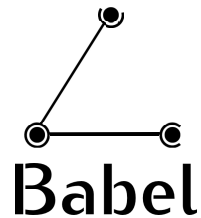
Babel
A flexible routing protocol

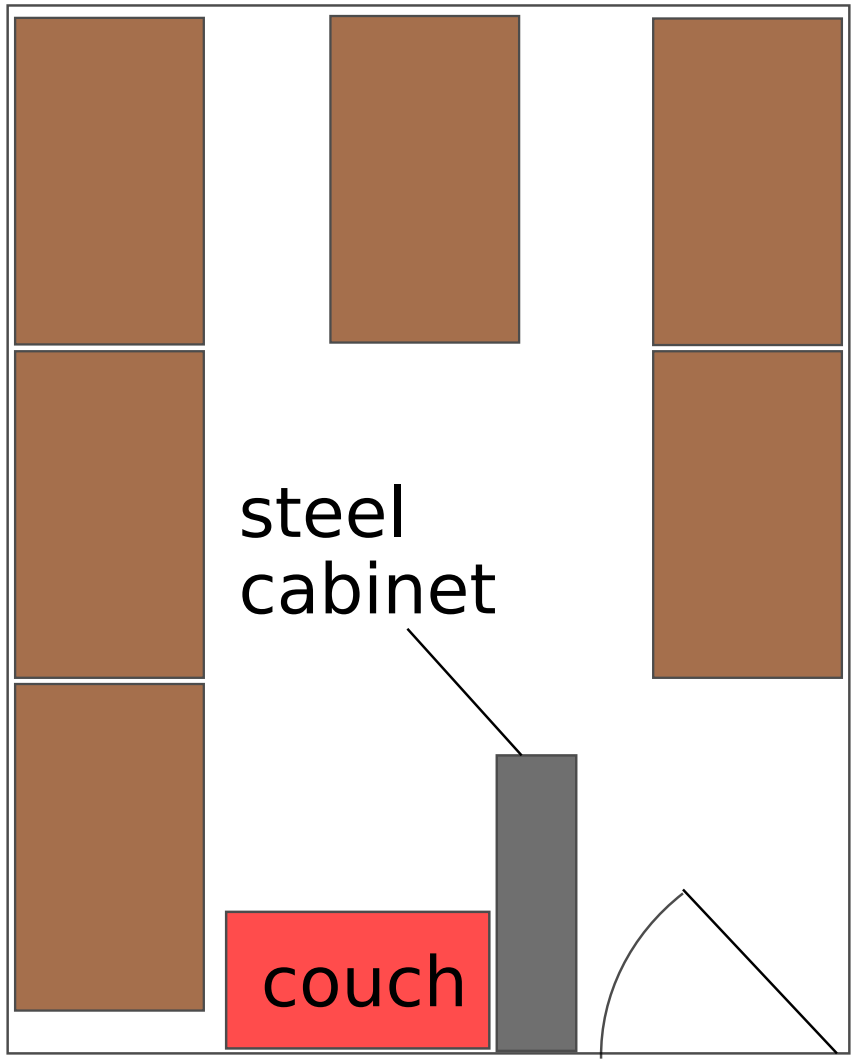
Juliusz Chroboczek
PPS
Université Paris-Diderot (Paris 7)

11 March 2014



The story

In December 2006, I started on a quest to bring wifi to the Ph.D. students' couch:



The story (2)

In December 2006, I started on a quest to bring wifi to the Ph.D. students' couch:

- I bought three home routers;
- reflashed them under OpenWRT;
- bought whisky, drank whisky with the network engineer, which got me an Ethernet jack (in the conference room) and an IPv6 prefix;
- installed OLSR;
- tried to set up an IPv6 OLSR mesh.

The story (3)

It did work, but not as well as I hoped:

- shortest hop is worst path routing;
- at the time, Unik-olsrd didn't clear transient routing loops fast enough;
- poor support for IPv6.

At the time, I didn't know enough about OLSR to fix the issues:

- use OLSR-ETX instead of RFC-compliant RFC;
- make Unik-olsrd's flooding more aggressive;
- fix OLSR for IPv6.

Instead of fixing OLSR, I designed Babel.

Babel

Babel is a modular protocol:

- a robust and mostly transient-free routing core;
- switchable metric computation;
- switchable route selection policies.

Currently, Babel has

- 4 different techniques for metric computation (different kinds of networks);
- a single route selection policy (good enough for all networks?).

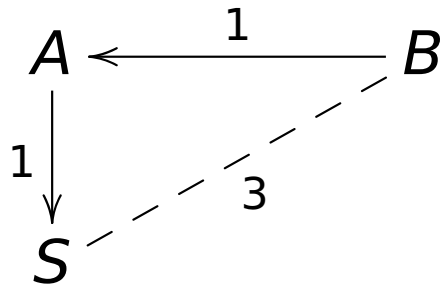
The Babel routing core

Babel is a loop-avoiding distance-vector protocol:

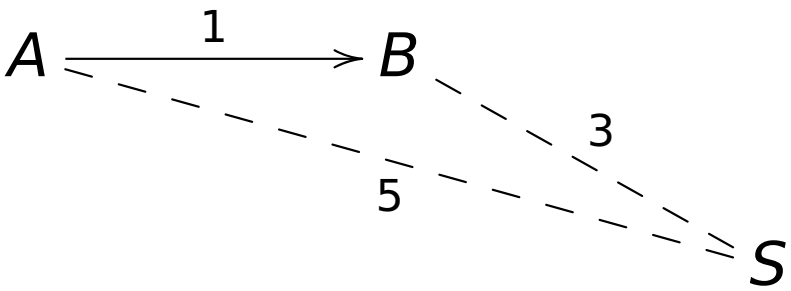
- uses distributed Bellman-Ford;
- an invariant guarantees loop-freedom: feasibility condition guarantees good transient behaviour.

Example of transient routing loop

Link-state protocol



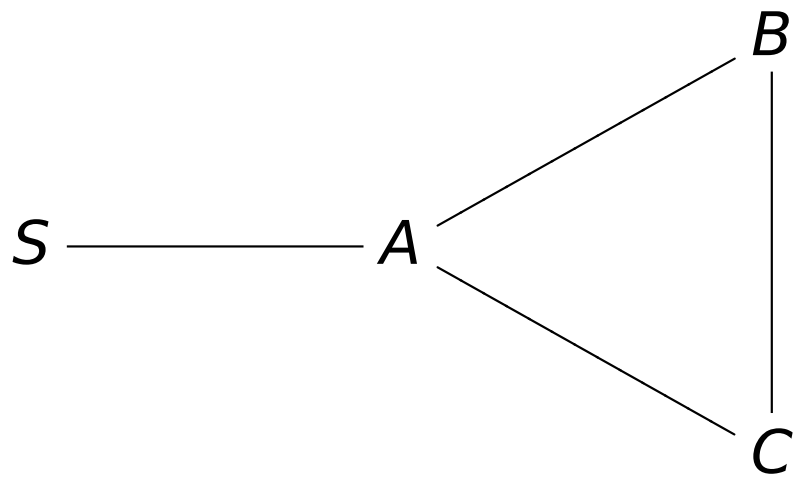
A uses the direct route to *S*
B goes through *A*



A switches to the route through *B* before *B* has switched to the direct route

This transient situation will persist until the topology change is successfully flooded to *B*.
With Babel, *A* will delay switching routes until it can be sure that *B* has switched to the direct route.

Distributed Bellman-Ford (1)



S	0	0	0	0
A	∞	1, nh $= S$	1, nh $= S$	1, nh $= S$
B	∞	∞	2, nh $= A$	2, nh $= A$
C	∞	∞	2, nh $= A$	2, nh $= A$

Converges in $O(\Delta)$.

Distributed Bellman-Ford (2)

Initially,

$$\mathrm{d}(S) = 0 \qquad \mathrm{d}(X) = \infty$$

Often enough, Y broadcasts $\mathrm{d}(Y)$ to its neighbours.
When X receives $\mathrm{d}(Y)$,

- if $\mathrm{nh}(X) = Y$,

$$\mathrm{d}(X) := c_{XY} + \mathrm{d}(Y)$$

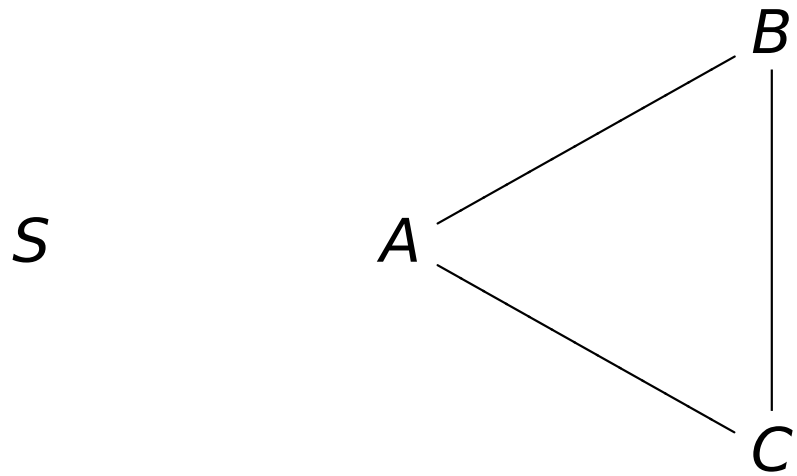
- si $c_{XY} + \mathrm{d}(Y) < \mathrm{d}(X)$

$$\mathrm{d}(X) := c_{XY} + \mathrm{d}(Y) \qquad \mathrm{nh}(X) := Y$$

Timeout: if $nh(X) = Y$, and Y stops broadcasting,

$$\mathrm{d}(X) := \infty \qquad \mathrm{nh}(X) := \bot$$

Distributed BF: counting to infinity



A	1, nh $= S$	3, nh $= B$	3, nh $= B$	3, nh $= B$
B	2, nh $= A$	2, nh $= A$	3, nh $= C$	3, nh $= C$
C	2, nh $= A$	2, nh $= A$	2, nh $= A$	4, nh $= A$

Converges in $O(\infty)$. (RIP: $\infty = 16$.)

Before convergence, there is a routing loop.

« Good news travel fast, bad news travel forever. »

BF: Feasibility conditions

BF is robust, we can ignore updates if they risk generating a loop.

When X receives $(\mathrm{d}(Y), f)$,

- if $\mathrm{nh}(X) = Y$ and $\mathrm{feasible}(Y, \mathrm{d}(Y), f)$

$$\mathrm{d}(X) := c_{XY} + \mathrm{d}(Y)$$

- if $c_{XY} + \mathrm{d}(Y) < \mathrm{d}(X)$ and $\mathrm{feasible}(Y, \mathrm{d}(Y), f)$

$$\begin{aligned} \mathrm{d}(X) &:= c_{XY} + \mathrm{d}(Y) \\ \mathrm{nh}(X) &:= Y \end{aligned}$$

where feasible is a function that guarantees the lack of loops.

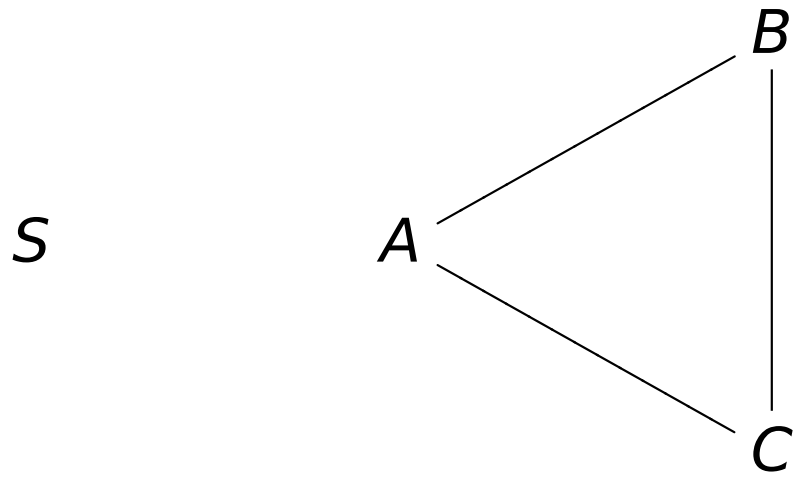
Feasibility conditions

BGP, *Path Vector*:
f is the complete path,
$\text{feasible}(f) = \text{self} \notin f$.

DSDV, **AODV**:
$\mathit{feasible}(d) \equiv c + d \leq \text{d(self)}$
Invariants: $\text{d}(X) \searrow$ and if $A \leftarrow B$ then $\text{d}(A) < \text{d}(B)$.

EIGRP/DUAL, **Babel**:
We maintain $\text{fd}(X) = \min_{t \leq \text{now}} \text{d}(X, t)$.
$\mathit{feasible}(d) \equiv d < \text{fd(self)}$
Invariants: $\text{fd}(X) \searrow$ and if $A \leftarrow B$ then $\text{fd}(A) < \text{fd}(B)$.

Feasibility: exemple

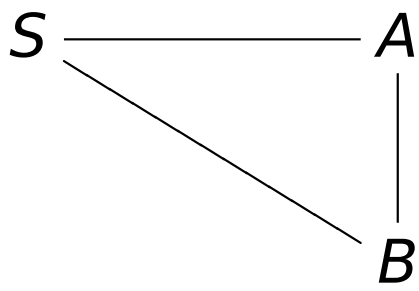


A	1, $\mathrm{fd} = 1$	∞, $\mathrm{fd} = 1$	∞, $\mathrm{fd} = 1$	∞, $\mathrm{fd} = 1$
B	2, $\mathrm{fd} = 2$	2, $\mathrm{fd} = 2$	∞, $\mathrm{fd} = 2$	∞, $\mathrm{fd} = 2$
C	2, $\mathrm{fd} = 2$	2, $\mathrm{fd} = 2$	∞, $\mathrm{fd} = 2$	∞, $\mathrm{fd} = 2$

Converges in $O(\Delta)$.

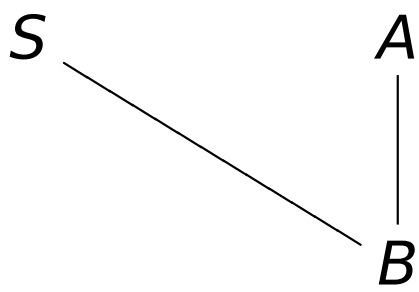
Feasibility: starvation

The feasibility conditions (1) et (2)
cause starvation.



$\mathrm{d}(A) = 1, \mathrm{fd}(A) = 1$

$\mathrm{d}(B) = 1, \mathrm{fd}(B) = 1$



$\mathrm{fd}(A) = 1$

$\mathrm{d}(B) = 1$

The only available route is not feasible.

Solving starvation

Idea: when no route is available, reboot the whole network.

DUAL/EIGRP makes a
global synchronisation (of routes towards S).

DSDV, AODV and Babel use sequenced routes.

Solving starvation: sequenced routes

Route announcements are equipped with a sequence number:

$$(s, \mathrm{d}(B))$$

where $s \in \mathbf{N}$ is incremented by the source:

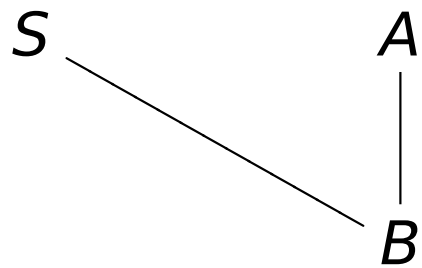
$$\begin{aligned} \mathrm{d}(S) &= (s, 0) \qquad (s \nearrow) \\ c + (s, m) &= (s, c + m) \end{aligned}$$

Define

$$(s, m) \leq (s', m') \quad \text{when} \quad \begin{array}{l} s > s' \text{ ou} \\ s = s' \text{ et } m \leq m' \end{array}$$

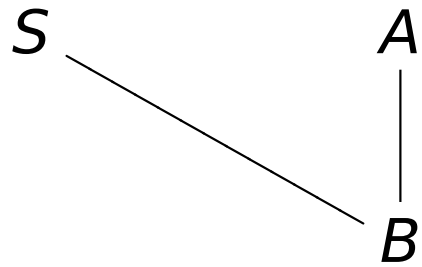
$$\mathrm{feasible}(s, m) \equiv (s, m) < \mathrm{fd}.$$

Sequenced routes: example



S	$(1,0)$	$(2,0)$	$(2,0)$
A	$\infty, \mathrm{fd} = (1,1)$	$\infty, \mathrm{fd} = (1,1)$	$(2,2), \mathrm{fd} = (2,2)$
B	$(1,1), \mathrm{fd} = (1,1)$	$(2,1), \mathrm{fd} = (2,1)$	$(2,1), \mathrm{fd} = (2,1)$

Temporary starvation

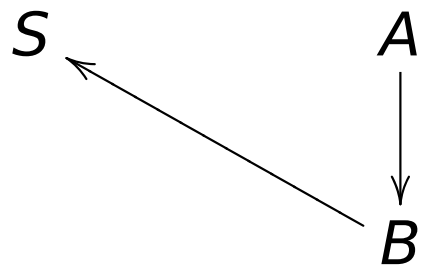


$$\begin{aligned} \mathrm{d}(S) &= (1,0) \\ \mathrm{d}(B) &= (1,1) \\ \mathrm{d}(A) &= \infty \qquad \mathrm{fd}(A) = (1,1) \end{aligned}$$

A must wait until S generates a new seqno and the network propagates it.
In Babel, temporary starvation is explicity signalled by A ($\neq$ DSDV).

Solving temporary starvation

When a Babel node suffers from temporary starvation (routes available but not feasible) it sends an explicit request for a new seqno.



Unlike AODV, this request is not broadcast, which avoids an increasing horizon search, a simple *hop count* is enough.

Multiple gateways

In general, we want it to be possible to have multiple nodes that announce the same prefix without synchronising sequence numbers.

Babel distinguishes source and destination.

A Babel announce contains a triple

$$(s, d, id)$$

where *id* uniquely identifies the node originating the route. Routes are indexed by source and destination.

Multiple gateways: loops

In the presence of multiple gateways,
Babel no longer guarantees loop-freedom.

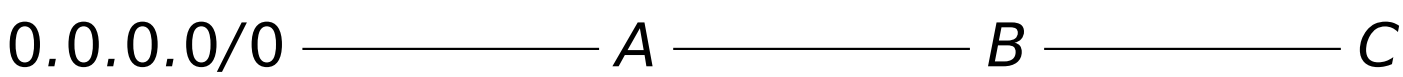
$$S_1 \text{ ——— } A \text{ ——— } B \text{ ——— } S_2$$

$$\mathrm{d}(A) = (17, 1) \qquad \mathrm{d}(B) = (43, 1)$$
$$\mathrm{fd}(A, S_1) = (17, 1) \qquad \mathrm{fd}(B, S_2) = (43, 1)$$

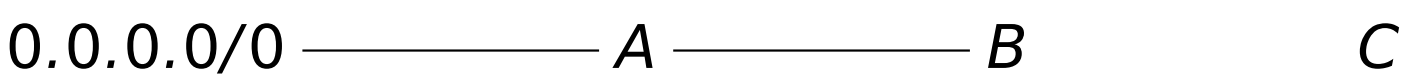
We guarantee that a loop disappears in $O(n)$, where n is the size of the loop.

Non-disjoint routes

A routing loop can also occur because of two routes towards overlapping prefixes.



The link between B and C disappears:

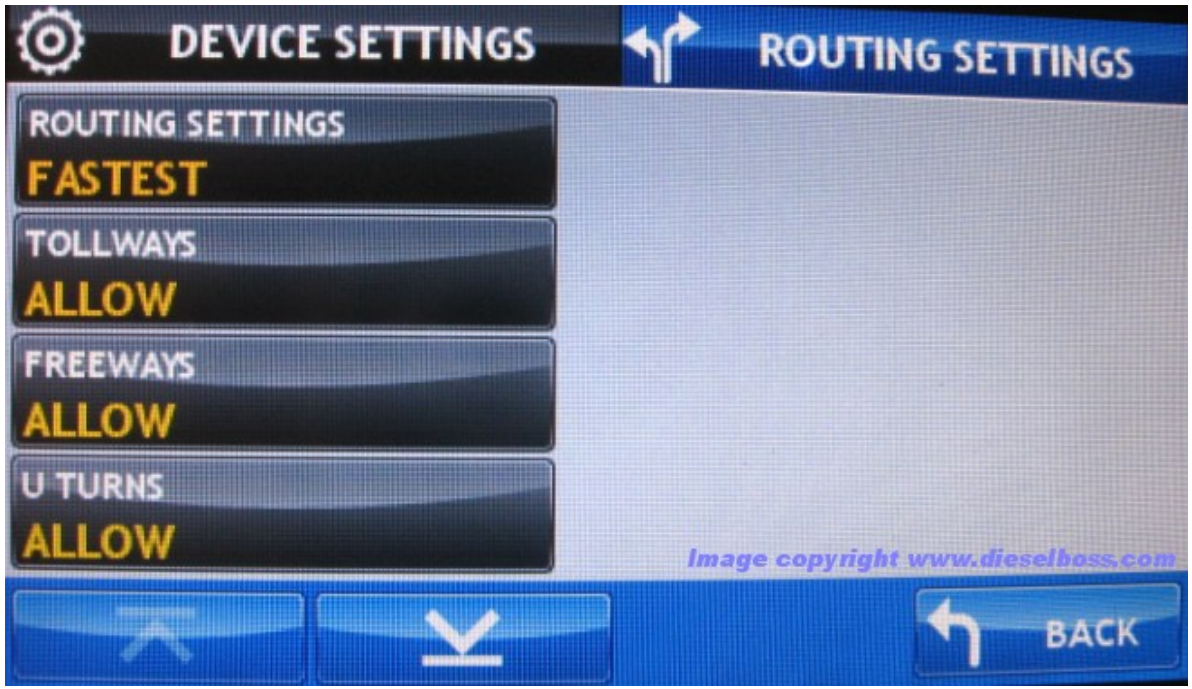


If B reroutes through A, there is a temporary routing loop. This can only happen after a retraction.

Babel obeys a hold time after a retraction, only applies to shorter prefixes ($\neq$ RIP).
This prevents automatic summarisation.

Metrics

On a GPS, you select the function to optimise:



The function to minimise is called the metric:

- distance: shortest path;
- time: fastest path;
- monetary cost: cheapest path;
- etc.

Neighbour sensing

A Babel node broadcasts sort-of-periodically

$$\text{Hello}(\text{seqno}, \text{interval}, \cdots)$$

a Hello message with a seqno, a bound on the time before the next Hello, and random additional data. For each neighbour B, a node broadcasts

$$\text{IHU}(B, \text{rate}, \cdots)$$

an I Heard You message with the number of recently received Hellos from B, and random additional data.

Metrics

Babel is metric-agnostic. According to RFC 6126,

- a metric MUST be strictly monotonic:

$$m < c \oplus m;$$

- a metric SHOULD be isotonic:

$$\text{if } m \leq m' \text{ then } c \oplus m \leq c \oplus m'$$

Strict monotonicity is enough to guarantee that Babel will converge to a loop-free Nash equilibrium. Isotonicity ensures that this equilibrium is actually the tree of shortest paths.

By default, Babel uses:

- hop-count with 2-out-of-3 sensing on wired links;
- ETX (packet loss) on wireless links.

But we can do better.

Metrics: radio-interference

Babel-Z3 for wireless meshes

The Z3 metric refines ETX by taking radio interference into account:

$$
\begin{aligned}
M(l \cdot r) &= C(l) + M(r) && \text{if } l \text{ and } r \text{ interfere} \\
M(l \cdot r) &= \frac{1}{2} C(l) + M(r) && \text{otherwise}
\end{aligned}
$$

This metric is not isotonic:

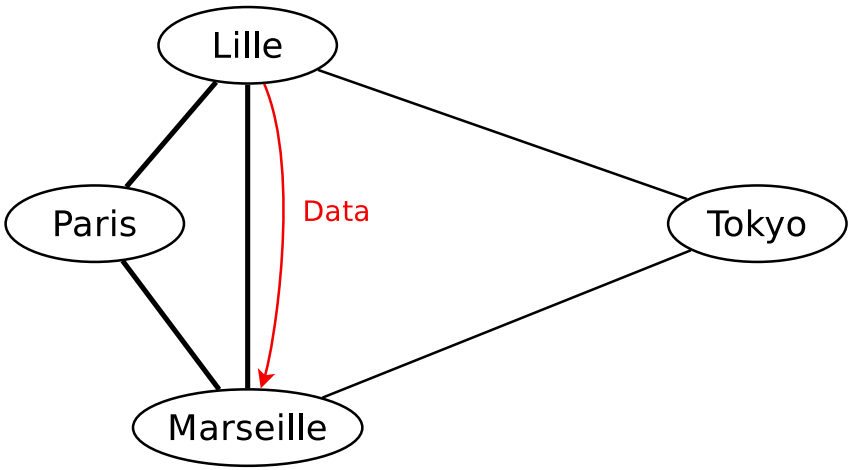
$$A \xrightarrow{1} B \xrightarrow[1.2]{1} C$$

It appears to work fine in practice, but it hasn't been evaluated formally: results difficult to reproduce.

Metrics: delay

Babel-RTT for Robust Overlay Networks

Nexedi have been using Babel to route in a distributed cloud. Babel requires no configuration.



Hop-count routing has a tendency to route through Tokyo.

Idea: use delay as a component of a routing metric. This causes a feedback loop, which can cause oscillations. We limit oscillations using a combination of three techniques:

- smoothing of the link cost;
- saturation of the link cost;
- time-sensitive route selection.

Route selection

Route selection: choose the best route among those available.

Goals:

- choose the route with smallest metric;
- prefer stable routes.

These are contradictory goals.

Initially, Babel was overly sensitive to short-term metric variations. Over the years, Babel's route selection policy accumulated increasing amounts of kludges to make it more sticky.

In early 2013, all of this has been scrapped, and Babel has a new route selection algorithm.

History-sensitive route selection

Hysteresis

For each route, we maintain:

- the announced metric M;
- the smoothed metric M_s.

M_s is continuous, and converges exponentially towards M:

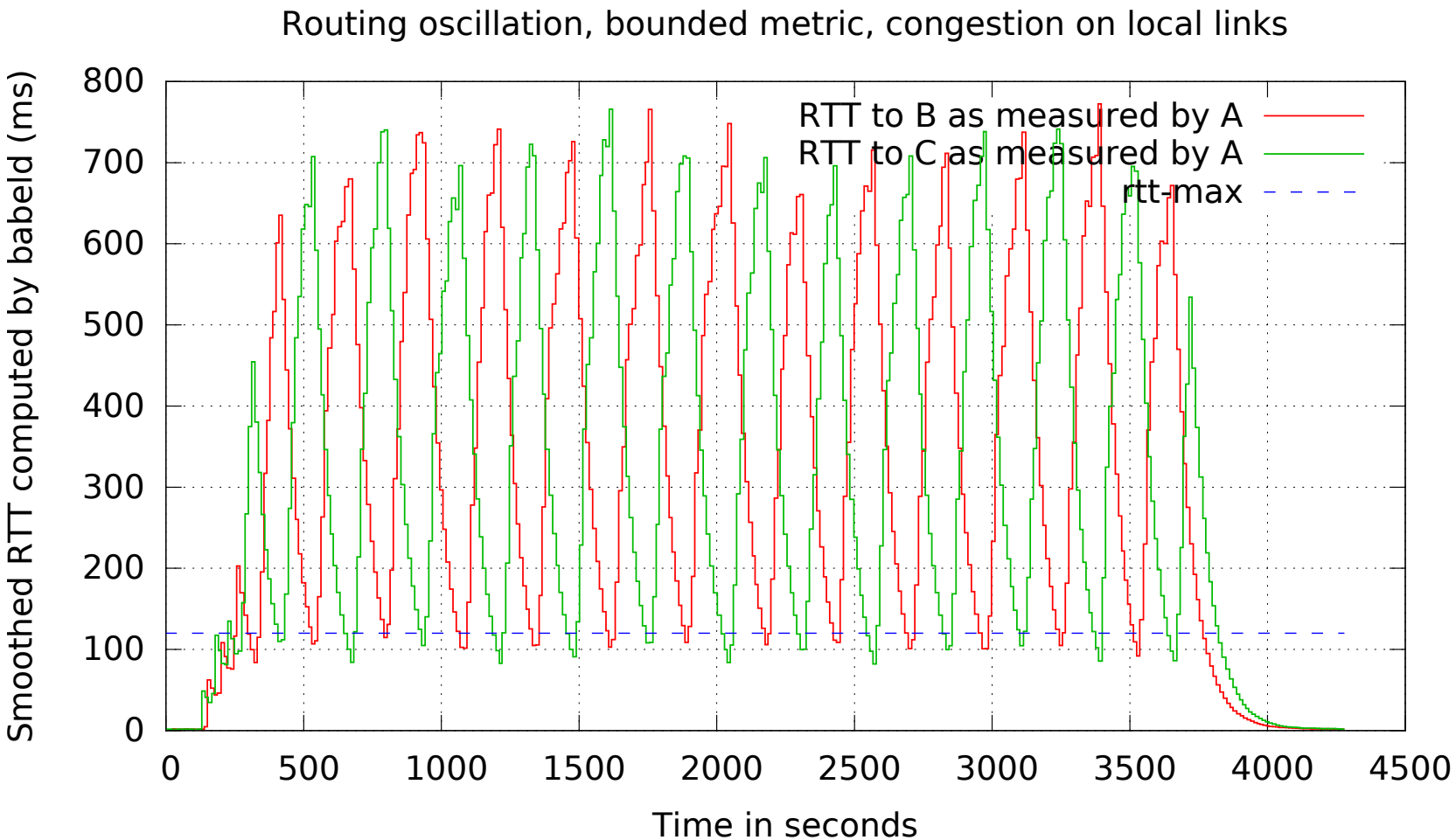
$$M_s := \beta(\delta) \cdot M_s + (1 - \beta(\delta)) \cdot M_a$$

with $\beta(\delta)$ chosen so that the time constant is 4 s.
We switch routes:

- when the current route is retracted ($M = \infty$);
- when both metrics are better ($M' < M$ and $M'_s < M_s$).

In effect, we do converge to the tree of shortest paths, but take our time switching routes unless we lose our current route. This is a form of hysteresis.

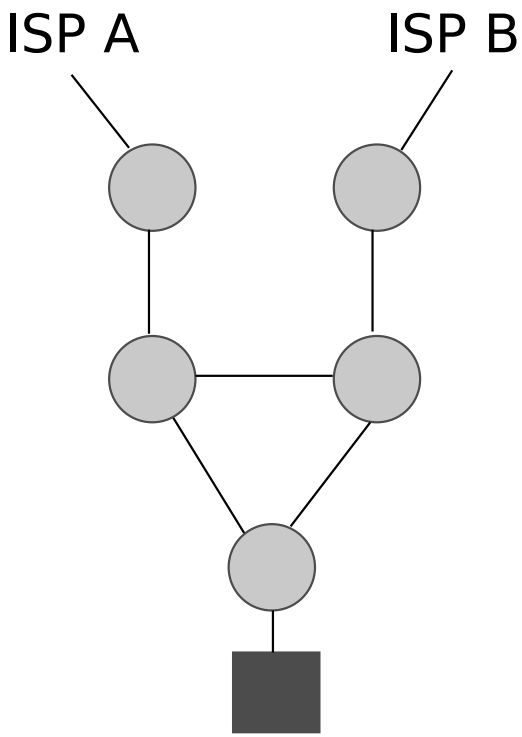
Limited oscillations



Source-sensitive routing

Source-sensitive routing is a modest extension to next-hop routing with wide-ranging consequences.

A packet is routed according to both its source and its destination. The routing table is indexed by destination-source pairs.



Provides a cheap form of multihoming with hostile ISPs. Motivated by the IETF Homenet working group. Works great with MPTCP.

Source-sensitive routing (2)

Source-sensitive routing raises a number of difficult challenges:

- the routing table is not totally ordered: need to define a routing policy;
- not necessarily implemented by the lower layers: complex disambiguation algorithm.

First complete implementation: Matthieu Boutier 2013.
IETF work ongoing.

Conclusions

Babel is a robust and flexible routing protocol:

- reasonable on wired networks;
- good on wireless meshes;
- great framework for experimenting with new ideas:
  - radio interference-sensitive metrics;
  - delay-based routing;
  - source-specific routing.

Having a production-quality implementation that you control and can freely modify is costly, but provides great opportunities for collaboration.