Let’s revive Babel-RTT

Juliusz Chroboczek
IRIF
Université de Paris

26 July 2021

Babel extensions

Babel extensions are either:

- in the process of being standardised (source-specific, Babel-MAC);
- or not used in production (radio diversity, ToS-specific).

One exception: Babel-RTT:

- used in production;
- only described in:
  - an expired IETF draft;
  - a rejected paper (not a very good one).

History

History:

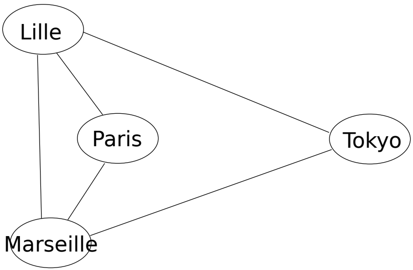
- Nexedi described the problem, early 2014;
- solution designed in 2014;
- implemented and written up in collaboration with Baptiste Jonglez, summer 2014;
- deployed in production by Nexedi, autumn 2014;
- presented to this WG, 28 March 2019;
- continuously deployed in production for 7 years!

Described in:

- draft-ietf-babel-rtt-extension-00 (October 2019);
- `https://arxiv.org/abs/1403.3488`.

Problem statement

Nexedi have been running a global overlay network between datacenters:



What happens when the Lille-Marseille link is down?

In 1/2 of the cases, unextended Babel chose to reroute the traffic through Tokyo.

Nexedi were not happy.

Solution: use RTT

In 1/2 of the cases, unextended Babel chooses to reroute the traffic through Tokyo.
That's not good.

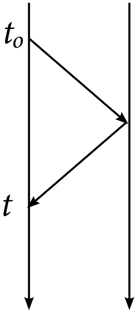
Initial suggestion: a GPS in every data center.
That's reportedly not practical.

Idea: measure RTT (two-way delay) and derive a metric from that. But

- the natural way to measure RTT requires asymmetric, synchronous interaction; Babel is a symmetric, asynchronous protocol;
- using RTT as input to a routing metric causes a (negative) feedback loop, which may lead to oscillations.

Measuring RTT (1)

The naive algorithm



The natural way to measure RTT is asymmetric and synchronous.

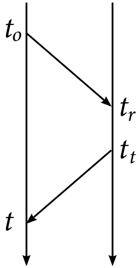
Client says “ping!”.
Server replies “pong!” as fast as possible.

RTT = $t - t_o$.

Babel is a symmetric, asynchronous algorithm.
The naive “ping” algorithm is a poor fit for Babel.

Measuring RTT (2)

Mills' algorithm



Mills' algorithm, used in HELLO and NTP.

The remote peer sends a packet with:

- t_o, the origin timestamp;
- t_r, the reference timestamp;
- t_t, the transmit timestamp.

RTT = $(t - t_o) - (t_t - t_r)$.

This is a symmetric, asynchronous algorithm that doesn't require clocks to be synchronised.

Its accuracy depends on:

- t_t computed as late as possible before transmission;
- t computed as early as possible after reception;
- clock drift negligible during a packet exchange.

Adapting Mills' algorithm in Babel

Babel uses multicast and unicast packets.

- Transmission timestamp t_t conceptually multicast, stored in Hello TLV;
- origin and reference timestamp unicast, stored in IHU TLV.

Granularity of timestamp is $1\,\mu s$.
(Originally 10 ms, but Dave complained.)

Packet format

Timestamp in Hello:

```
 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 = 3    |     Length    |      Transmit timestamp       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

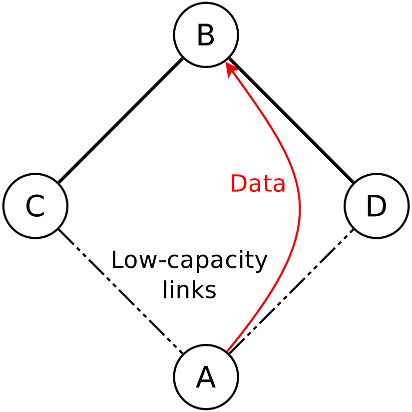
Timestamp in IHU:

```
 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 = 3    |     Length    |        Origin timestamp       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |        Receive timestamp      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Should we be using distinct types?

Oscillations

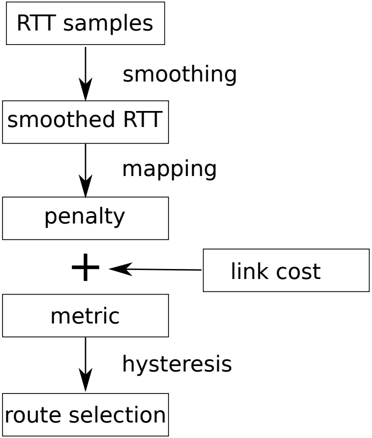
Using RTT as a routing metric leads to oscillations



In principle, Babel doesn't care. However, oscillations may lead to packet reordering, which harms higher layer protocols.

From RTT to route selection

Babeld uses a complex process to map RTT to values usable in route selection.



Mills' algorithm yields RTT samples.

Our goal is route selection.

The RTT samples are processed in order to minimise:

- noisy signal;
- oscillations

Conclusion

Babel-RTT is the only widely-deployed Babel extension that is not being standardised.

Reasons:

- simple algorithm, but difficult to make it work well;
- lack of a theoretical understanding.

I intend to revive draft-ietf-babel-rtt-extension for publication as an Experimental RFC.

Please object now! Please review!