Source-specific routing *for* multihoming (multi-gateways) *in* wireless mesh networks

Matthieu Boutier
Juliusz Chroboczek

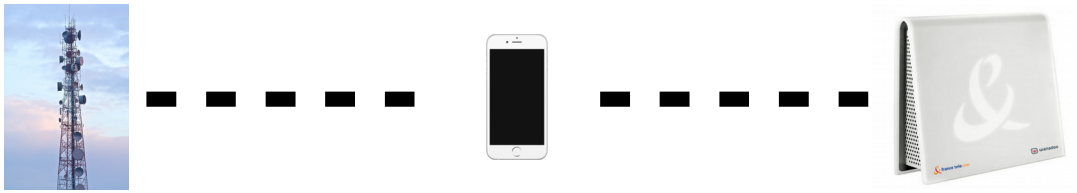
Laboratoire PPS - Université Paris Diderot
boutier@pps.univ-paris-diderot.fr

5 August 2015

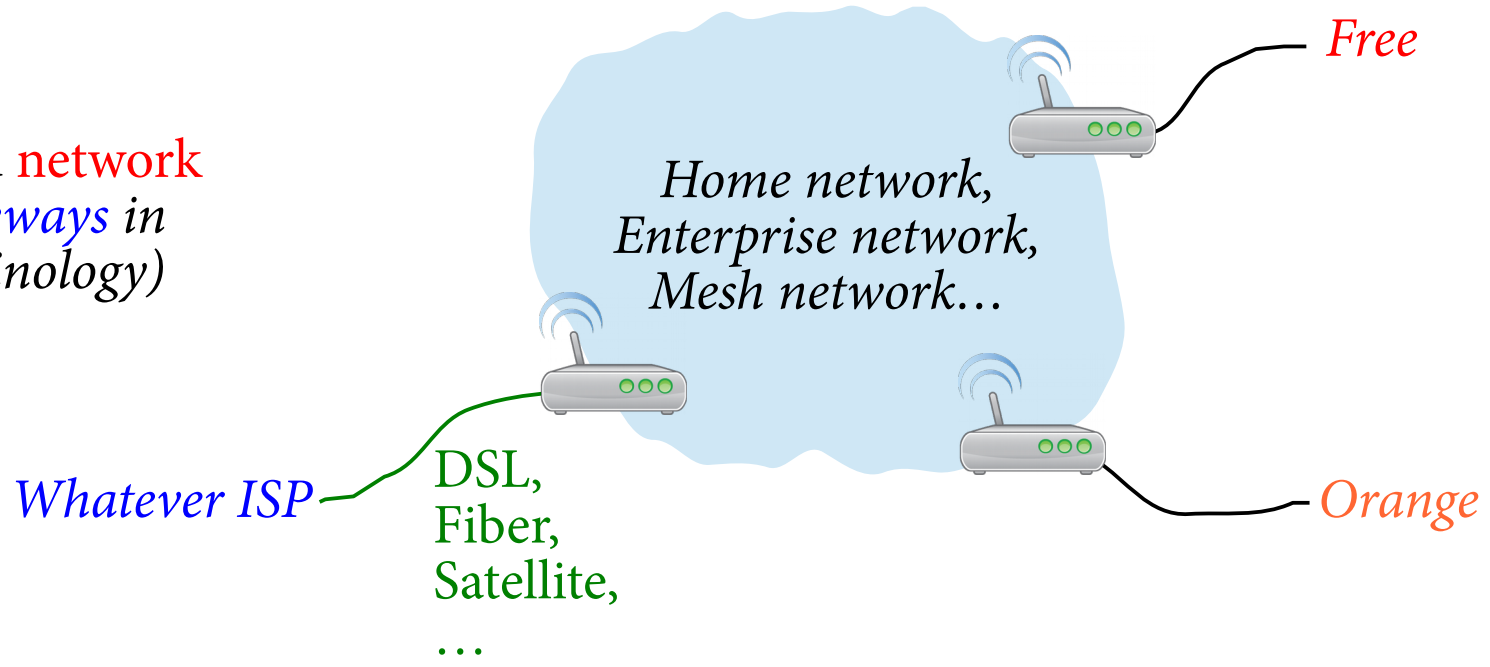
Wireless Battle of the Mesh v8

Multihoming

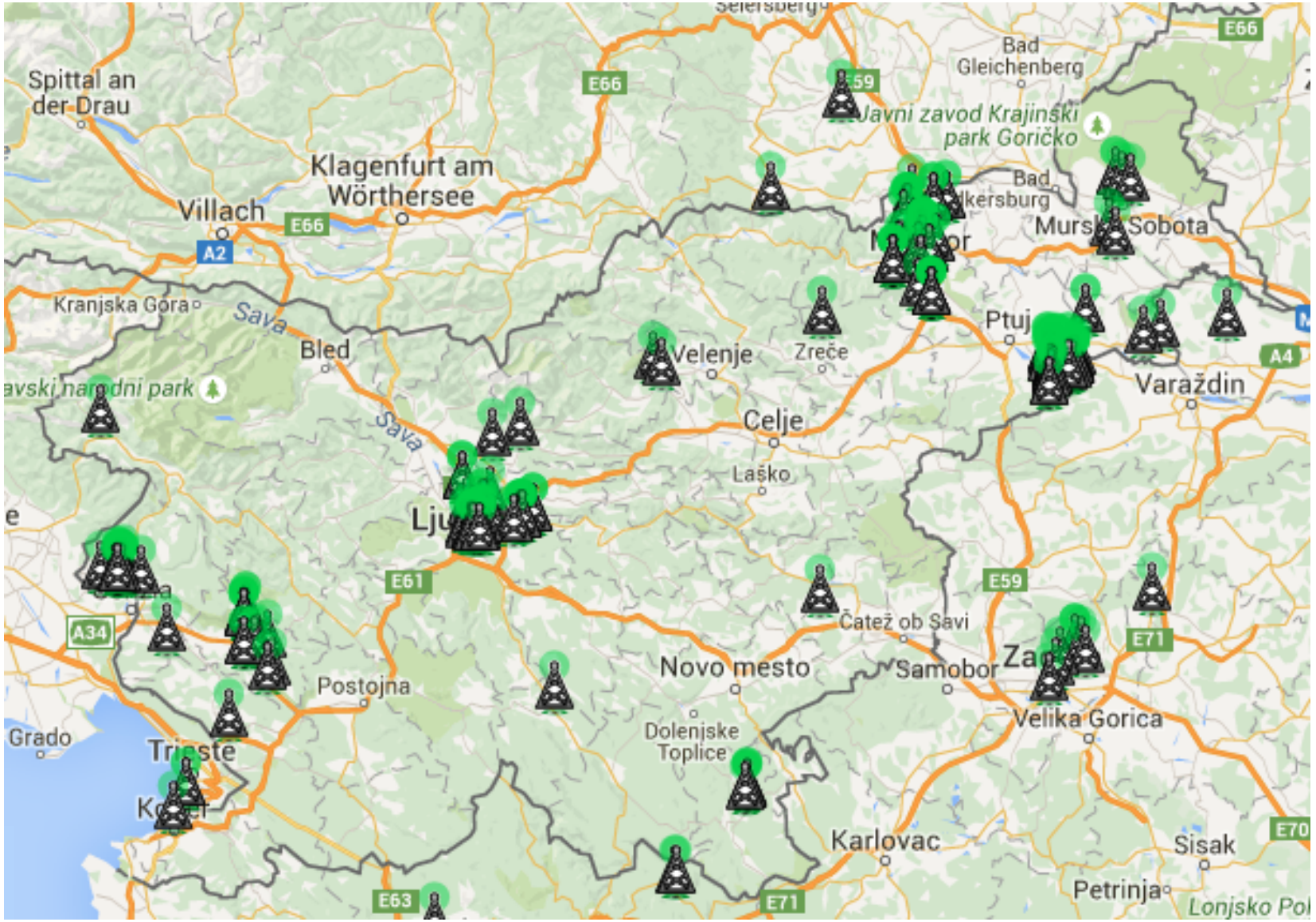
Multihomed hosts



Multihomed network
(*mutli-gateways in mesh terminology*)



We want Multihoming

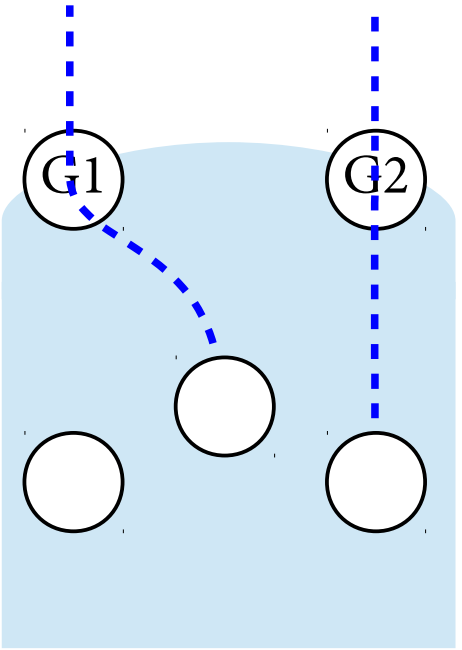


One gateway is not always enough!

Multihoming is difficult

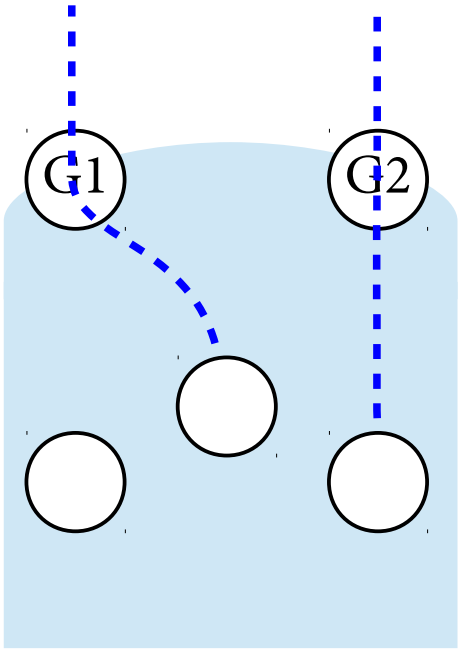
Multihoming provides multiple paths to the Internet:
- → reliability,
- → load balancing possibilities.

Problems are:
- → How to achieve reliability and performances,
  (performances: load balancing)
- → How to keep connections alive (TCP).



Classical Multihoming

Acquire a PI prefix (Provider Independent addresses)



Advantages:

- no configuration,
- reliability and partial load balancing,
  (no incoming traffic control)
- use the classical protocols.
  (routing, transport, application)

Flaws:

- Deal with ISP:
  → accept packets from the PI prefix,
  → announce the PI prefix to the Internet.
  (one more entry in the global routing table)

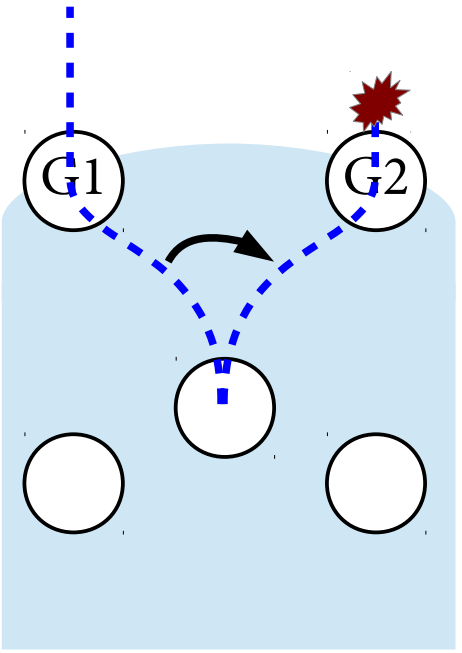
Host-centric Multihoming

Each ISP provides PD addresses.

Each gateway is bound to the addresses provided by the ISP.

→ If a TCP flow switch gateways, it will collapse.

Recall that a TCP connection is identified by:
→ source: (src addr, src port),
→ destination: (dst addr, dst port)



Many routing solutions exists!

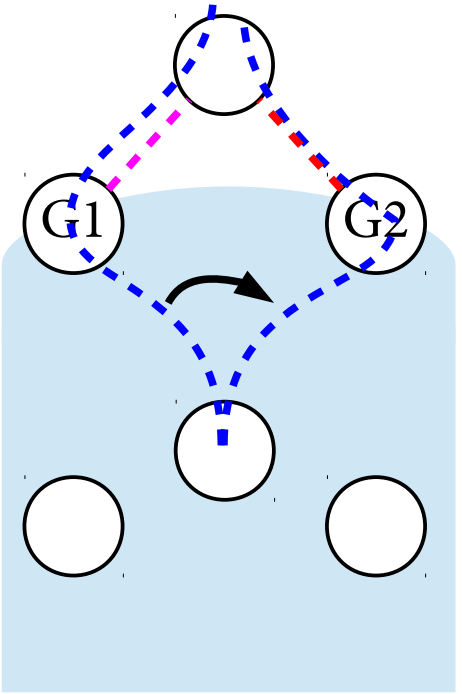
Multihoming: dedicated server

Advantages:

- Everything just works,
- use the classical protocols.
  (routing, transport, application)

Flaws:

- single point of failure,
- require an external server.



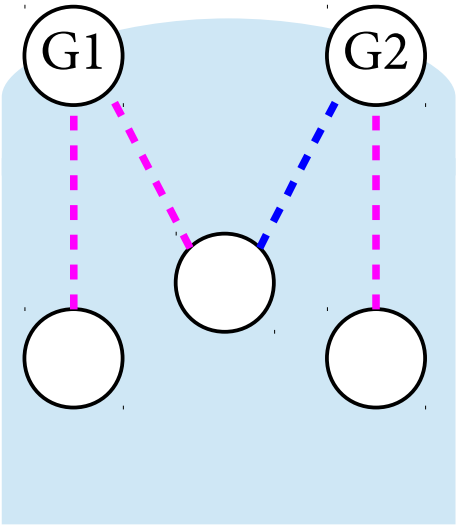
Multihoming: node-gateways tunnels

Advantages:

- Everything just works,
- fine tune your preferred gateway,
- keep TCP connections: you can't switch.

Flaws:

- The chosen gateway is not necessarily the best one,
- TCP connections dropped on gateway failure,
- Need to configure the tunnels:
  either manually or automatically.



Many active and deployed mesh network protocols provide quite automatic configuration:

- OLSRv1 smart gateways,
- BMX6 Tunnel Announcements,
- Batman-adv gateways.

Back to the problem

We want:

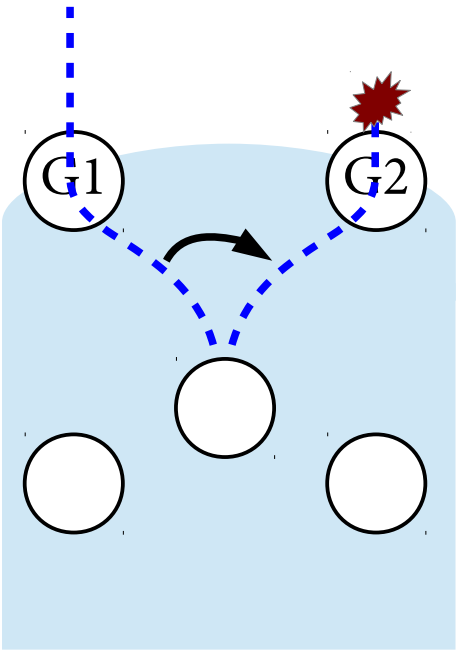
- to route packets to some gateway,
- to keep TCP connections alive,

But:

- TCP can't survive if the flow roam, because the source address associated to the gateway changes.

So we want:

- to keep the source address associated to that gateway.

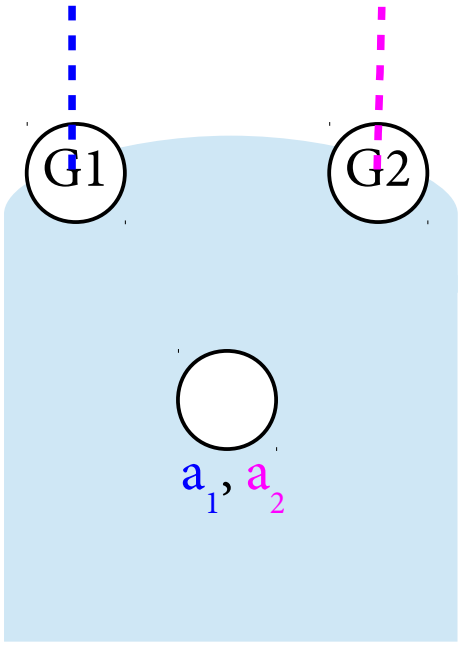


A natural idea…
from IPv6 multihoming

In IPv6, each ISP provides an infinite amount of addresses to the network.

Idea:

- each host receive one IP per provider,
- route packets depending on their source address.

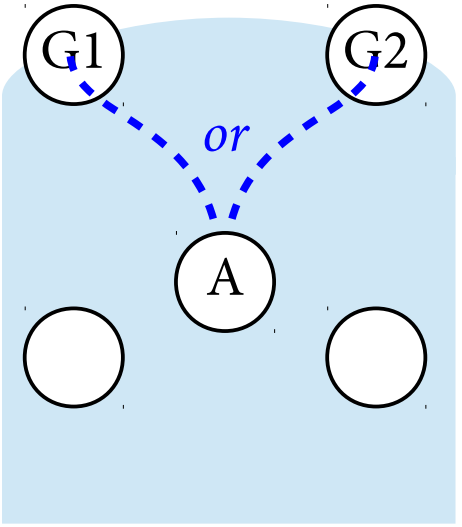


Classical routing in the Internet: next-hop routing

Router chooses, for each packet:

→ the next-hop

→ depending on the destination address only



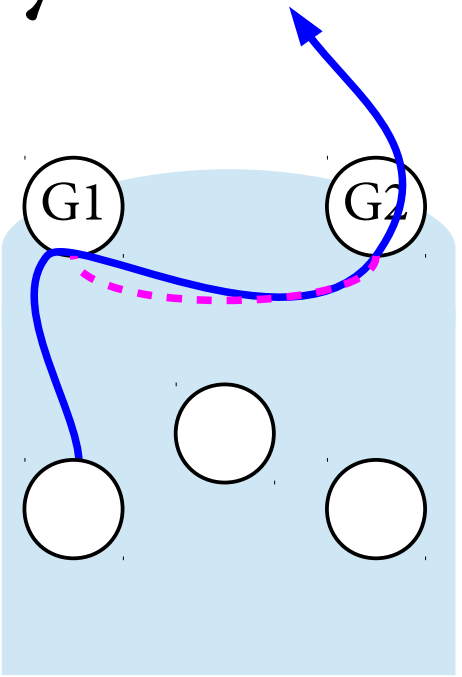
routing table of A

destination	next-hop
::/0	G1 *or* G2
…	…

A hack: tunnels between gateways

Uses classical routing protocol, with:

- tunnels between gateways,
- traffic engineering to choose the tunnel.
  (match the source address of the packets)



Advantages:

- use the classical protocols,
- keep TCP connections.

Flaws:

- much more configuration,
- TCP connections dropped on gateway failure,
- useless traffic between gateways.
  (probably not suitable for mesh networks)

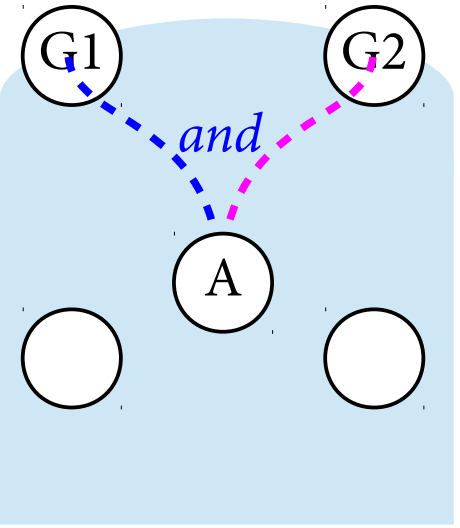
In brief: good for centrally-administrated-10GByte-wired-networks.

Dwarfs
standing on the shoulders of giants

A (now) natural idea: Source-specific routing

Modest extension of next-hop routing:
same paradigm, same mechanisms

Router chooses, for each packet:
→ the next-hop
→ depending on the destination and source addresses

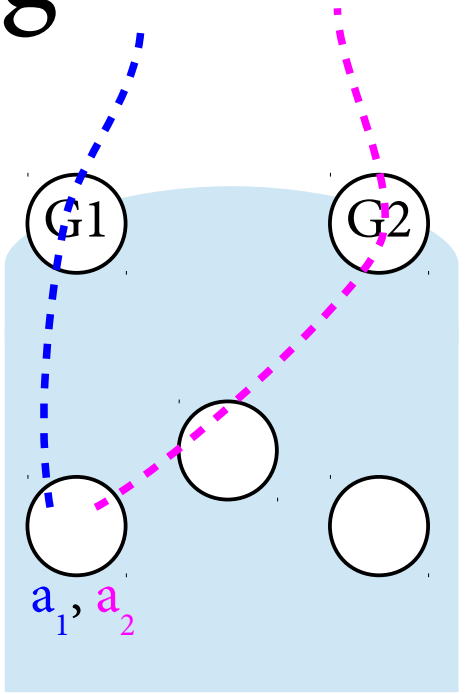


routing table of A

destination	source	next-hop
::/0	2001:db8:1::/48	G1
::/0	2001:db8:2::/48	G2
…	…	…

← *source-specific route entry*

Multihoming with Source-specific routing



Advantages:

- no configuration, just routing,
- keep TCP connections,
- *(bonus)* choose your paths!

Flaws:

- TCP connections dropped on gateway failure,
- Need to use a new routing protocol.

Source-specific routing implementations

Existing implementations (by date):

- OSPF (partial) ← by Markus Stenberg
- Babel ← by us (Matthieu Boutier and Juliusz Chroboczek)
  (first real, complete, production quality implementation)
- IS-IS (IPv6 only) ←by David Lamparter & Christian Franke
- OLSR (IPv6 only) ← by Henning Rogge

Perhaps future implementations:

→ B.A.T.M.A.N., BMX6… ???? *(this week ?)*

Going through Source-specific Babel

Source-specific routing is good for multihomed networks.

But it requires some changes:

- source-specific routing tables, (I will be boring for 2 minutes)
- compatible protocol extension,
- speaking to the kernel.

Need to choose the source address of outgoing packets.

Classic routing tables and ambiguity

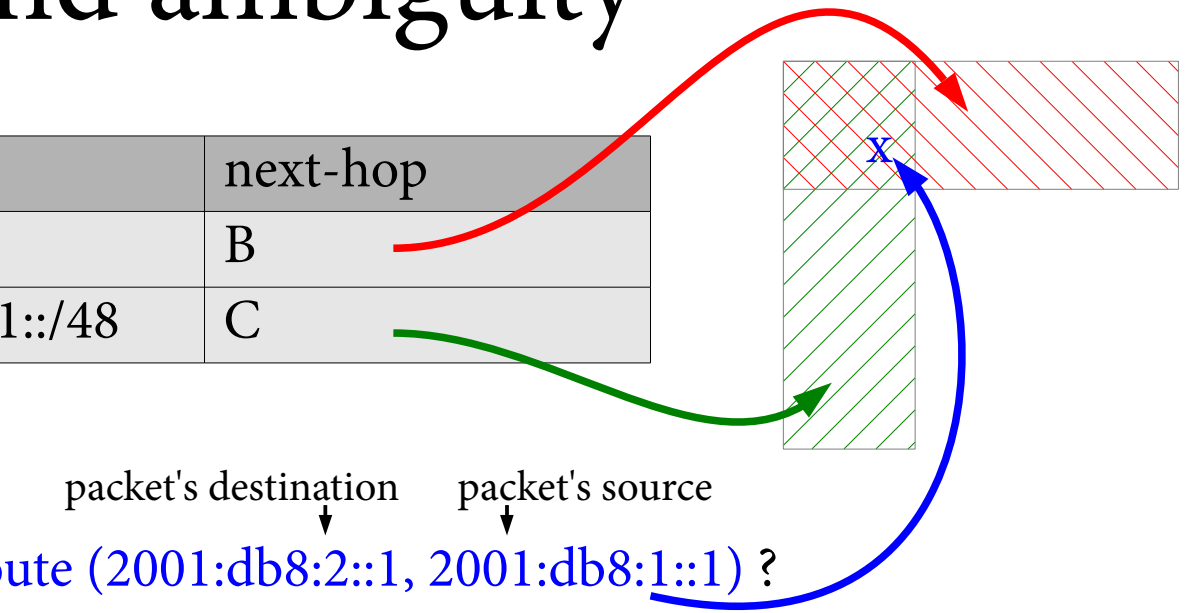
	destination	next-hop
→	2001:db8:2::/48	B
	::/0	C

These entries all match
the address 2001:db8:2::1

We choose the most specific prefix (*longest match rule*).
→ order induced by the inclusion.

Source-specific routing tables and ambiguity

destination	source	next-hop
2001:db8:2::/48	::/0	B
::/0	2001:db8:1::/48	C



→ we no longer have a total order on entries matching a single packet.

Consensus: routing by destination first.

(and all routers MUST have the same behaviour, or persistent routing loops occurs)

Source-specific Babel extension

Source-specific routing is good for multihomed networks.

We have seen how to interpret routing tables:
lexicographic order by destination first.

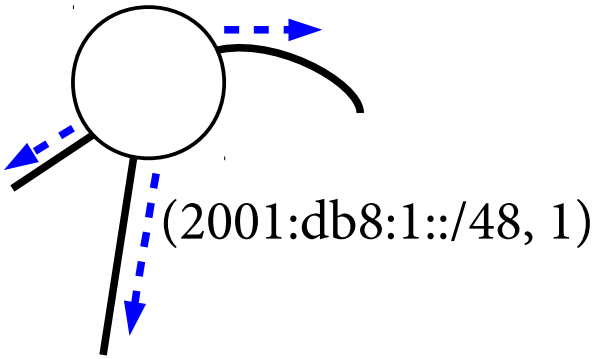
Extending Babel still needs:

- compatible protocol extension,
- speak to the kernel.

Need to choose the source address of outgoing packets.

Source-specific extension of Babel

classical Babel

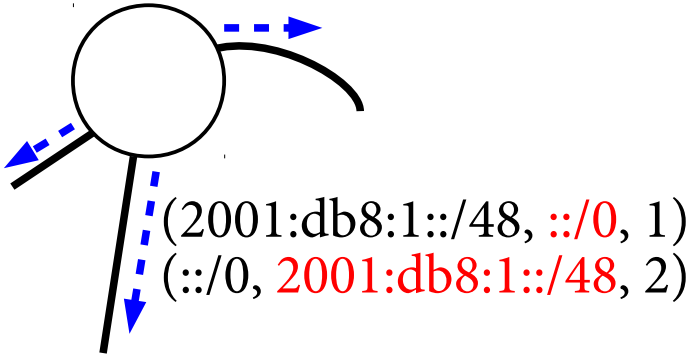


destination	metric	next-hop
…:1::/48	1	NH 1

Routes with source ::/0 are announced as non-specific.

Complete interoperability for non-specific routes,

source-specific Babel



destination	source	metric	next-hop
…:1::/48	::/0	1	NH 1
::/0	…:1::/48	2	NH 2

We use 3 new TLVs.

source-specific routes silently dropped by non-SS routers.
(not a problem with a SSR backbone.)

Source-specific Babel: last step

Source-specific routing is good for multihomed networks.

We have seen how to interpret routing tables:
lexicographic order by destination first.

We have a backward compatible Babel extension, which computes source-specific routing tables, such as:

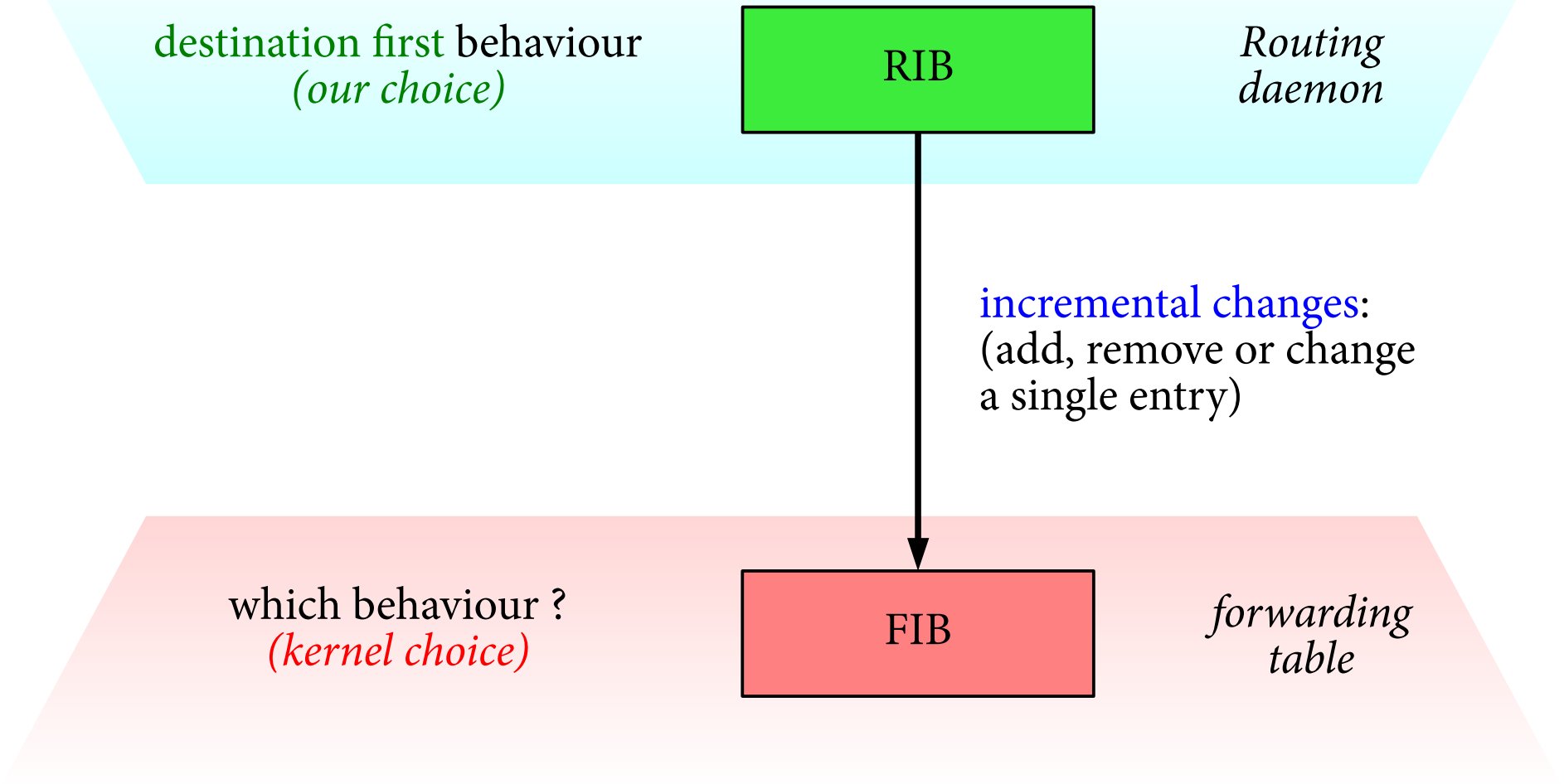
destination	source	next-hop
2001:db8:2::/48	0.0.0.0/0	B
0.0.0.0/0	2001:db8:1::/48	C

Extending Babel still needs:

- to speak to the kernel.

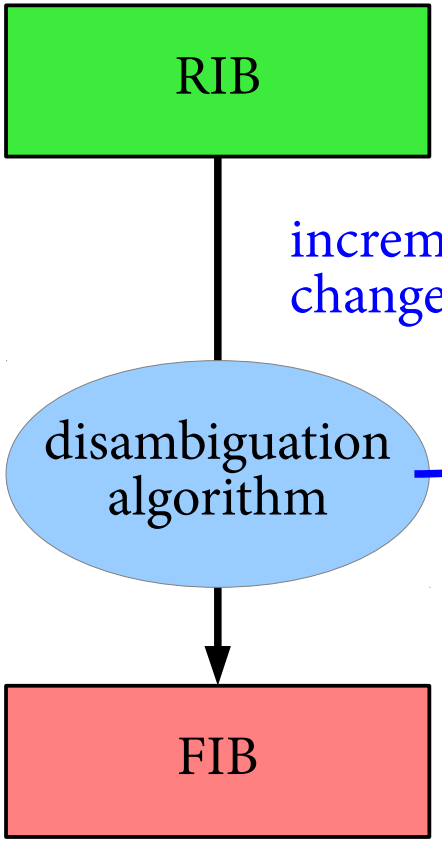
Need to choose the source address of outgoing packets.

Routing protocol and forwarding table

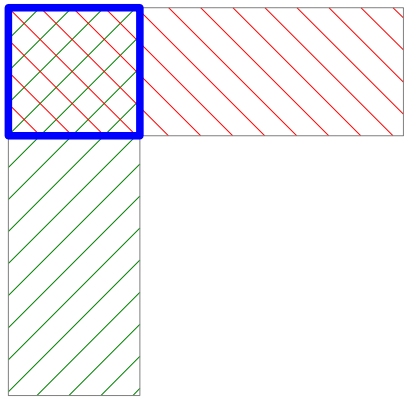


Disambiguation algorithm (idea)

destination first behaviour
(our choice)



Main idea: for each ambiguity, we maintain more specific entries (kernel only)



whatever behaviour
(forwarding plane choice)

Needed in IPv4 or

in IPv6 with
(not-so-)old Linux kernels.

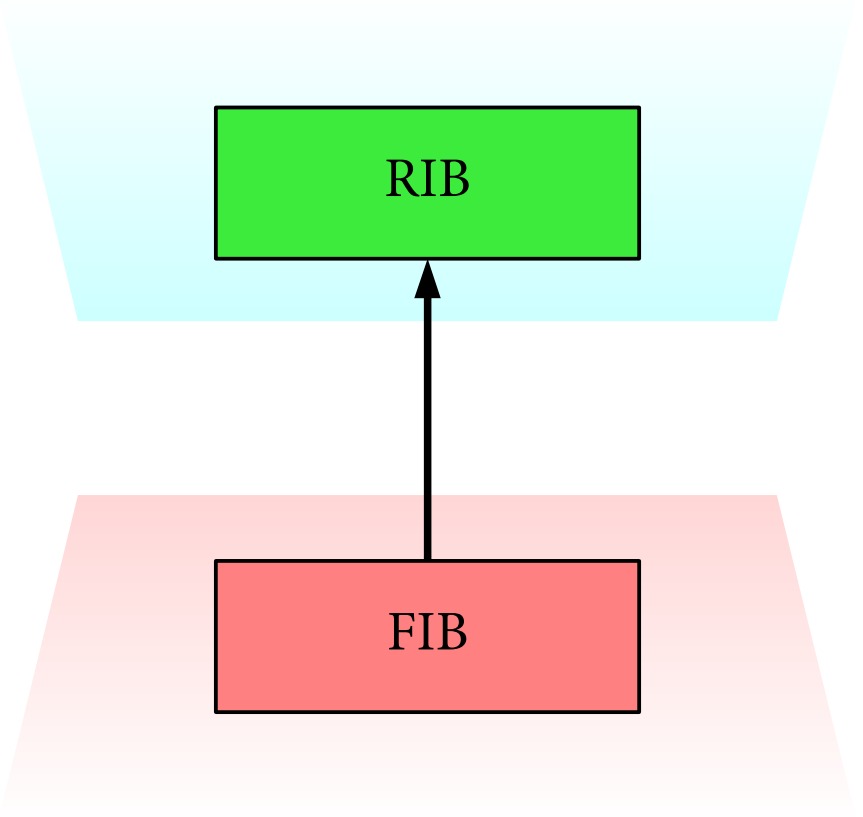
Redistribution in Babel

- Automatic redistribution of source-specific routes, using the native API:

  → *IPv6*
  → *recent kernels (> 3.11)*

- Explicit configuration (filters):

  → new action: `src-prefix`
  (only for redistribution)




```
redistribute […] src-prefix <prefix>

redistribute ip 0.0.0.0/0 eq 0 src-prefix 192.168.42.0/0
```

Entracte

Source-specific routing is good for multihomed networks.

We have seen how to interpret routing tables:
lexicographic order by destination first.

We have a backward compatible Babel extension, which computes source-specific routing tables.

We have a working source-specific extension of Babel.
→ *usable on "any" Linux kernel,*
→ *easily portable to support your favourite kernel,*
→ *Babel is The Only One able to deal with v4 and v6,*
→ *Active deployment in some homenet testbeds (opkg install hnet-full).*

Babel does source-specific routing (main branch).

Need to choose the source address of outgoing packets.

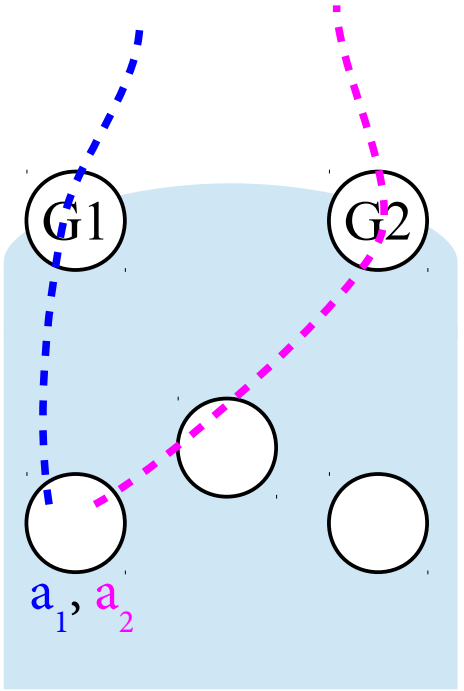
Address selection problem

There is a RFC for that:

- RFC 6724 (obsoletes 3484),
  → both source and destination address

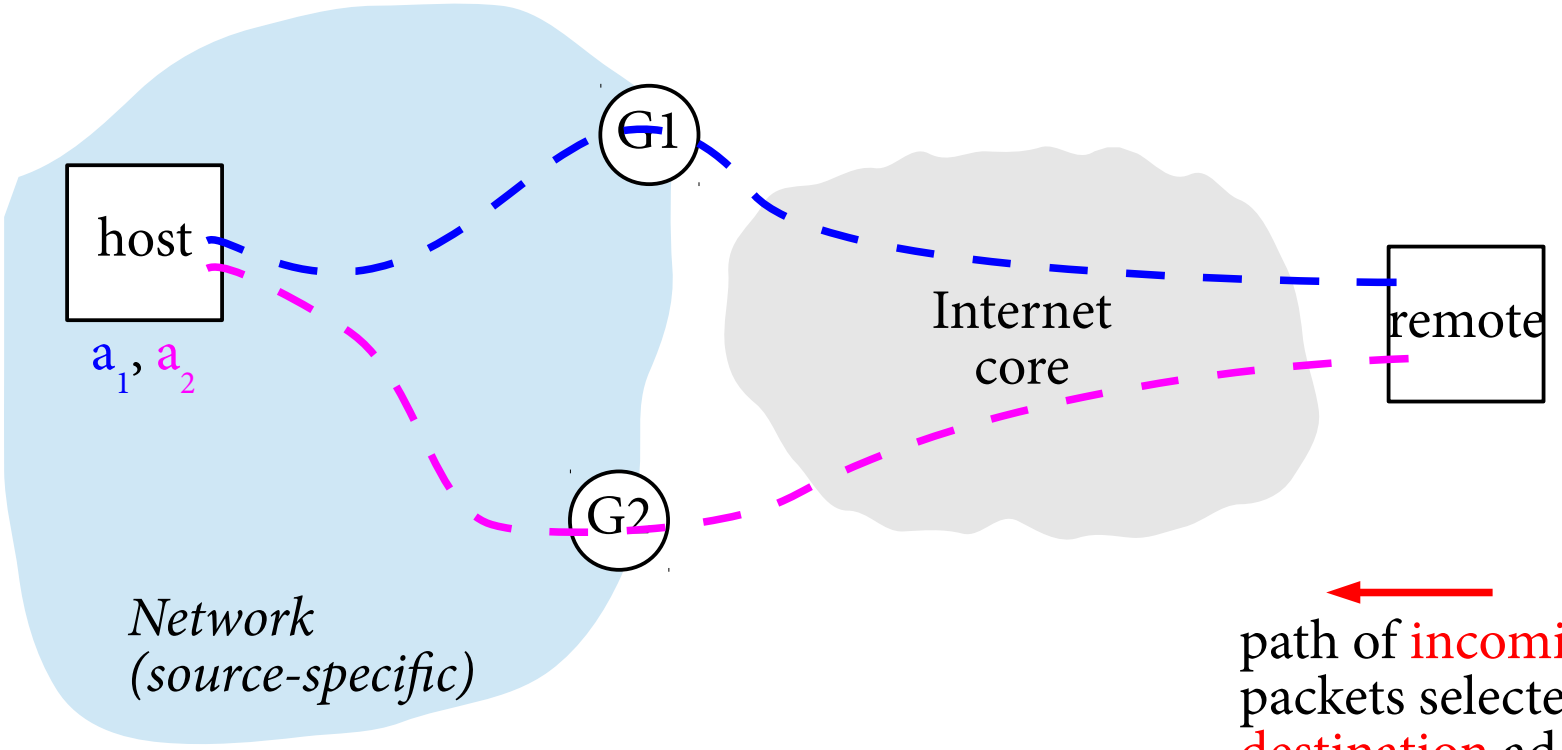
There has been some other works in this area:

- Happy eyeballs: Success with dual stack hosts,
  → try IPv4 & IPv6 simultaneously
- Shim6: but it's *more than that*,
- …



This is an open research area.
And it's broken in Linux for source-specific routes…

Don't choose address: choose paths!



path of incoming
packets selected by
destination address
(remote host choice)

path of outgoing
packets selected by
source address
(local host choice)

→ higher layers can do multipath:

- transport (MPTCP)
- application

Multipath TCP

- Compatible multipath replacement of TCP
  - provides reliability,
  - provides performances (load balancing).

→

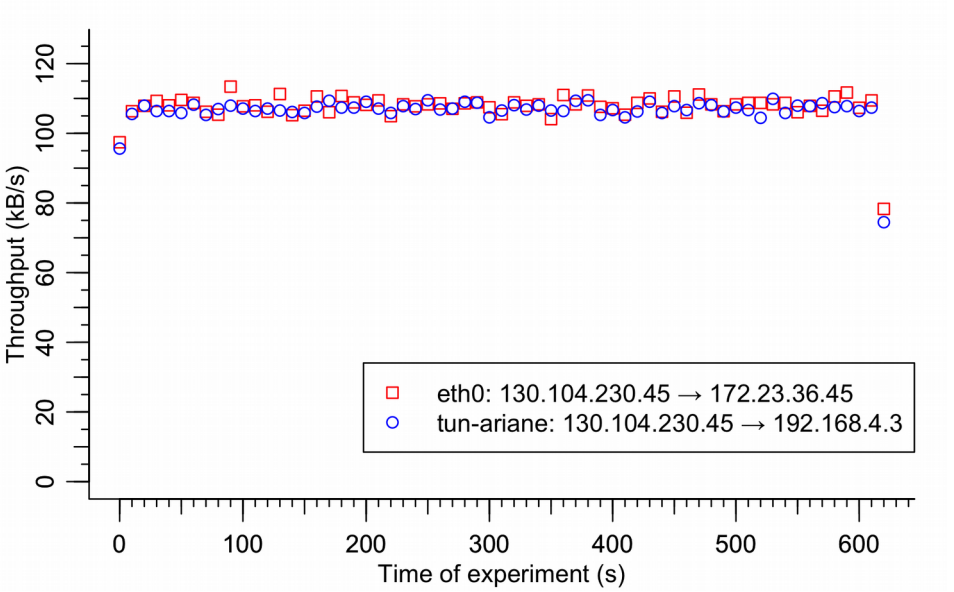
Application (7)
Transport (4)
Network (3)
…

Everything works out-of-the box
with source-specific routing:

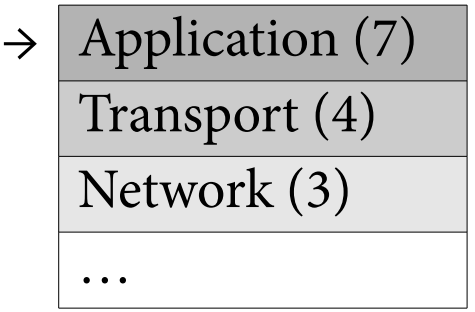
→ don't change the application,

→ just use TCP connections,

→ *(just rebuild your kernel…)*



Multipath at application layer



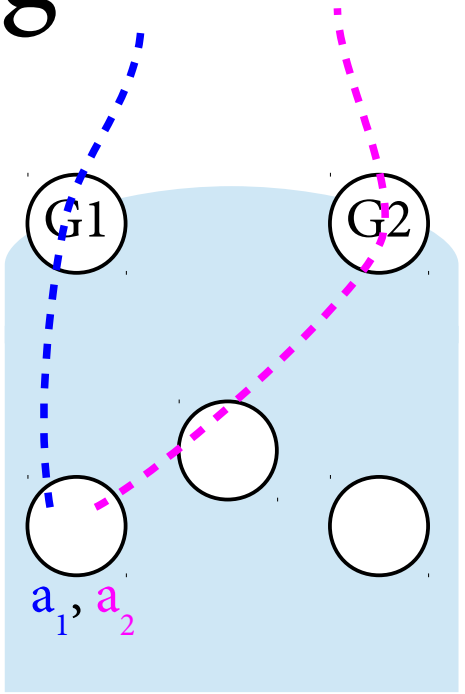
Advantages:

- More flexibility (think retransmissions),
- Keep control on the traffic sent,
- Be smarter: optimize delay, throughput,…
  (application dependant problem)
- *(don't need to rebuild your kernel!)*

Example: mp-mosh
(extends the mobile shell)

- probe paths,
- optimize RTT,
- may duplicate to minimize loss ratio.

Multihoming: Source-specific routing



Advantages:

- no configuration, just routing,
- keep TCP connections,
- may achieve reliability,
- may achieve best end-to-end performances.

Flaws:

- Need multipath protocols,
- Need to use a new routing protocol.

Conclusion

Source-specific routing is good for multihomed networks.

Babel is working production-quality source-specific protocol.
→ *usable on Linux, easily portable, does v4 and v6,*
→ *used in practice.*

Source-specific routing provides multipath opportunities:
Higher layers need mutlipath support:
→ use Multipath TCP,
→ let design new multipath applications