsets of a single

total order on all of the routing criteria; equivalently, there must exist a consistent linearisation of all of the orderings used by the different implementations. Indeed, consider again a simple linear topology:

```
X --- A --- B --- Y
```

Suppose that X announces a source-specific default route, while Y announces a flow-id-specific default route. A packet that matches both routes must be treated consistently by A and B, lest a routing loop arise.

A simple (if brutal) way of meeting the linearisation requirement is to require all routers to be specific in one dimension only: to allow a router to perform source-specific routing, flow-id-specific routing, but not both at the same time.

5. Applicability to link-state protocols

While our implementation is an extension of the Babel routing protocol, our work applies equally well to any distance vector routing protocol, such as RIPv2, RIPng or EIGRP. The question remains about link-state routing protocols.

The currently deployed link-state protocols (OSPF and IS-IS) are actually hybrid protocols: they divide the network into areas, and perform link-state routing within areas and distance-vector routing within areas. We are therefore confident that our techniques can be used to extend link-state protocols with source-specific inter-area routing (a simplified case of that has been implemented for OSPF [STENBERG]); in OSPF terms, source-specific routes are analoguous to Type 5 LSAs.

Whether it is possible to extend the current link-state protocols with support for intra-area source-specific routing, or whether it is desirable to do so, are currently open questions.

6. Conclusions

Source-specific routing is a modest extension to ordinary next-hop routing that makes a number of useful scenarios possible. In this document, we have described the difficulties associated with source-specific routing, and described the solution we adopted within our implementation of source-specific routing within the Babel routing protocol.

We expect our experience to be useful to future implementers of source-specific routing within other routing protocols.

7. References

[BABEL] Chroboczek, J., "The Babel Routing Protocol", RFC 6126, February 2011.

[BABEL-EXT] Chroboczek, J., "Extension Mechanism for the Babel Routing Protocol", Internet Draft draft-chroboczek-babel-extension-mechanism-00, June 2013.

[BAKER] Baker, F., "IPv6 Source/Destination Routing using OSPFv3", Internet Draft draft-baker-ipv6-ospf-dst-src-routing-00, February 2013.

[BCP38] 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.

[CLARK] Reed, D. and D. Clark, "Source Routing for Campus-wide Internet Transport", September 1980.

[RFC5095] Abley, J., Savola, P., and G. Neville-Neil, "Deprecation of Type 0 Routing Headers in IPv6", RFC 5095, December 2007.

[STENBERG] Stenberg, M., "Hnet (core) package", 2012, <https://github.com/fingon/hnet-core>.

[TROAN] Troan, O. and L. Colitti, "IPv6 Multihoming with Source Address Dependent Routing (SADR)", Internet Draft draft-troan-homenet-sadr-00.

Authors' Addresses

Matthieu Boutier
PPS, University of Paris-Diderot
Case 7014
75205 Paris Cedex 13,
France

Email: boutier@pps.univ-paris-diderot.fr

Juliusz Chroboczek
PPS, University of Paris-Diderot
Case 7014
75205 Paris Cedex 13,
France

Email: jch@pps.univ-paris-diderot.fr