Source-sensitive routing

Matthieu Boutier, Juliusz Chroboczek
Laboratoire PPS
Université Paris-Diderot (Paris 7)

29 July 2013

Next-hop routing

Most of the internet uses next-hop routing.

- a router examines the destination of a packet;
- a router chooses the next hop only.

Routing table: maps prefix to next hop:

$$(\texttt{2001:DB8:0:2::/64}, B)$$
$$(\texttt{2001:DB8:0:3::/64}, C)$$

Next-hop routing: specificity

In general, routing tables are ambiguous.

Internet —— A —— B —— C —— LAN

$$(\texttt{2001:DB8:0:2::/64}, C)$$
$$(\texttt{::/0}, A)$$

If a packet is destined to 2001:DB8:0:2::42,
both entries match.

The entry chosen is the most specific :
“longest prefix rule”.

Next-hop routing: specificity (2)

The entry chosen is the most specific :
"longest prefix rule".

A prefix P is more specific than Q,

$$P \leq Q$$

when

$$\text{for all packets } p, p \in |P| \text{ implies } p \in |Q|$$

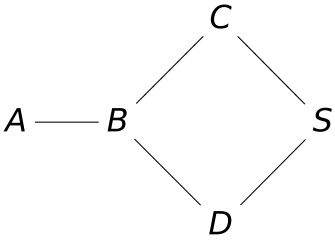
or, equivalently,

$$|P| \subseteq |Q|.$$

Property: any two prefixes are either disjoint or ordered. We call this a locally total order.

Limitations

Limitations: some routing policies cannot be implemented by next-hop routing.

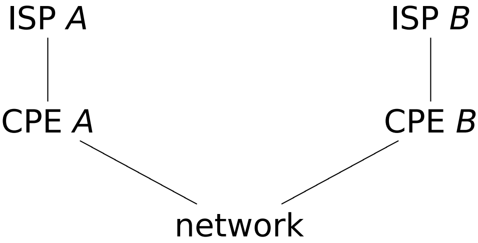


If B has selected C as its next hop to S, then there is no way A can send its packets to S through D. (The route $A \cdot B \cdot D \cdot S$ cannot be selected since its prefix $B \cdot D \cdot S$ has not been selected.)

B should chose the nexthop depending on the packet's source address.

Limitations (2)

Home network connected to two ISPs:



There are two default routes!

The network must choose the right CPE depending on the source address.

Limitations (3)

Network with a tunnel (VPN).

If the tunnel announces a default route, again, there are two default routes. The tunnel has a tendency to enter itself.

Usual solution: host route towards the tunnel endpoint.
Cleaner solution: packets routed depending on their source.

Manually configured routing rules

Under Linux, such situations are usually solved by using manually configured routing rules:

```
ip rule add from 192.168.4.0/24 table 4
ip rule add from 0.0.0.0/0 table 5
```

Similar features exist in some other OSes.

Not applicable to homenet:

- manual configuration;
- fixed topology.

Source routing

Fully general solution source routing.

In source routing, the sending host determines the full route to the destination and inserts it in the packet header. Routers are dumb.

See [Clark 1980] for a convincing argument in favour of source routing.

Source routing

Fully general solution source routing.

In source routing, the sending host determines the full route to the destination and inserts it in the packet header. Routers are dumb.

See [Clark 1980] for a convincing argument in favour of source routing.

Not usable in the Global Internet :

- source-routed packets are easily identified and shot down by a hostile ISP;
- recently forbidden for claimed security reasons (RFC 5095).

Source-sensitive routing

Source-sensitive or source-specific routing is a mild generalisation of next-hop routing.
A router still chooses just the next hop, but can examine both the destination and the source.

Routing tables now map (dest, source) pairs ("patterns") to next hops :

$$(\texttt{2001:DB8:0:2::/64}, \texttt{::/0}, B)$$
$$(\texttt{2001:DB8:0:3::/64}, \texttt{::/0}, C)$$
$$(\texttt{::/0}, \texttt{2001:DB8:0:2::/64}, D)$$
$$(\texttt{::/0}, \texttt{2001:DB8:0:3::/64}, E)$$

Note: we write the destination first unlike [Troan 2013].

Source-sensitive routing (2)

Source-sensitive routing is a compromise between next-hop routing and source routing:

- routing choices are firmly in the hands of the routers (like in next-hop routing);
- hosts can communicate their routing choices to the network by choosing a source address.

The largest subset of source routing that's deplyable?

Source-sensitive routing: specificity

Recall the specificity ordering :

$(D, S) \leq (D', S')$ when $p \in |(D, S)|$ implies $p \in |(D', S')|$

This is a pointwise product:

$$(D, S) \leq (D', S') \text{ when } D \leq D' \text{ and } S \leq S'.$$

Unfortunately, this is no longer a (locally) total order.

Source-sensitive routing: ambiguity

The following pair of patterns are neither disjoint nor ordered:

$$(\texttt{2001:DB8:0:2::/64}, \texttt{::/0})$$
$$(\texttt{::/0}, \texttt{2001:DB8:0:3::/64})$$

A packet destined to `2001:DB8:0:2::1` and sourced from `2001:DB8:0:3::1` matches both patterns.

Source-sensitive routing: ambiguity

The following pair of patterns are neither disjoint nor ordered:

$$(\texttt{2001:DB8:0:2::/64}, \texttt{::/0})$$
$$(\texttt{::/0}, \texttt{2001:DB8:0:3::/64})$$

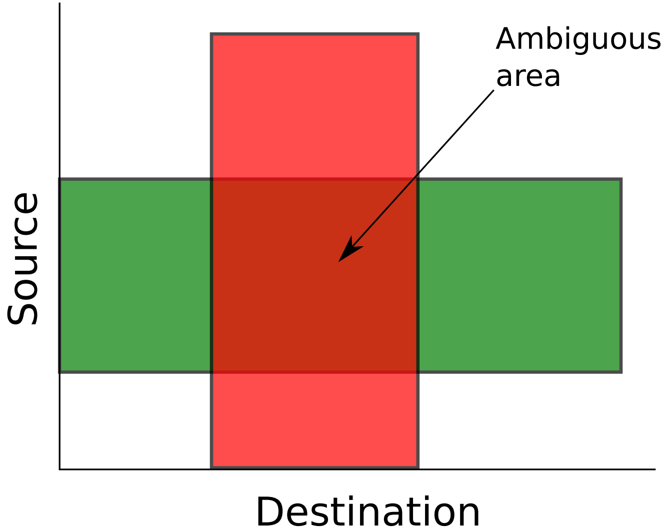
A packet destined to `2001:DB8:0:2::1` and sourced from `2001:DB8:0:3::1` matches both patterns.

Therefore, the following routing table is ambiguous:

$$(\texttt{2001:DB8:0:2::/64}, \texttt{::/0}, B)$$
$$(\texttt{::/0}, \texttt{2001:DB8:0:3::/64}, C)$$

We call this situation a conflict.

Source-sensitive routing: ambiguity (2)



Solving ambiguity (1)

In order to resolve conflicts, we need to choose a disambiguation rule.

Properties:

- the disambiguation rule must induce a locally total ordering (else conflicts);
- the disambiguation rule must be the same for all routers (else persistent routing loops).

Any linearisation of the specificity ordering will work, as long as it satisfies the above properties.

Solving ambiguity (2)

Destination wins

Consider the following topology:

$$\text{Internet} \;\text{———}\; A \;\text{———}\; B \;\text{———}\; C \;\text{———}\; \text{LAN}$$

A announces a source-sensitive default route

$$(\texttt{::/0}, \texttt{2001:DB8:0:3::/64})$$

while C announces a route towards its connected LAN:

$$(\texttt{2001:DB8:0:3::/64}, \texttt{::/0})$$

A packet from B that matches both routes should be routed toward C — the only choice that has a chance of reaching the LAN.

In case of conflict, the destination wins.
The same semantics has been proposed by Troan, Baker and others.

Solving ambiguity (3)

Destination wins

In case of conflict, the destination wins.

$$
(D, S) \leq (D', S') \quad \text{when} \quad D < D' \quad \text{or} \quad D = D' \text{ and } S \leq S'
$$

This is just the lexical product of destination by source.

(This is one reason why we write destination first in our routing tables.)

Implementation

Two existing implementations:

- Stenberg: special case for OSPFv3 (BIRD) on Linux;
- Boutier: fully general case for Babel on Linux.

(Rumours of a third implementation?)

Both implementations use the Linux rule API which has exactly the wrong semantics.

Something needs to be done to force the right behaviour in the presence of ambiguity.

Implementation (2)

Disambiguation routes

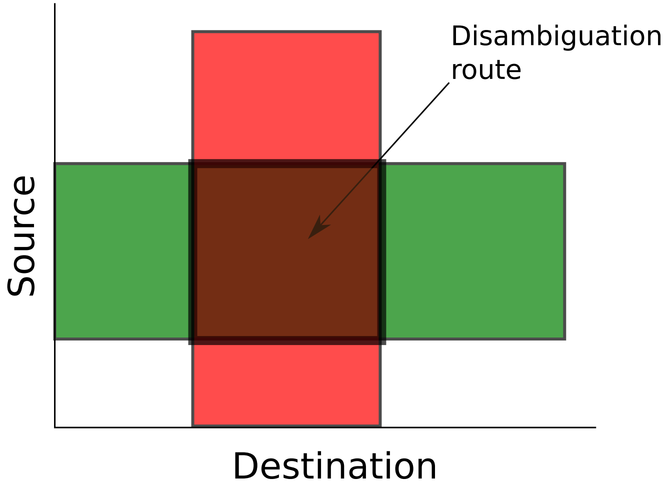
Linux's kernel API has the wrong semantics. We need to force behaviour in ambiguous cases.

Solution: insert enough disambiguation routes to avoid ambiguity.

$$(\texttt{2001:DB8:0:2::/64}, \texttt{::/0}, B)$$
$$(\texttt{::/0}, \texttt{2001:DB8:0:3::/64}, C)$$
$$(\texttt{2001:DB8:0:2::/64}, \texttt{2001:DB8:0:3::/64}, B)$$

Implementation (3)

Disambiguation routes



Implementation (4)

Boutier's algorithm computes disambiguation routes dynamically and inserts them in the kernel tables.

It does not keep a list of previously inserted disambiguation routes: recomputes the set of disambiguation routes when flushing a route.

Implementation (4)

Boutier's algorithm computes disambiguation routes dynamically and inserts them in the kernel tables.

It does not keep a list of previously inserted disambiguation routes: recomputes the set of disambiguation routes when flushing a route.

Boutier has a complete implementation of his algorithm. (Minor limitations when running multiple routing protocols on a single host.) Boutier's implementation manipulates kernel tables dynamically — no manual intervention (just like with an ordinary routing deamon).

`git://git.wifi.pps.univ-paris-diderot.fr/babels.git`

Interoperability with plain Babel

Boutier's fork of Babel interoperates with plain Babel:

- source-sensitive routes are encoded as a separate TLV, ignored by plain Babel;
- no persistent routing loops will occur whatever the topology;
- blackholes might occur unless source-sensitive routers form a connected subgraph of the network.

If the topology is wrong, a hybrid Babel network fails gracefully. This is analogous to what happens when filtering.

Interoperability with plain Babel

Boutier’s fork of Babel interoperates with plain Babel:

- source-sensitive routes are encoded as a separate TLV, ignored by plain Babel;
- no persistent routing loops will occur whatever the topology;
- blackholes might occur unless source-sensitive routers form a connected subgraph of the network.

If the topology is wrong, a hybrid Babel network fails gracefully. This is analogous to what happens when filtering.

It is not correct in general to cast source-sensitive routes to non-specific ones. Persistent routing loops might occur.

Terminology issues

We need help from people good at coining terms:

- source-sensitive routing? source-specific routing?
- (D, S) pair: pattern? generalised prefix? routing class?
- ordered or disjoint: locally-total order?
- partial and total specificity orderings: natural ordering and strong ordering?

Status and further work

- Production-quality implementation (done);
- more testing (in progress);
- merge the implementation into Babel (not yet);
- write a better Internet-Draft
  (good feedback (thanks!), in progress);
- write-down the algorithm (in progress);
- prove the algorithm correct
  (not difficult?);
- write a cool demo (Mosh? MPTCP?)
  (not started);
- work out interoperability issues
  (good progress);
- work out OSPF/IS-IS issues (not started).

Conclusion

Source-sensitive routing is a mild extension to next-hop routing that is deployable in practice, politically acceptable that solves a number of real-world problems some of which are relevant to homenet.

- Complete implementation exists and is freely available;
- interesting problems, theoretical, operational and practical;
- well-understood properties;
- write-up in progress.

Rejoice!