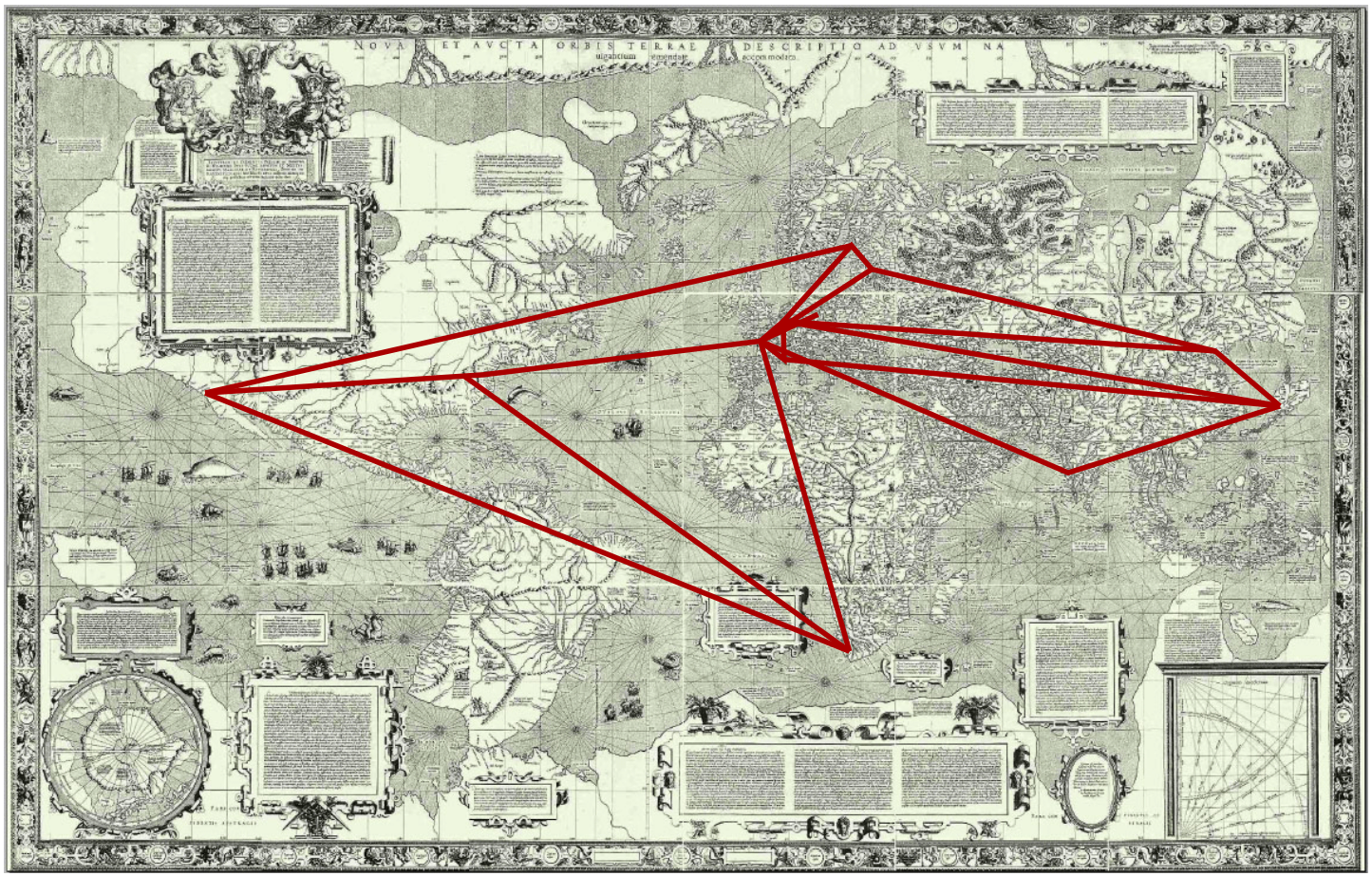
Re6st and Babel
Resilient distributed cloud

Juliusz Chroboczek
IRIF
Université de Paris

1 July 2021

Distributed cloud

2010: J.-P. Smets tells me about the distributed cloud.

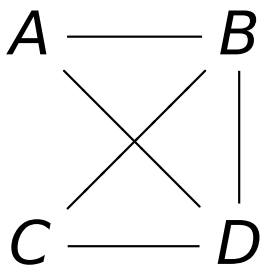


Important take-away: requires convex connectivity.

Reliability issues

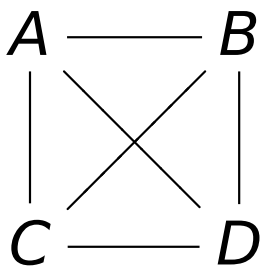
Connexity vs. convexity

The internet remains (generally) connected.



Connected: for all X, Y,
there exists a path from X to Y.

But it is seldom convex.



Convex: for all X, Y,
there is a direct link from X to Y.

Cloud algorithms require convexity.

Reliability issues are unavoidable

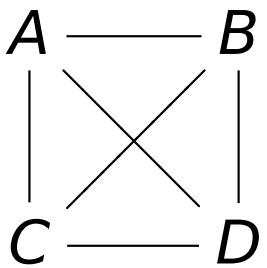
Quadratic behaviour

As the number of nodes in a network grows, the number of links grows as

$$\frac{n(n-1)}{2} \approx \frac{n^2}{2}.$$

Nodes	Links
10	45
100	4 950
1 000	499 500
10 000	49 995 000

As the number of nodes grows, failure of at least some links becomes unavoidable.



It’s not the fault of the network providers.
No matter how reliable your provider,
in a large enough network some links will fail.

The re6st software stack

Re6st builds a resilient convex overlay.

Three layers :

- tunnelling software (GRE, OpenVPN, IPsec, etc.) ;
- re6st proper, builds enough tunnels for the network to be connected ;
- Babel dynamic routing protocol, finds paths across the network to make it convex.

Loose coupling between the three :

- easier to understand and debug;
- can use multiple off-the-shelf tunnelling protocols (works around broken firewalls);
- can use off-the-shelf routing protocol (routing protocols are hard, let’s go shopping).

Tunnelling protocol

A tunnelling protocol establishes network links across the Internet.

Re6st is protocol-agnostic, can use OpenVPN, GRE, IPsec, etc.:

- uses OpenVPN by default;
- can use different technologies when OpenVPN doesn’t work (misconfigured firewalls);
- can use whatever technology is efficient or fashionable.

Re6st proper: tunnel establishment

Re6st proper establishes tunnels.

Very simple algorithm:

```
while true {
    if num_neighbours < 20 {
        p := random_node()
        establish_new_tunnel(p)
    }
}
```

Tends towards a 20-regular random(ish) graph.

This graph is very well connected with very high probability (Bollobás, 2001).

(Thanks to F. de Montgolfier.)

Dynamic routing: Babel

Babel is a dynamic routing protocol, finds best paths in a network:

- fully documented (IETF standard);
- small implementation, easy to modify;
- flexible metrics;
- extensible.

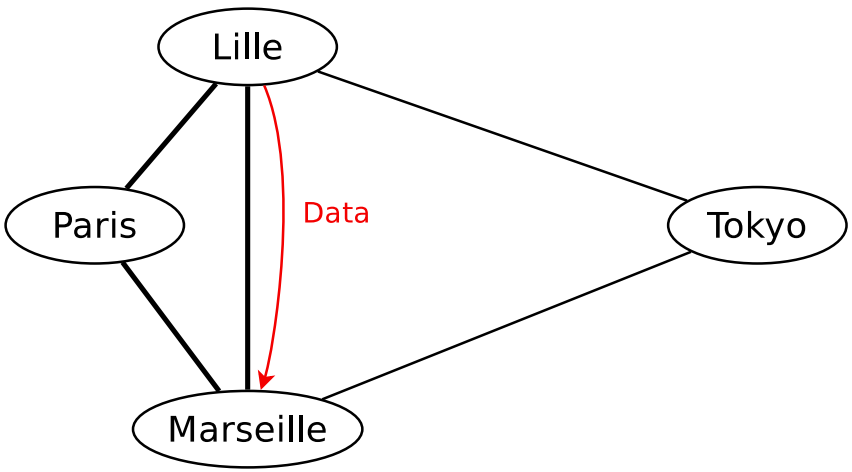
(In principle, re6st could work with a different routing protocol, e.g. OSPF, although we'd lose these nice properties.)

Problem: suboptimal routing

Off-the-shelf Babel uses

- shortest routes on wired links;
- minimal loss routes on wireless links.

Nexedi's network consists of wide area lossless tunnels.



If the link between Lille and Marseilles is lost, standard Babel will route through Tokyo in 50% of cases.

We need a way to distinguish Tokyo from Paris.

- No manual configuration;
- no extra hardware (GPS, etc.).

Solution: latency-based routing

Babel is an extensible protocol with flexible metric computation: it is possible to write an extension that uses a different metric (without impairing interoperability).

Solution: measure latency (RTT):

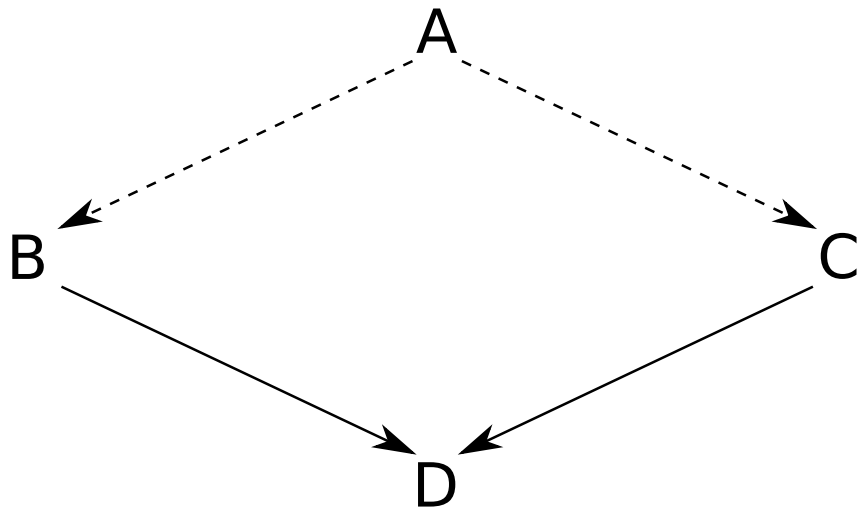
- Lille-Paris has RTT = 12 ms ;
- Lille-Tokyo has RTT = 300 ms.

We use Mills's algorithm (with smoothing) :

- few changes to Babel (asynchronous algorithm);
- no manual configuration (geographic position, etc.);
- no extra hardware (GPS, atomic clock, etc.);
- no need for a real-time OS.

Latency-based routing: stability issues

When two parallel routes are available, latency-based routing tends to oscillate between the two.



- Traffic follows the A-B-D route;
- latency of A-B-D increases;
- traffic switches to A-C-D;
- latency of A-C-D increases…

The solution is to apply a non-linear map to latencies, and to apply a fair amount of hysteresis.

Oscillations:

- every 10 minutes in the lab;
- in practice, never!

Conclusion

Resilient overlay for distributed cloud computing.

Three loosely coupled components:

- tunnelling protocols;
- re6st proper, randomly builds connected network;
- Babel: dynamic routing protocol:
  - makes the network convex (full reachability);
  - latency-based routing minimises latency.

`https://re6st.nexedi.com/`
`https://www.irif.fr/~jch/software/babel/`

Partnership between Academia and small business.