# Koala protocol

## @ Discovery Midterm Review

Genc Tato

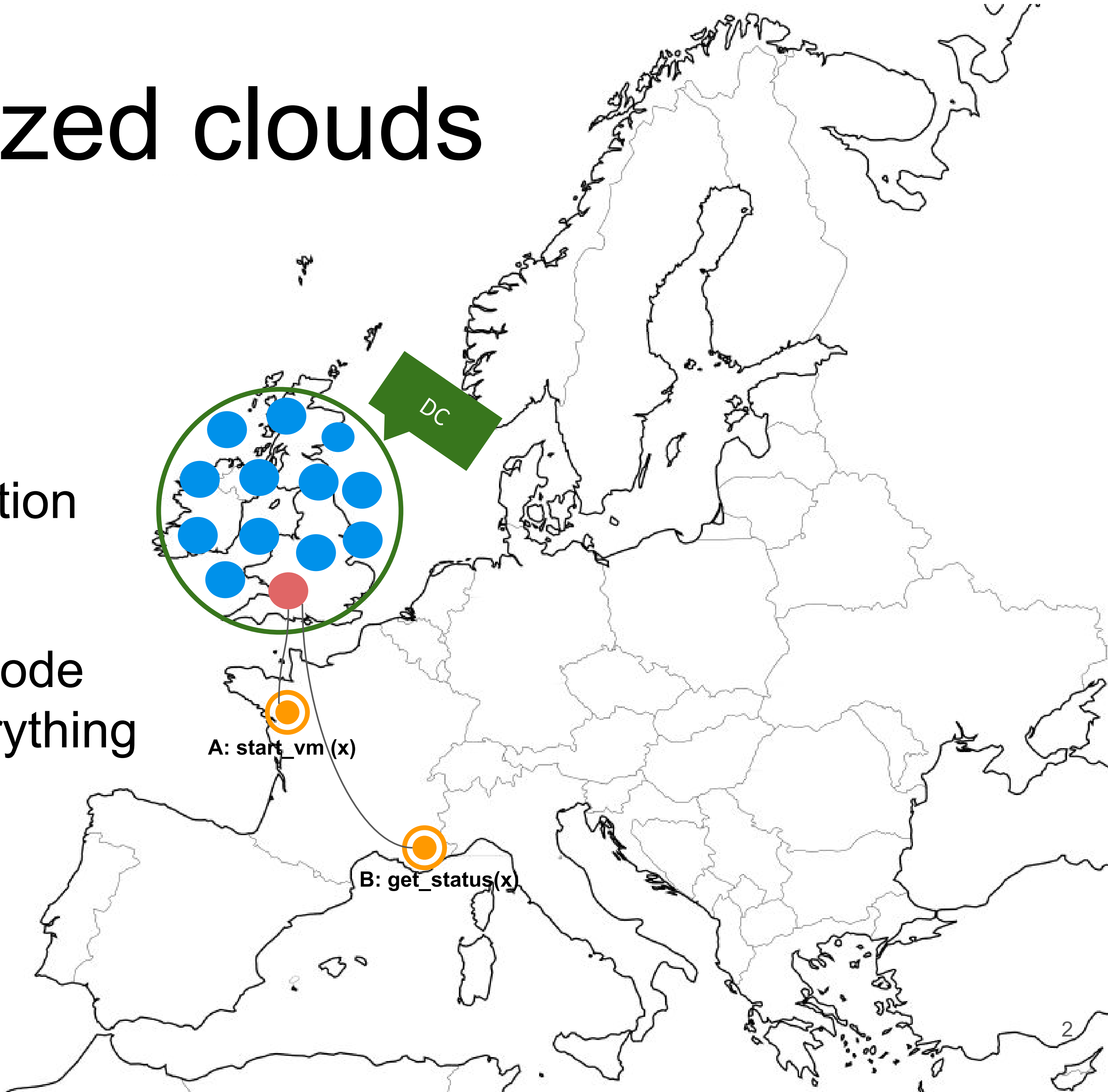
Supervised by:

- Cédric Tedeschi
- Marin Bertier

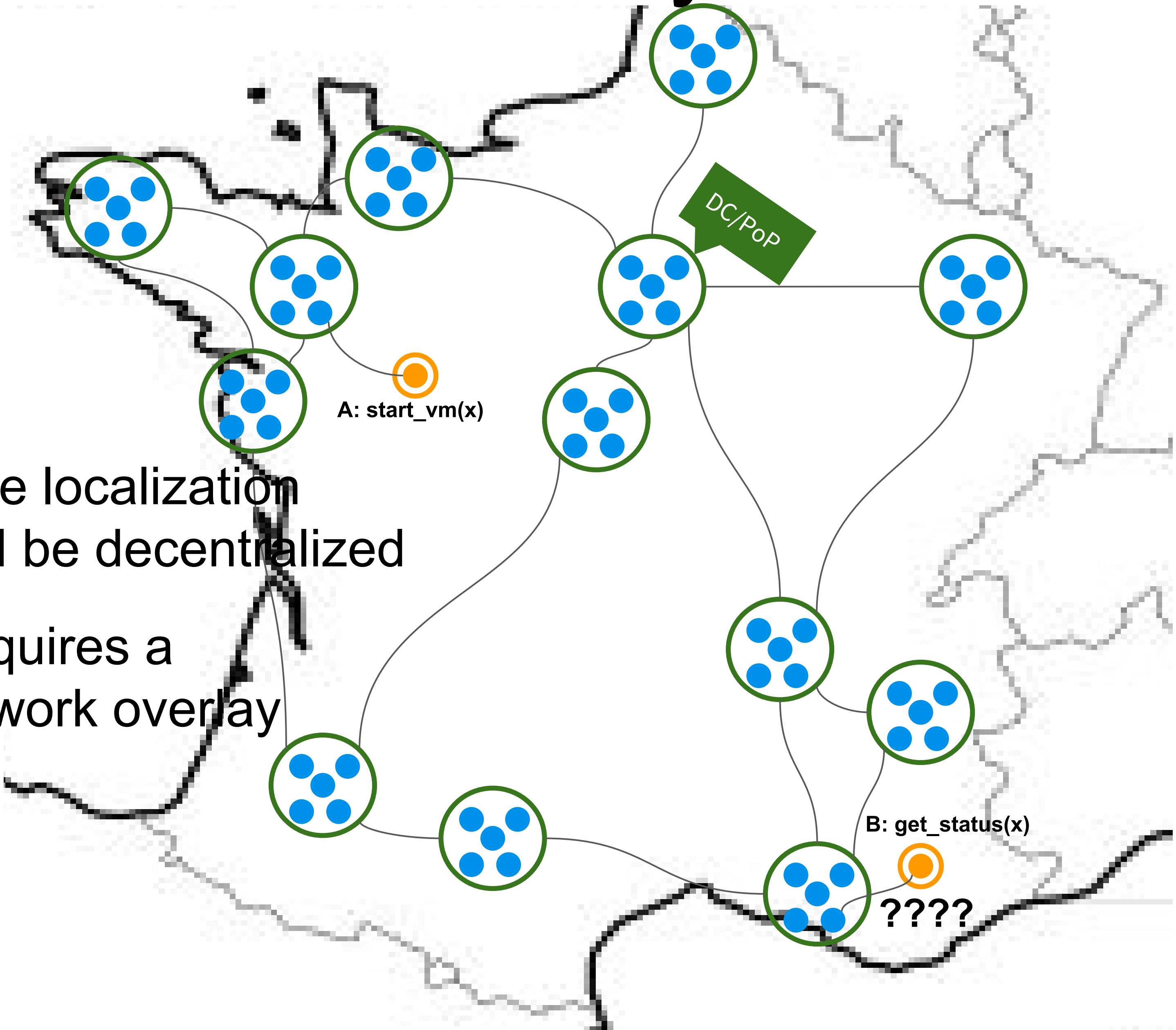
# Centralized clouds

Service localization is centralized

- Controller node knows everything



# Discovery Cloud



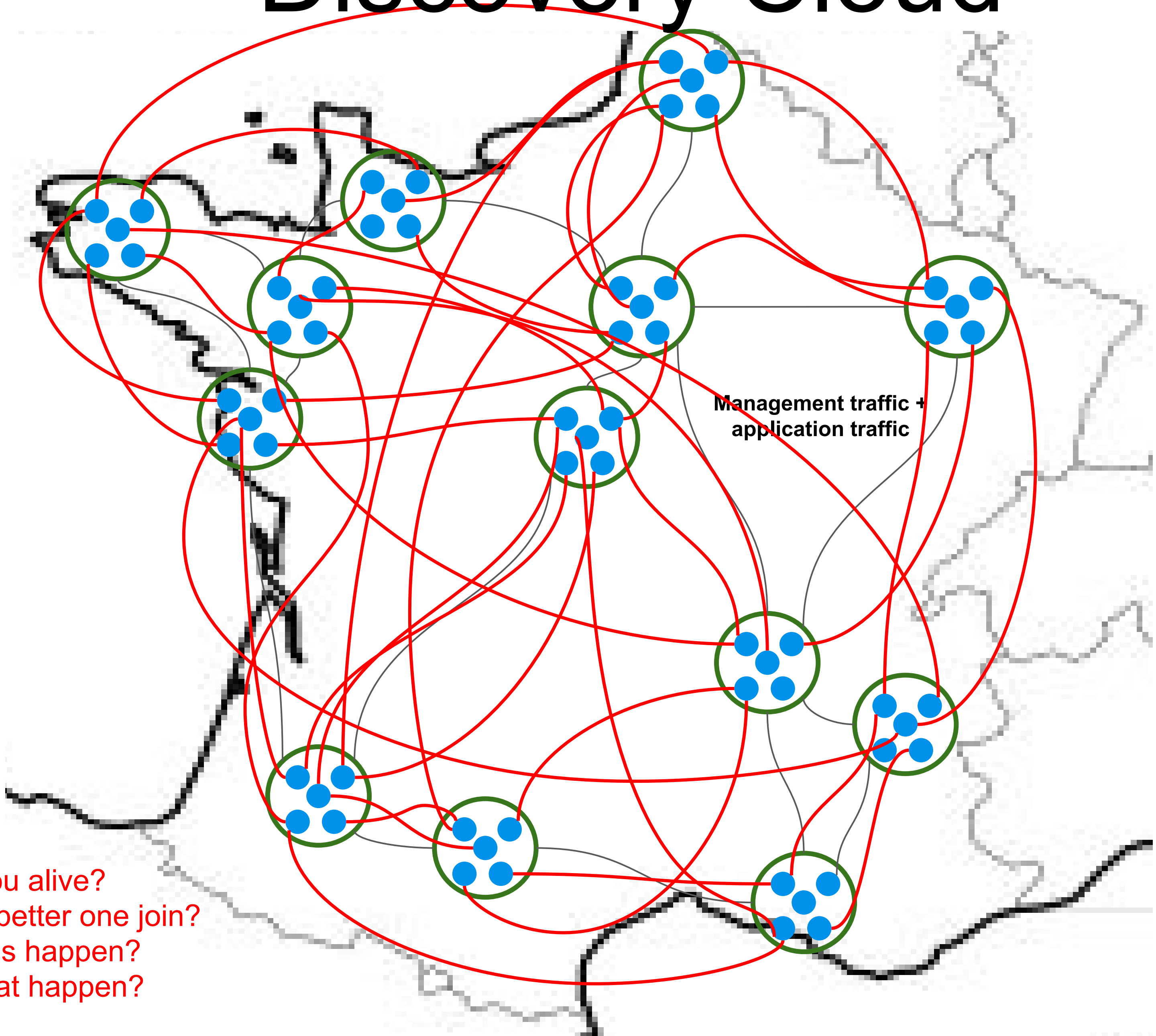
Service localization should be decentralized

- Requires a network overlay

# Koala: An overlay for decentralized clouds

...but how should this overlay be?

# Discovery Cloud



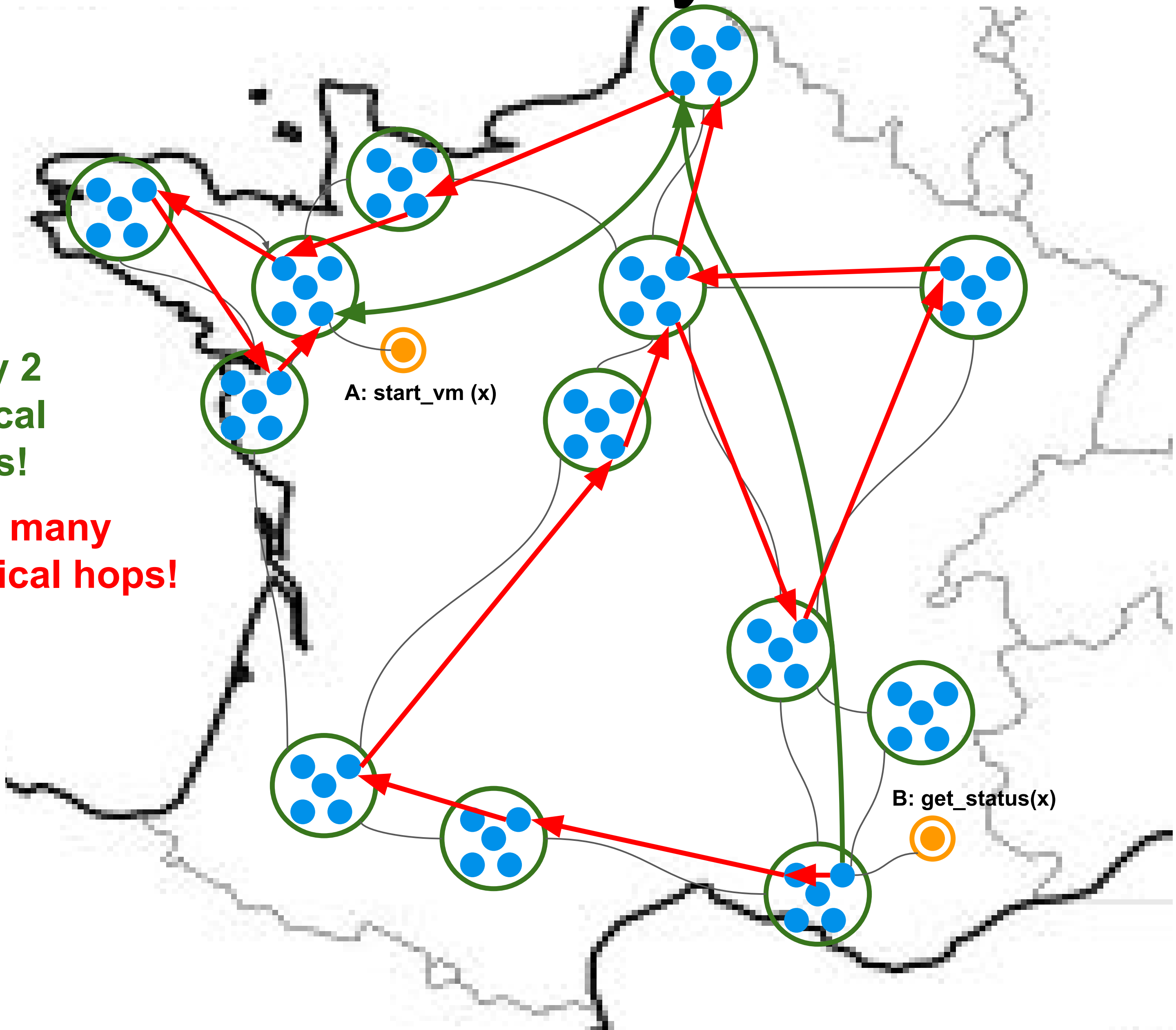
Are you alive?
Did a better one join?
Did this happen?
Did that happen?

# Koala: An overlay for decentralized clouds

...but how should this overlay be?

1. **Lazy (update overlay only when needed)**

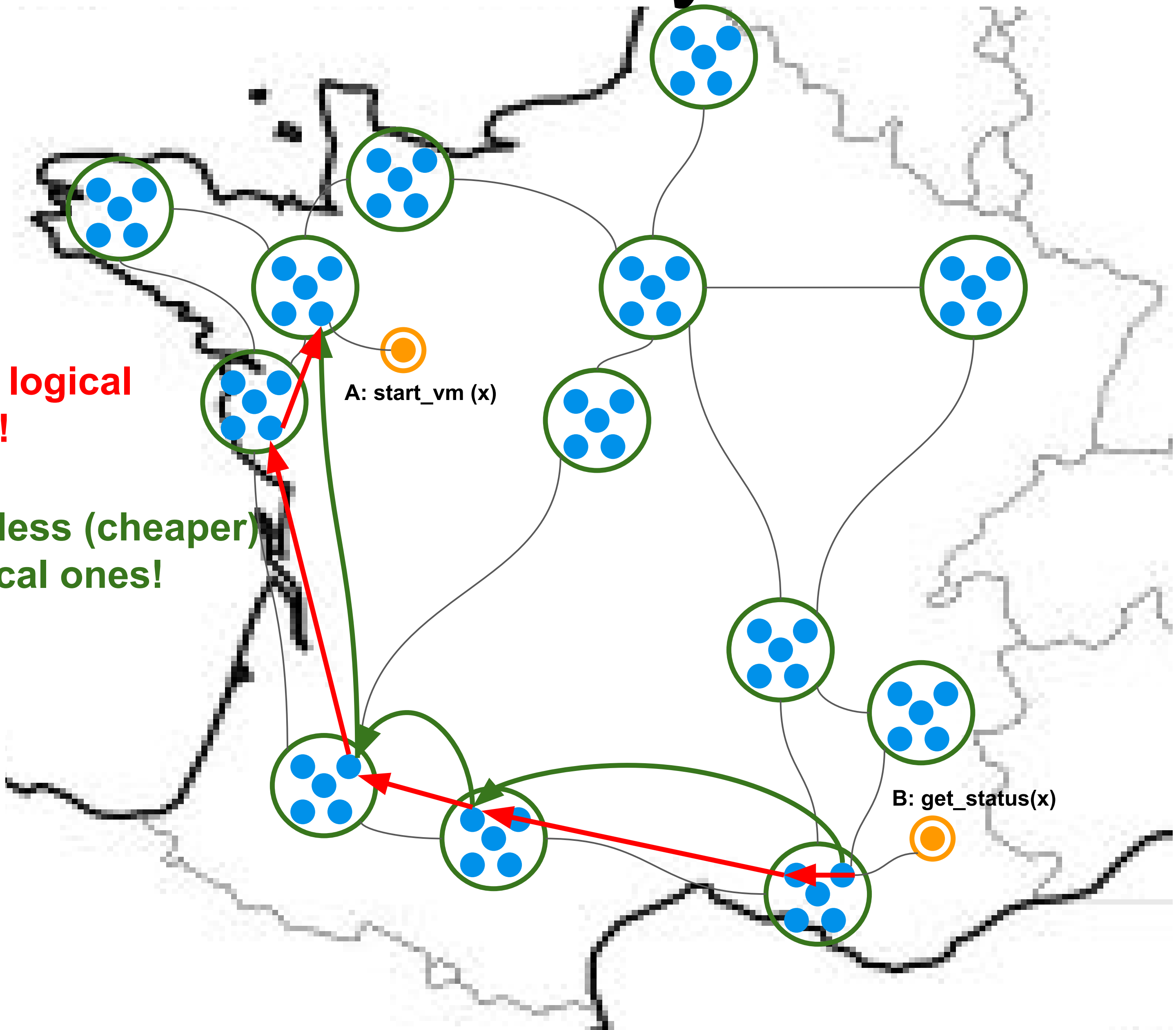
# Discovery Cloud



Only 2 logical hops!

...but many physical hops!

# Discovery Cloud



More logical hops!

...but less (cheaper) physical ones!

# Koala: An overlay for decentralized clouds

...but how should this overlay be?

1. Lazy (update overlay only when needed)
2. **Locality-aware**

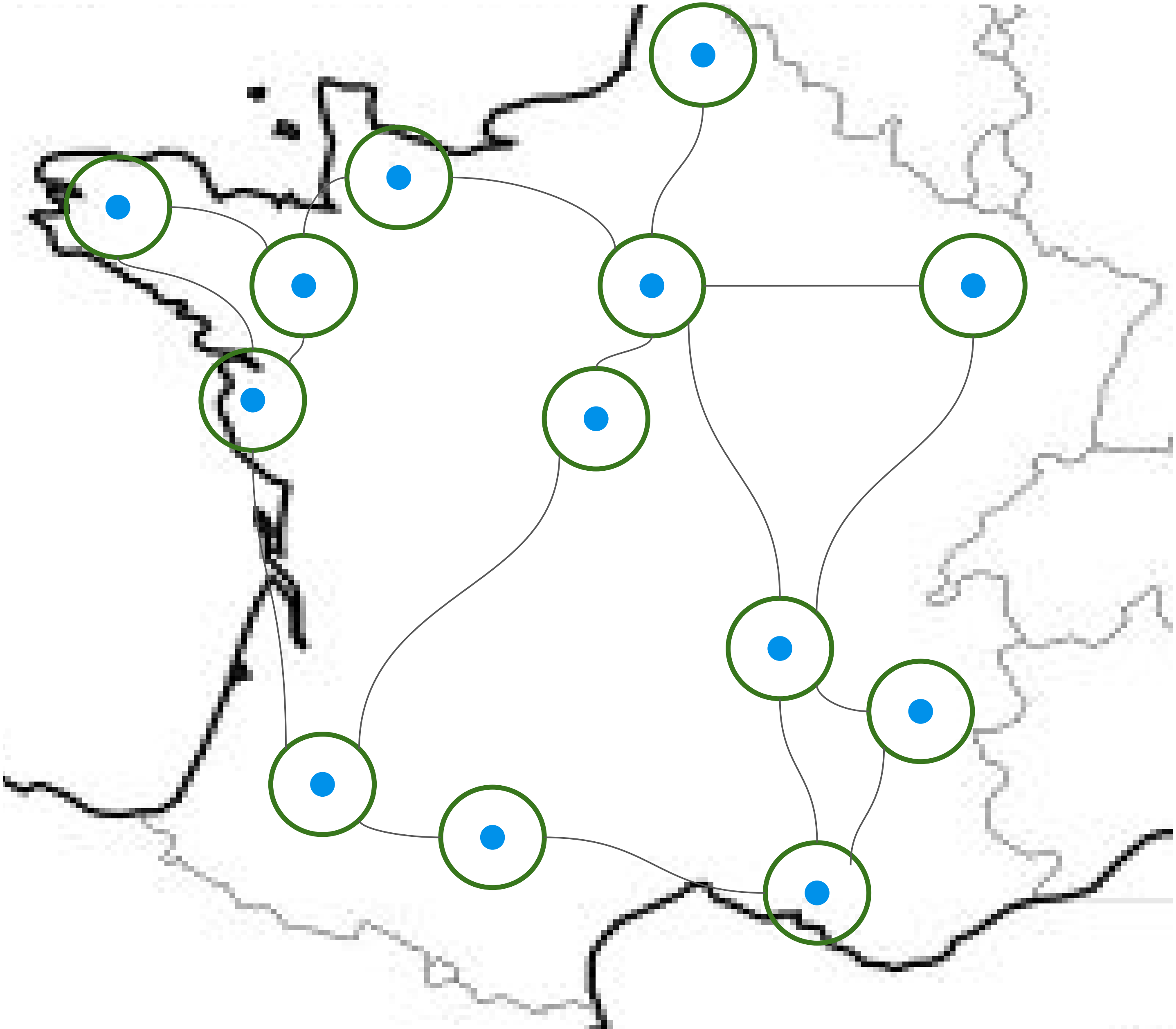
# Laziness and locality-awareness

1. How do we implement laziness?
2. How do we integrate locality-awareness?

First, focus on communication between different PoPs.

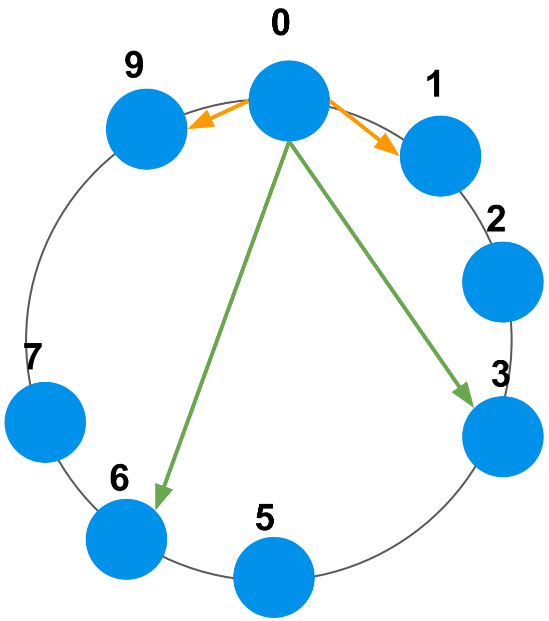
(nodes in same PoPs is discussed later)

# Flat structure



# Koala’s basics

Nodes are organized in a ring and are identified by a circular id.



Routing table of node 0

| ID | IP | RTT | IID |
|---|---|---|---|
| 9 | a.a.a.a | 50 | - |
| 1 | b.b.b.b | 150 | - |
| 3 | x.x.x.x | 125 | 4 |
| 6 | y.y.y.y | 250 | 7 |

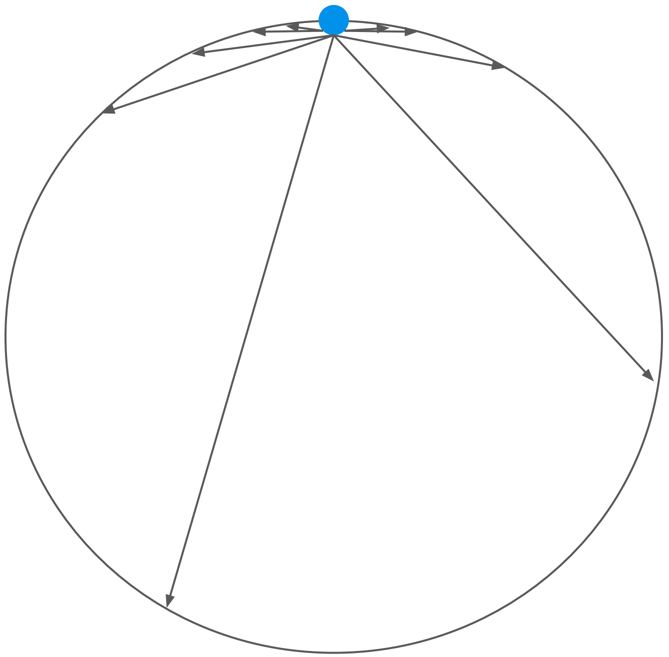
Neighbors

Long links

# Ideal long links

Still want to be an **O(log N)** hops protocol.

*Tell me who your long links are, I will tell you how efficient your routing is!*



Continuous Kleinberg distribution:

$$\mathbf{p(d) = 1/(d^*\mathit{ln}\ N)}$$

Where:

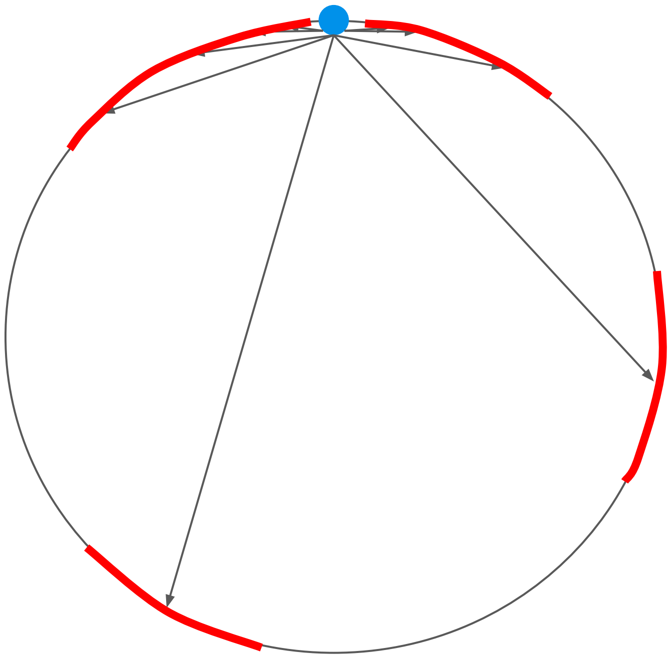
d = logical distance,
N = total nr. of nodes

Generate IDs using p(d) -> **Ideal IDs**

# Ideal long links

Generating IDs does not mean contacting them, as these nodes may not exist.

Koala is **lazy**: it does not search, it waits to learn.



Continuous Kleinberg distribution:

$$p(d) = 1/(d * \mathit{ln}\ N)$$

Where:

d = logical distance,

N = total nr. of nodes

Close to ideal is still ideal.

# Laziness: Piggybacking

Embed information about nodes within the message

2 sources:

1. Nodes in the path
2. Nodes in the routing table of the nodes in the path

# Locality-awareness

Find the cheapest logical path (latency-wise)

We could choose systematically the cheapest hop.

...but the logical distance needs to be reduced as well

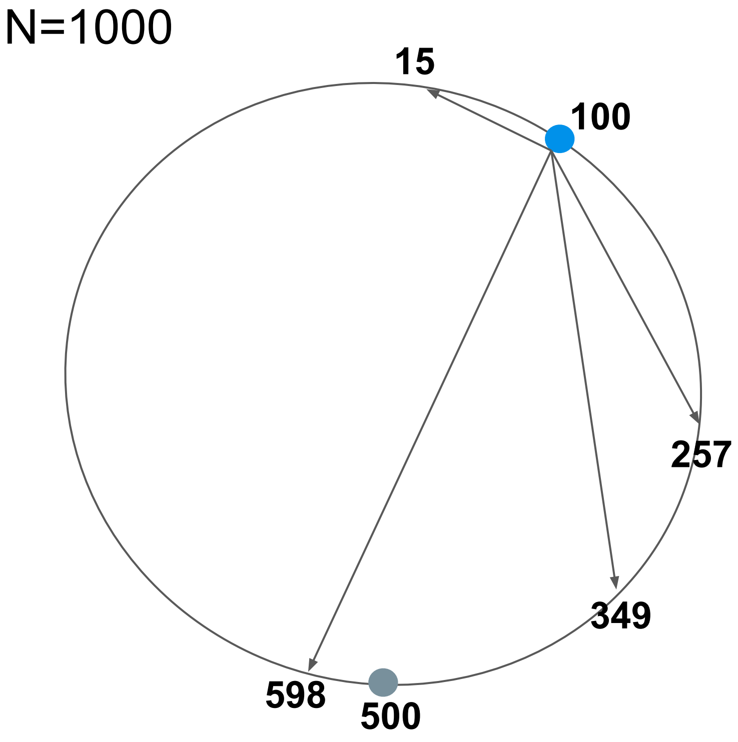
A tradeoff between logical distance and latency

For each entry we calculate:

$$\mathbf{Q(re) = 1/(\alpha \times re.distance + (1-\alpha) \times norm(re.RTT))}$$

# Locality-awareness: Routing

Route from **100** to **500**



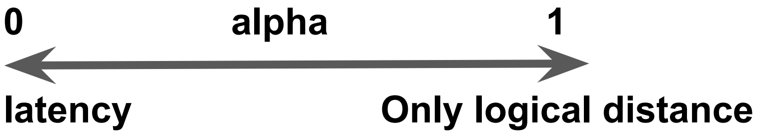
Routing table of node 100

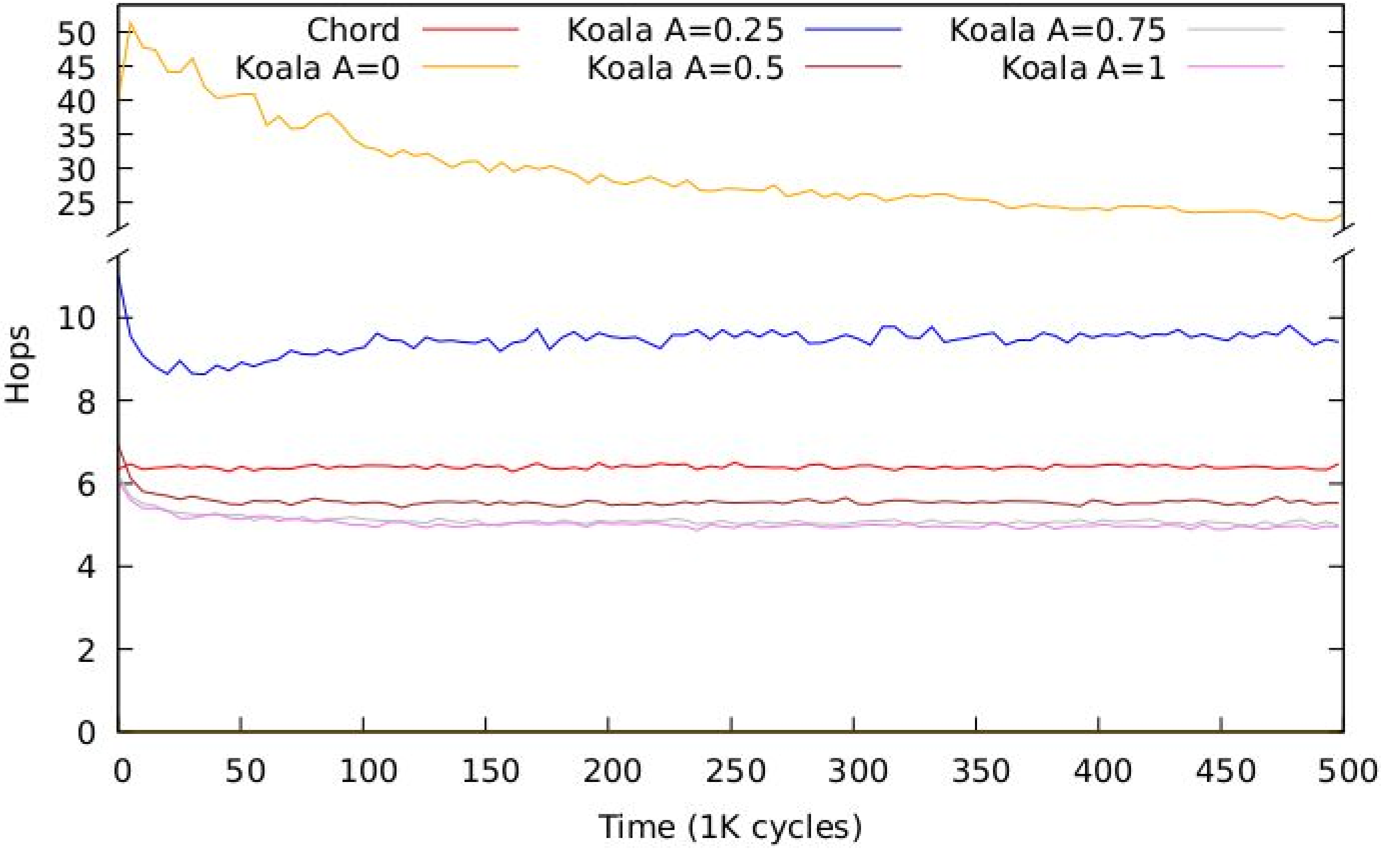
| ID | RTT | DistToDest |
|---|---|---|
| 15 | 340 | 485 |
| 257 | 105 | 243 |
| 349 | 194 | 151 |
| 598 | 1230 | 98 |

**$Q(re) = 1/(\alpha \times re.distance + (1-\alpha) \times norm(re.RTT))$**

# Locality-aware vs Greedy (hops)

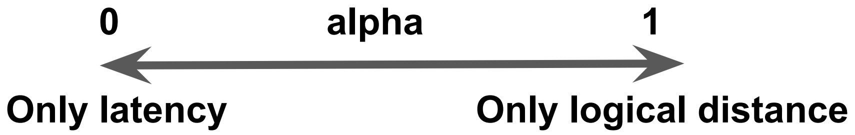
N=10K

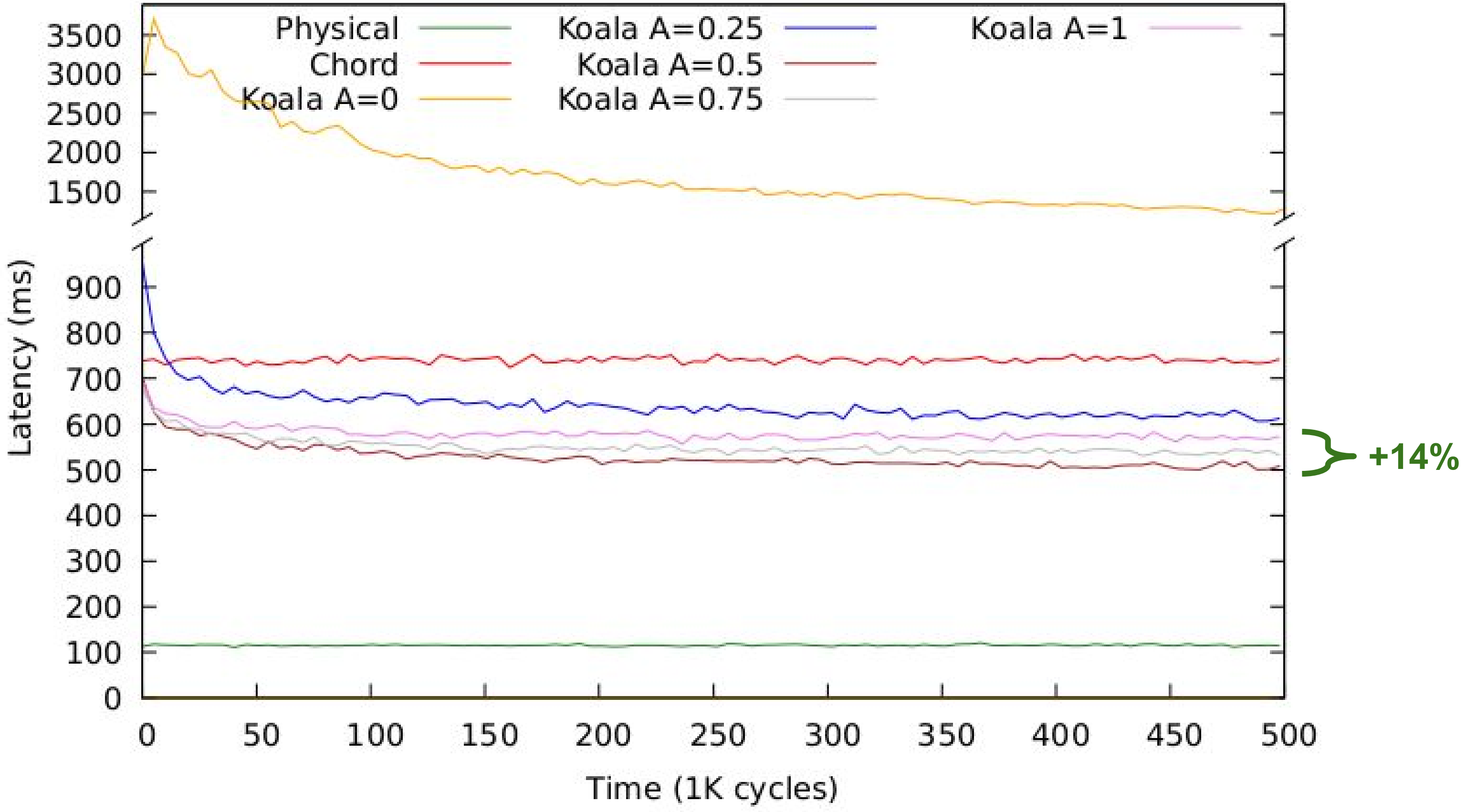




# Locality-aware vs Greedy (latency)

N=10K



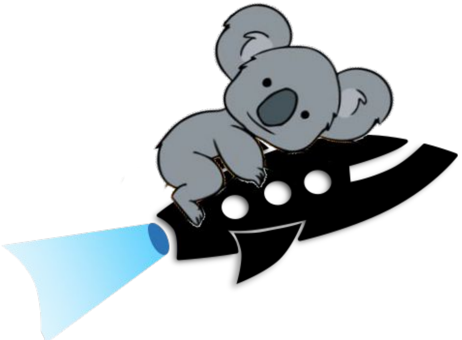


# Conclusion, current and future work

Further evaluation concerning scalability, locality-awareness and resilience have reported in a paper submitted to CloudCom2017.

Currently we are working on the adaptation of our protocol to address the two-layers nature of our physical topology (intra-PoP and inter-PoP) while still remaining flat (no hierarchies)

In the context of the Discovery, we aim at using Koala as a communication bus for OpenStack (a decentralized broker)



# Thank you!

Questions?