The art of service discovery at scale

Nitesh Kant,
Software Engineer,
Netflix Edge Engineering.

 *@NiteshKant*

Nitesh Kant *@NiteshKant*

Who Am I?

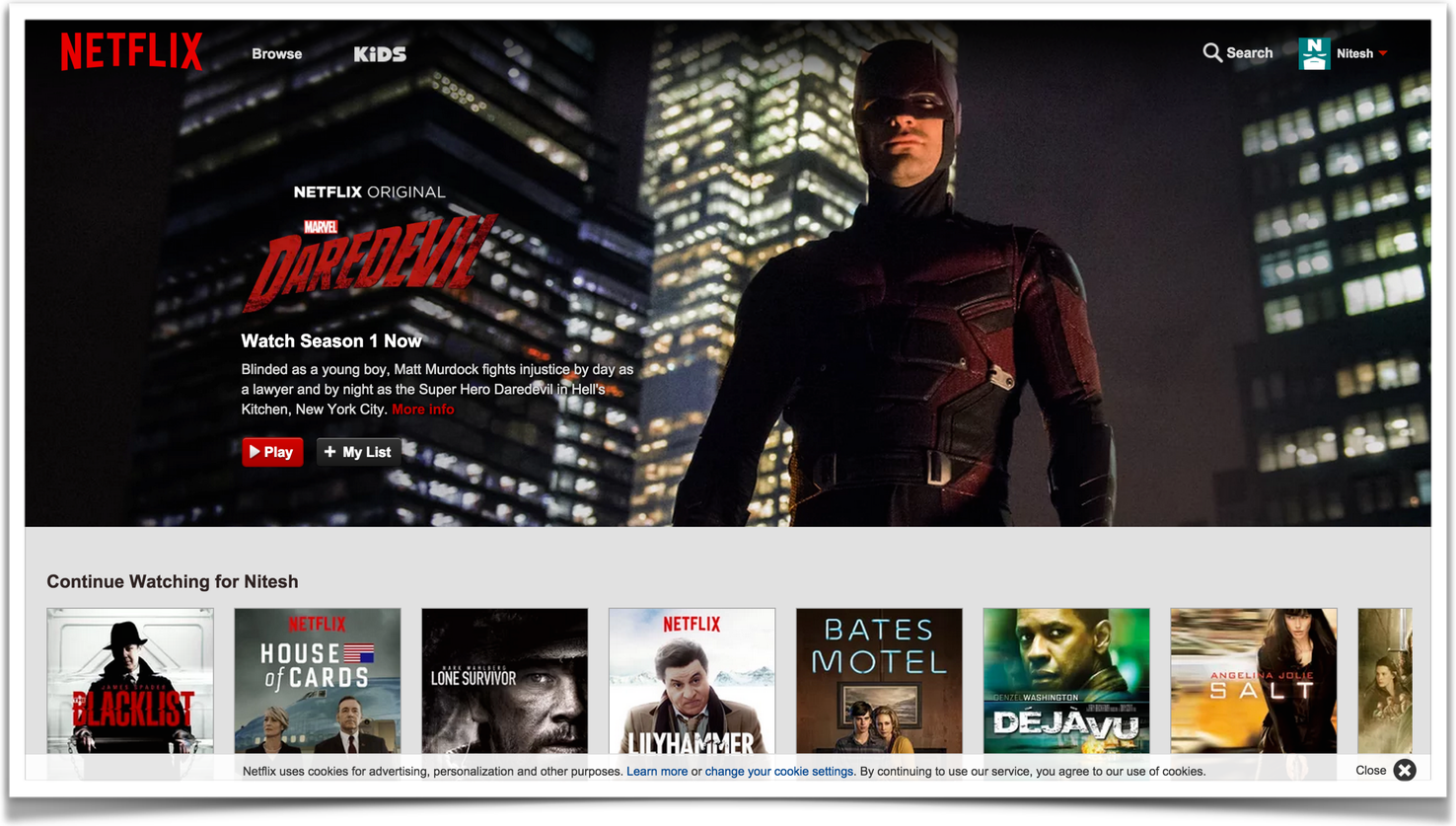
- Engineer, Edge Engineering, Netflix.
- Core contributor, RxNetty*
- Contributor, Zuul**
- Ran Netflix's Service Discovery for a year.
- Conceptualized and designed Eureka v2*** .

* https://github.com/ReactiveX/RxNetty

** https://github.com/Netflix/zuul

*** https://github.com/Netflix/eureka

What is Service Discovery?

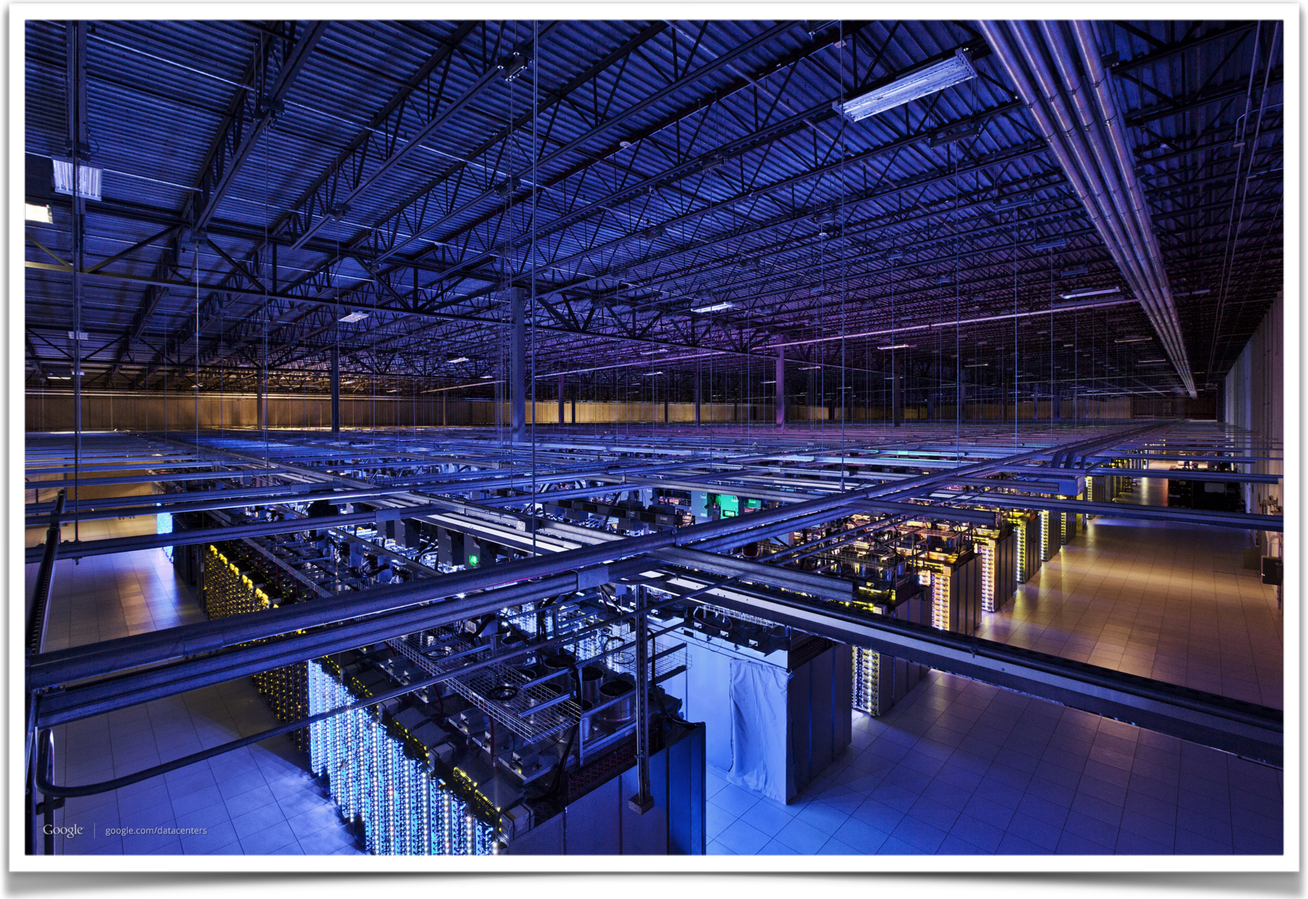


Delivering Netflix

What goes behind the scenes?

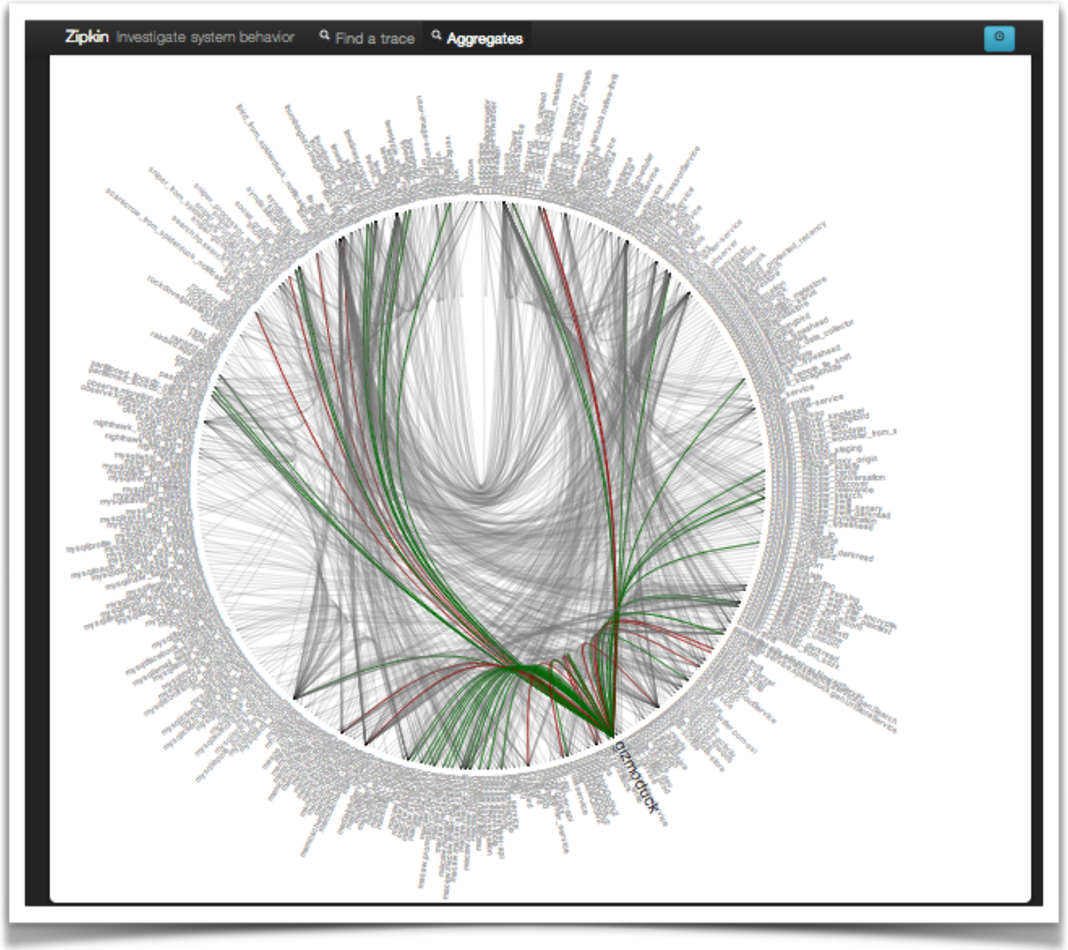
Pipe Dream

Always available, infinitely scalable machine.



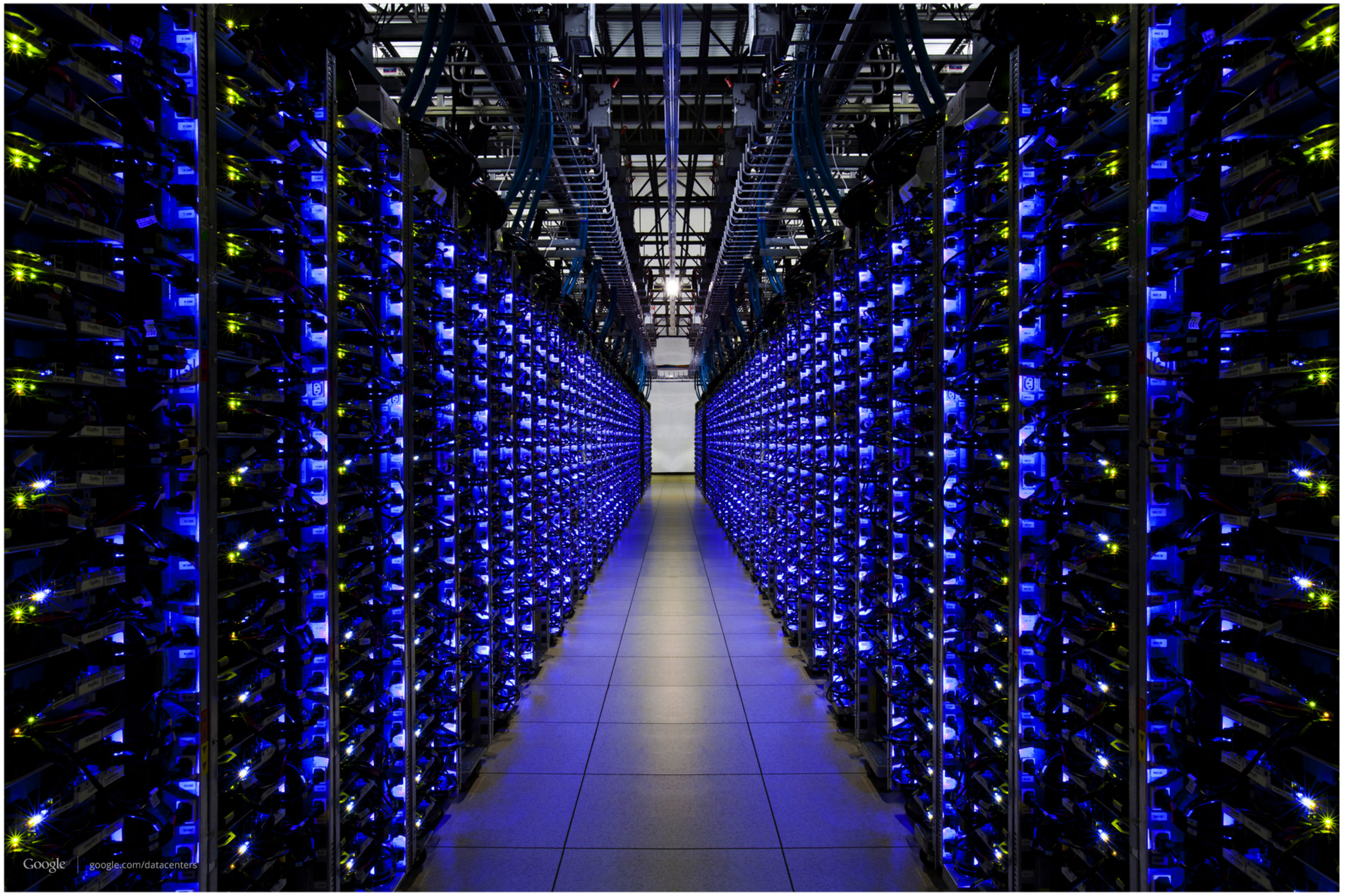
Reality

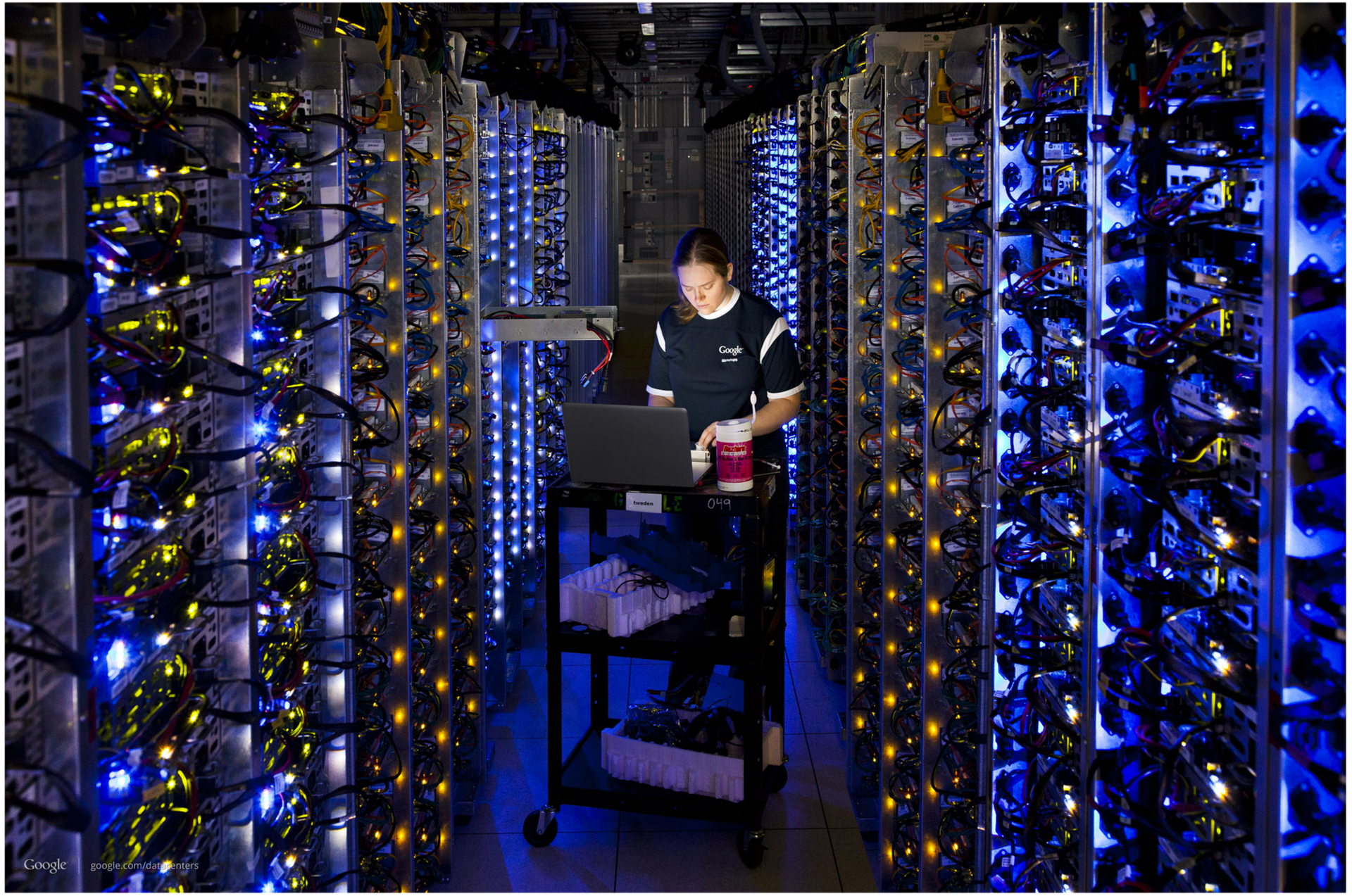
Thousands of machines with hundreds of microservices

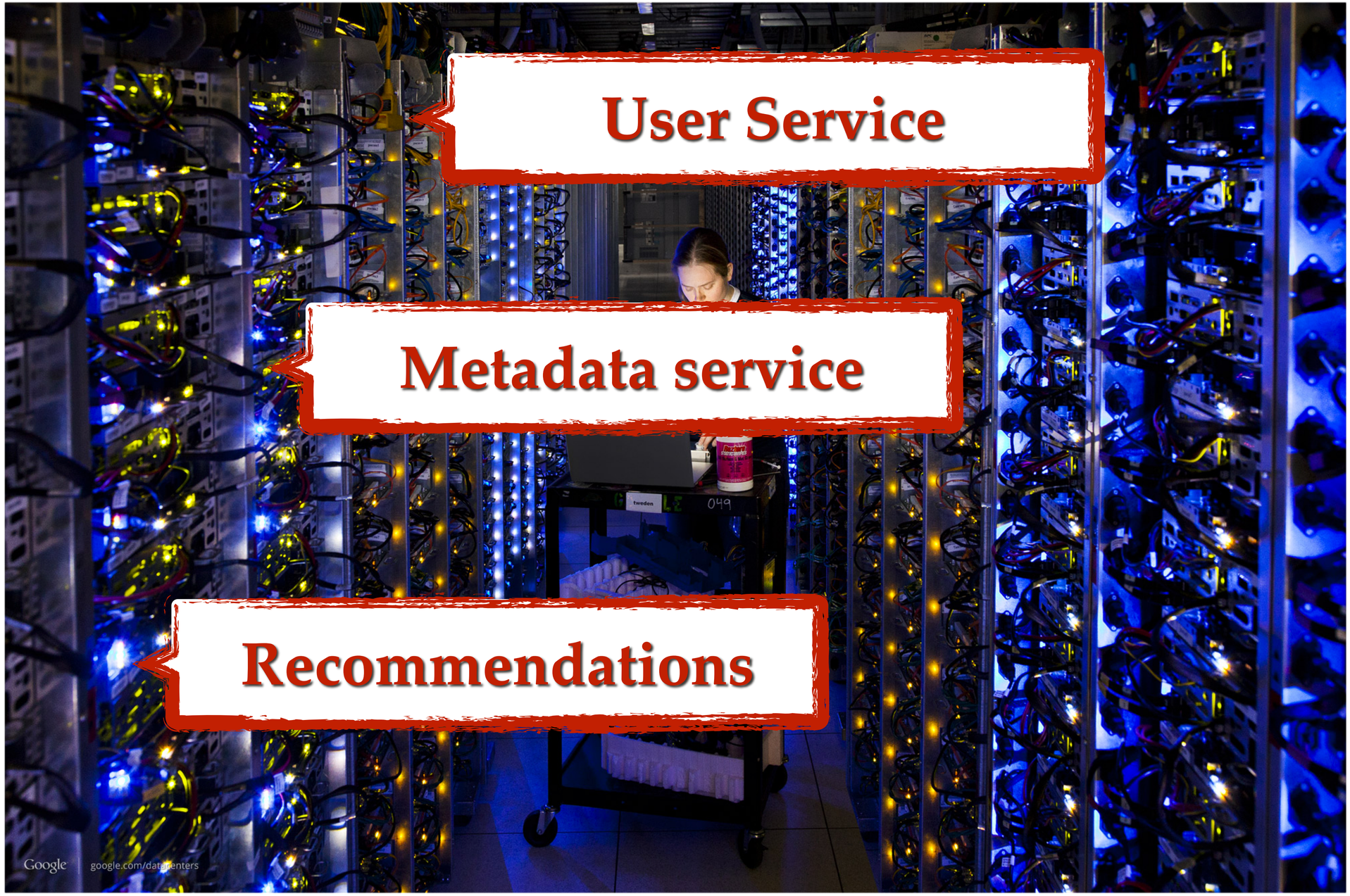


Microservices

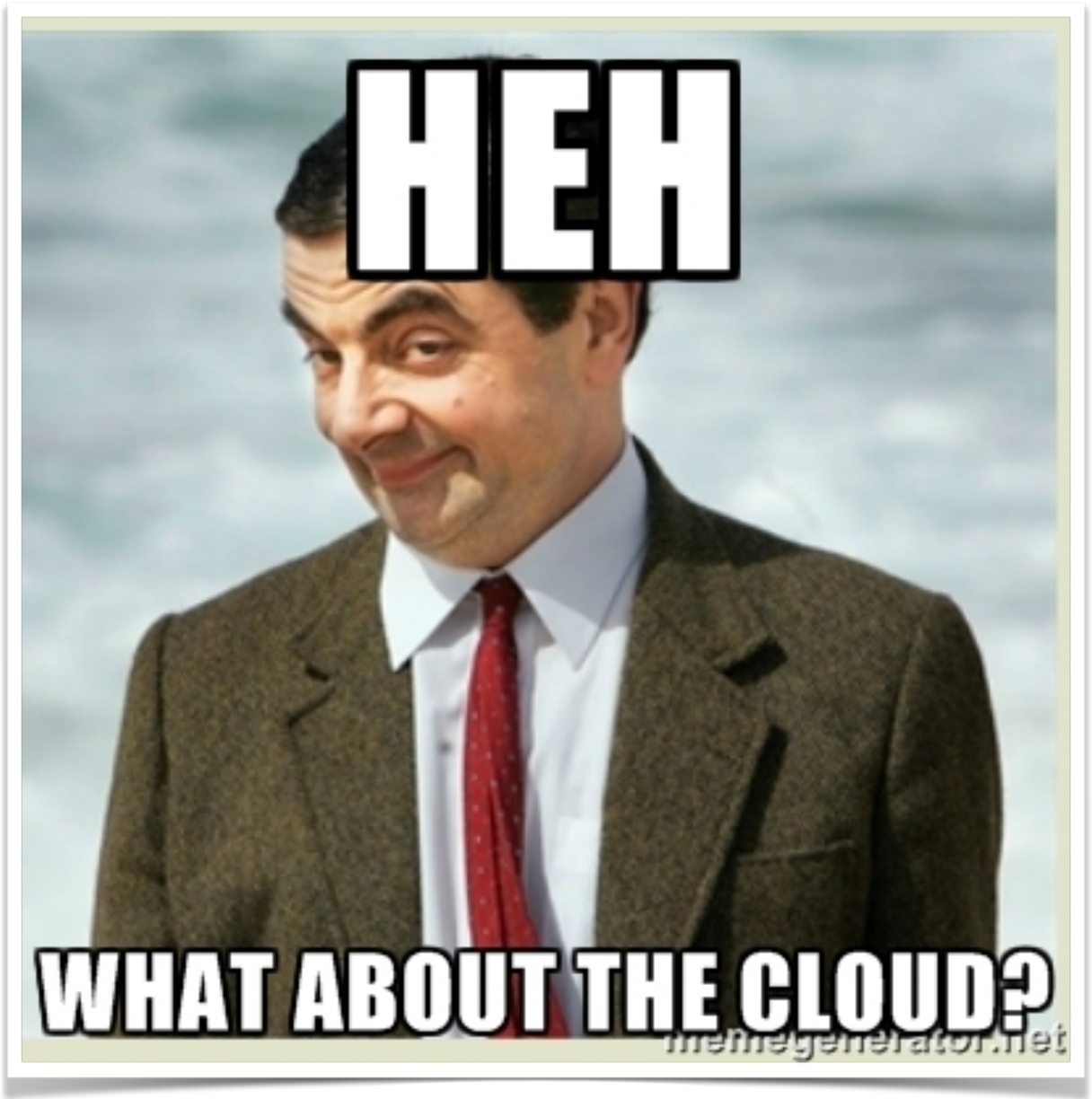
A typical relationship between different microservices.

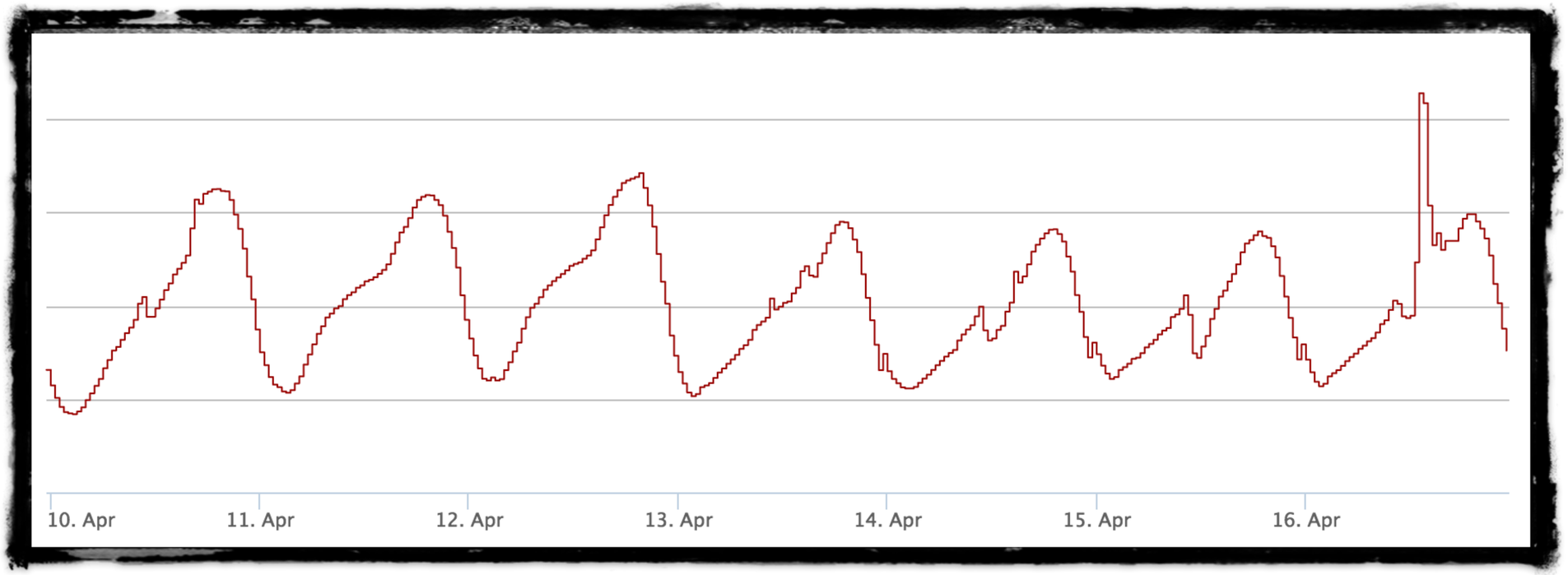

Google
google.com/datacenters


Google
Google
google.com/datacenters


User Service
Metadata service
Recommendations
Google
google.com/datacenters

Service Discovery!

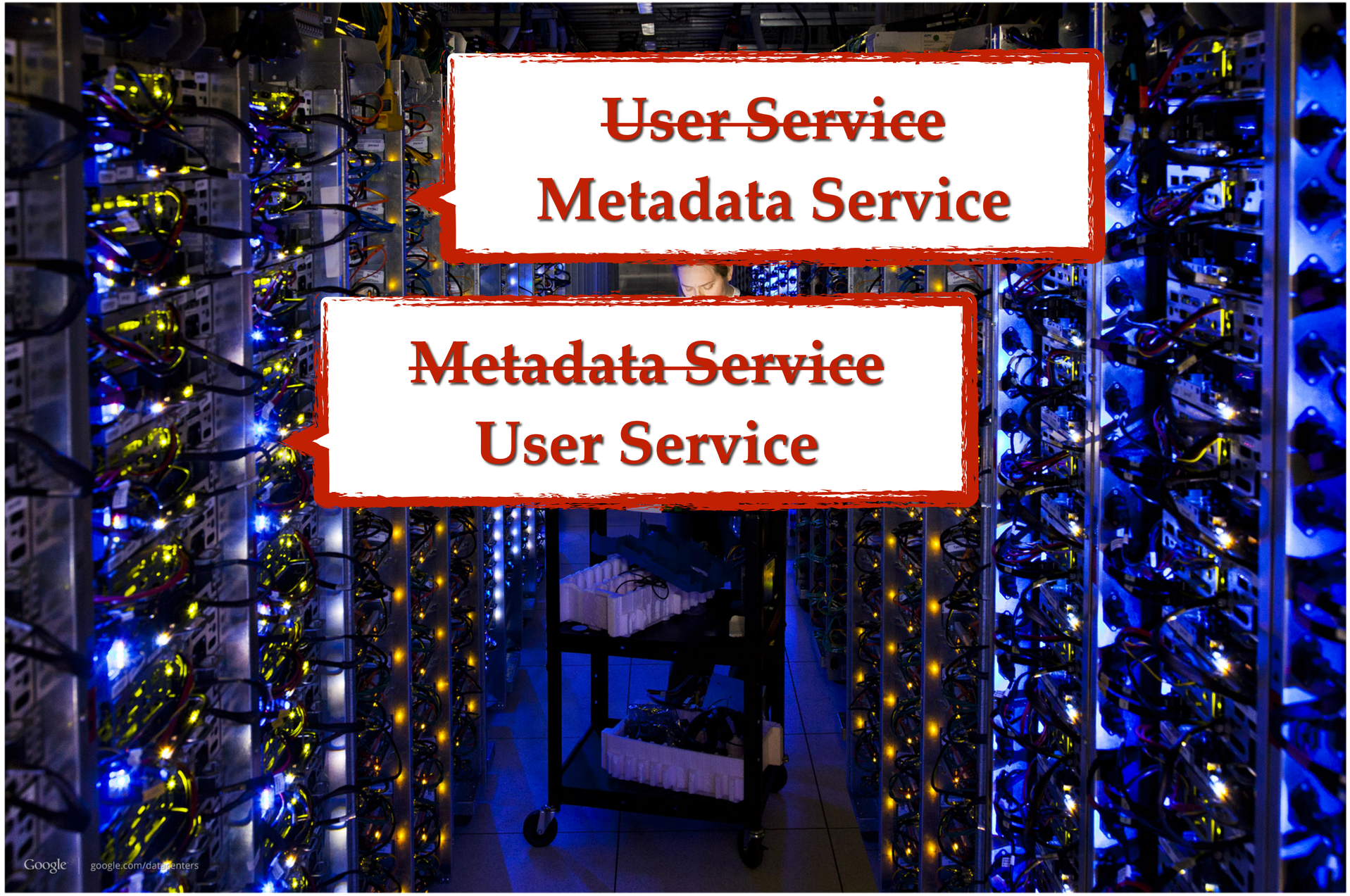

HEH
WHAT ABOUT THE CLOUD?
memegenerator.net



Instance variations

We auto-scale up/down all the time.

VMs crash

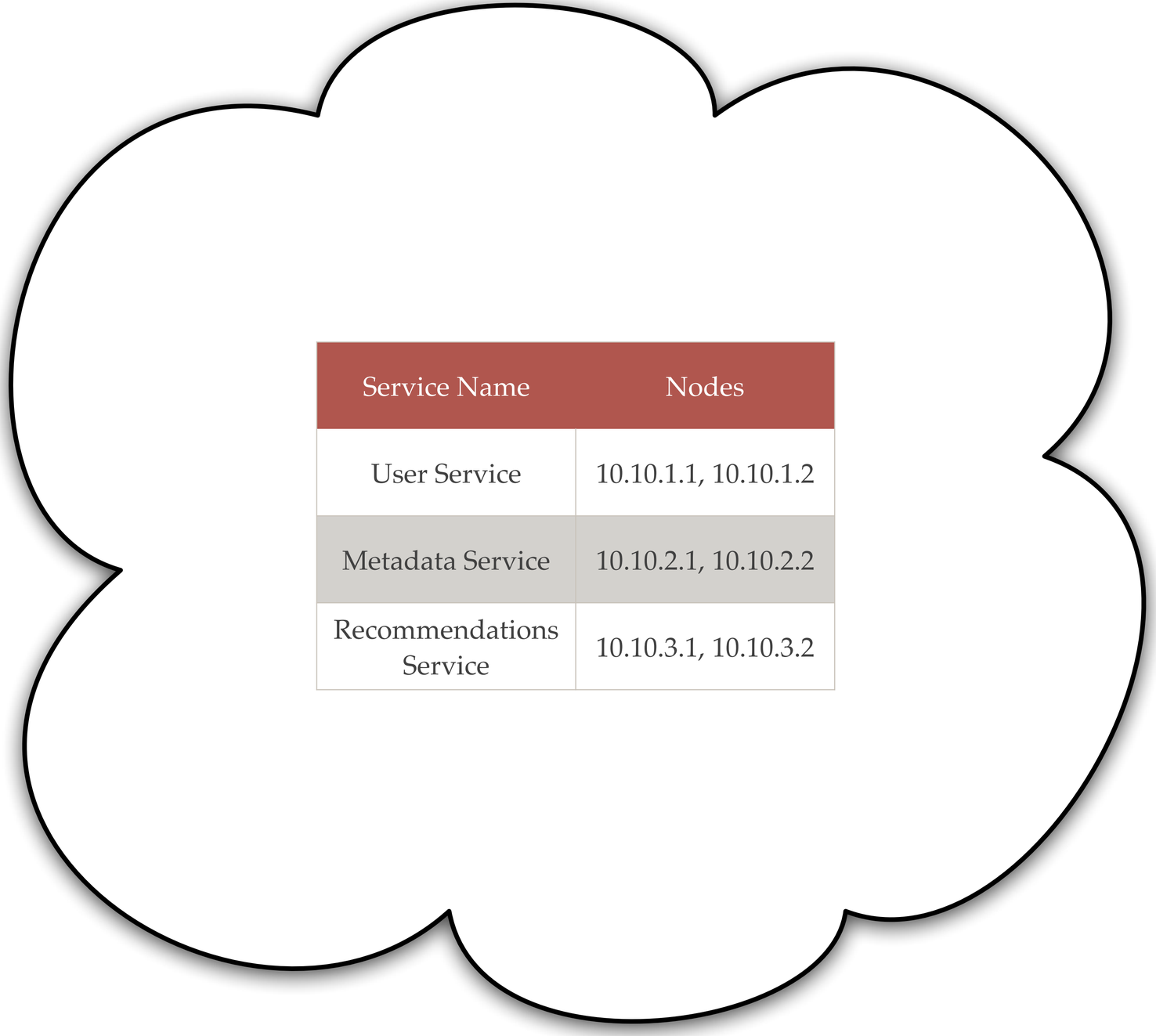

~~User Service~~
Metadata Service
~~Metadata Service~~
User Service
Google
google.com/datacenters

It’s an ever changing eco-system,
static mappings, don’t work.

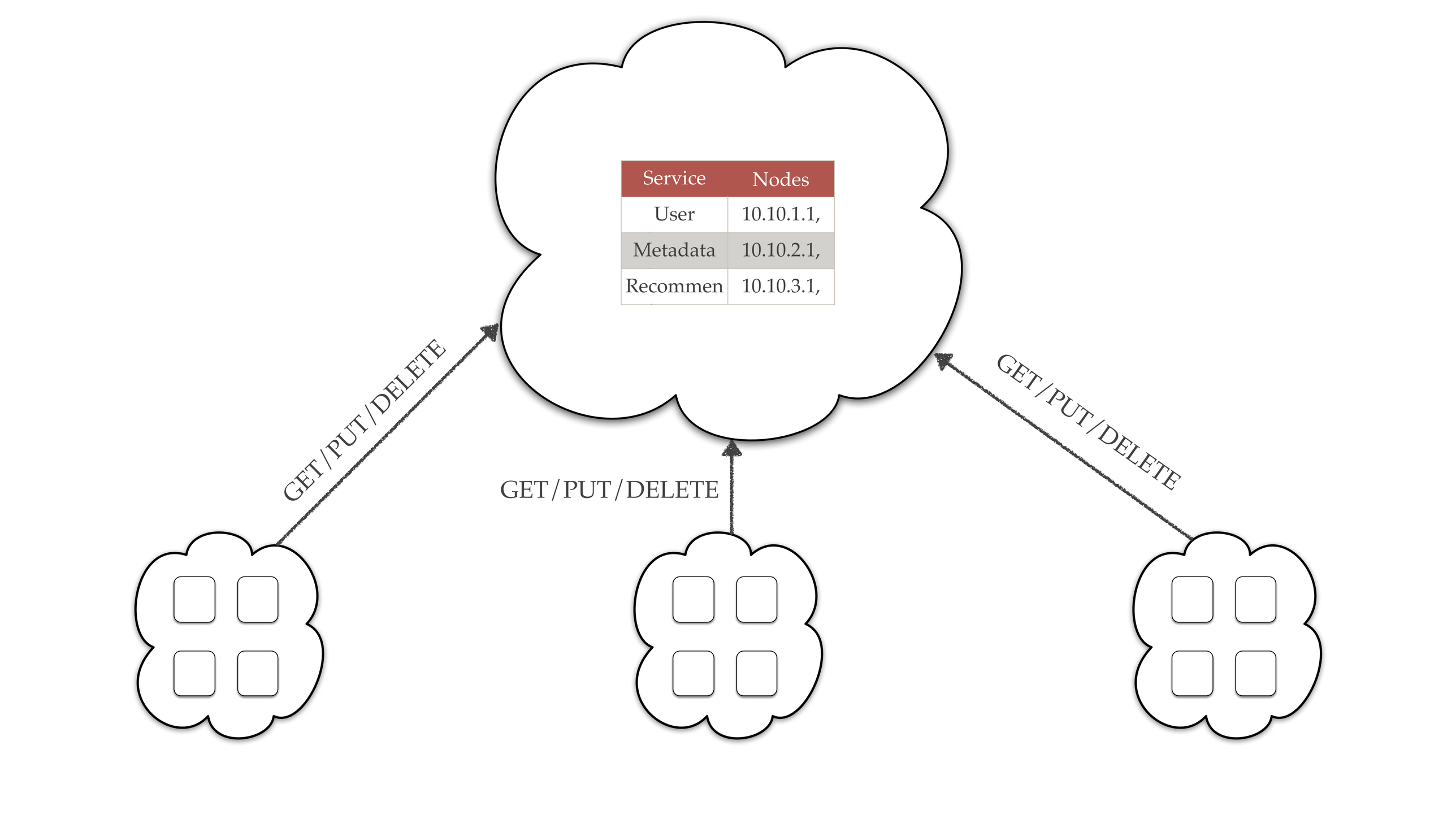
Service discovery is a problem of cloud environments.


Google
google.com/datacenters

Service Name	Nodes
User Service	10.10.1.1, 10.10.1.2, 10.10.1.3, 10.10.1.4
Metadata Service	10.10.2.1
Recommendations Service	10.10.3.1, 10.10.3.2, 10.10.3.3

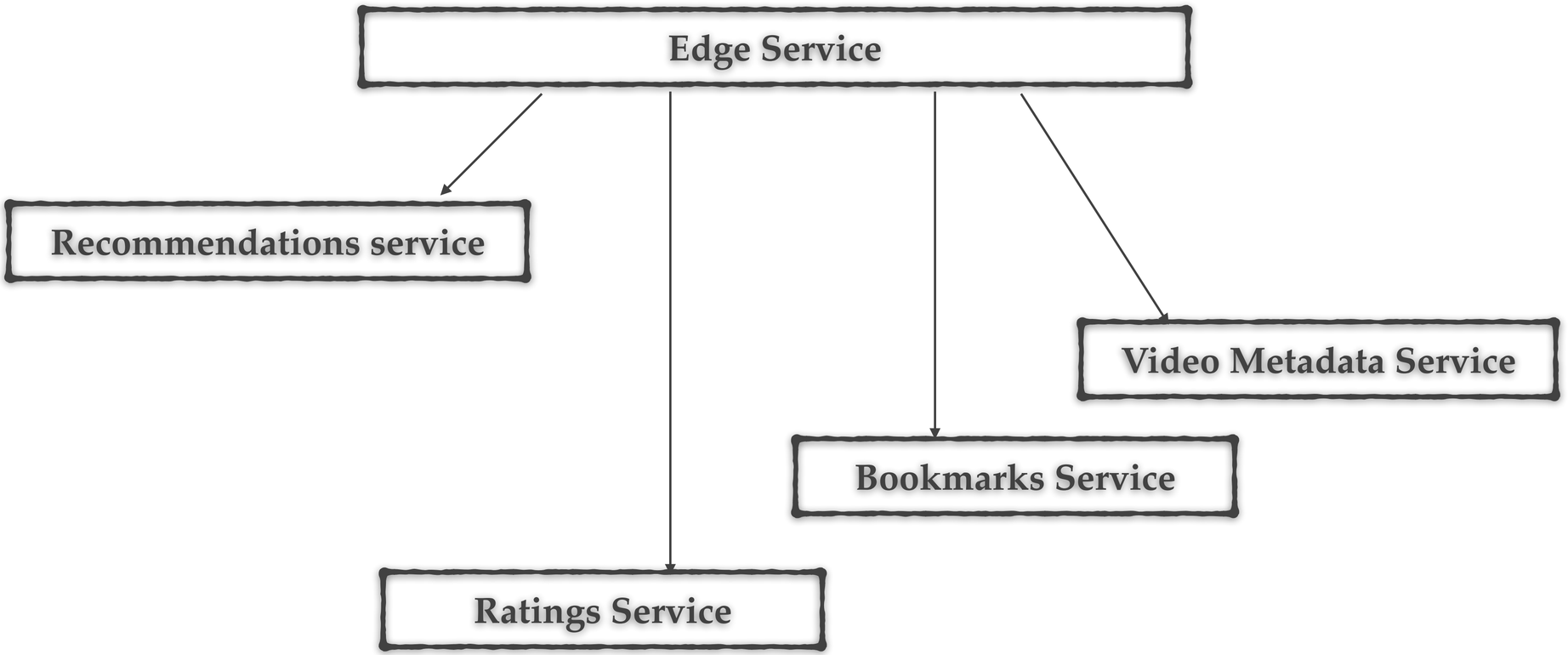


Service Name	Nodes
User Service	10.10.1.1, 10.10.1.2
Metadata Service	10.10.2.1, 10.10.2.2
Recommendations Service	10.10.3.1, 10.10.3.2


Service
Nodes
User
10.10.1.1,
Metadata
10.10.2.1,
Recommen
10.10.3.1,
GET/PUT/DELETE
GET/PUT/DELETE
GET/PUT/DELETE

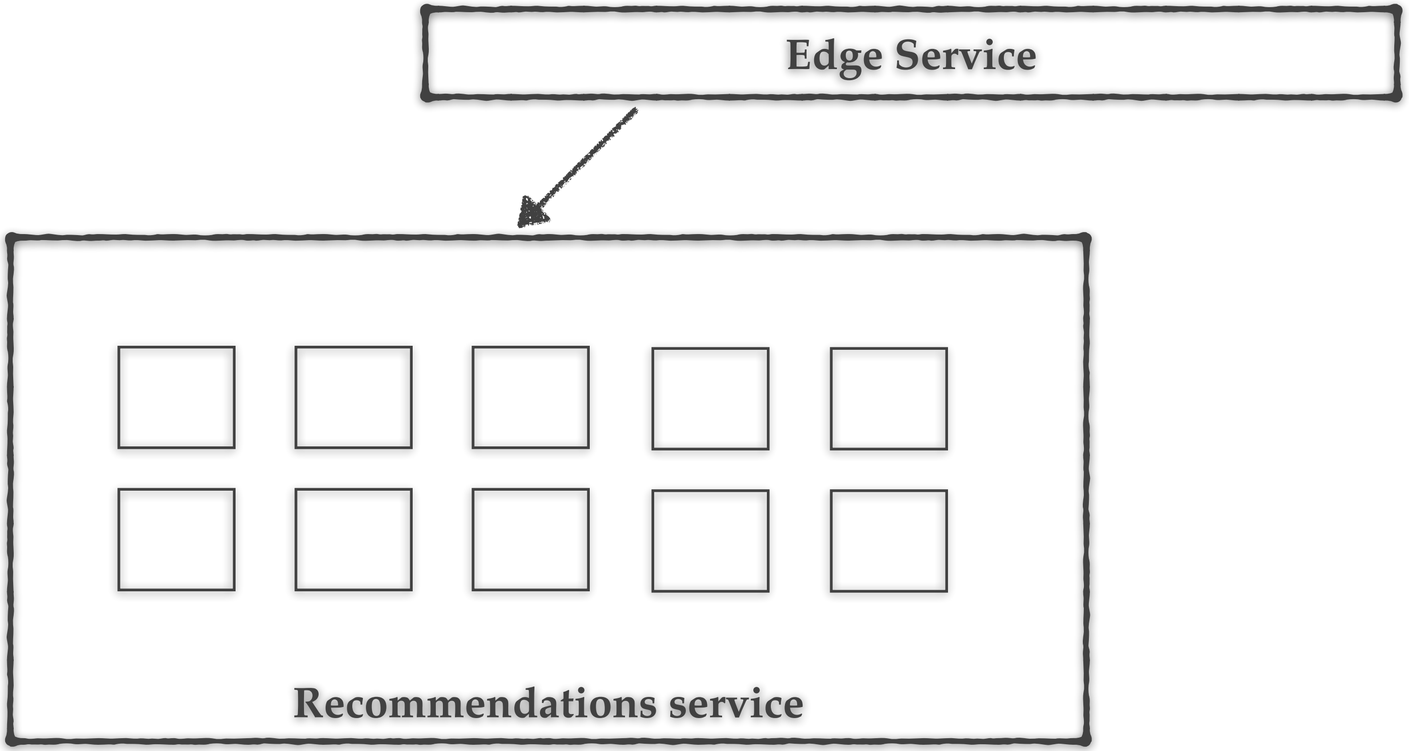
How is service discovery used?

Delivering Netflix



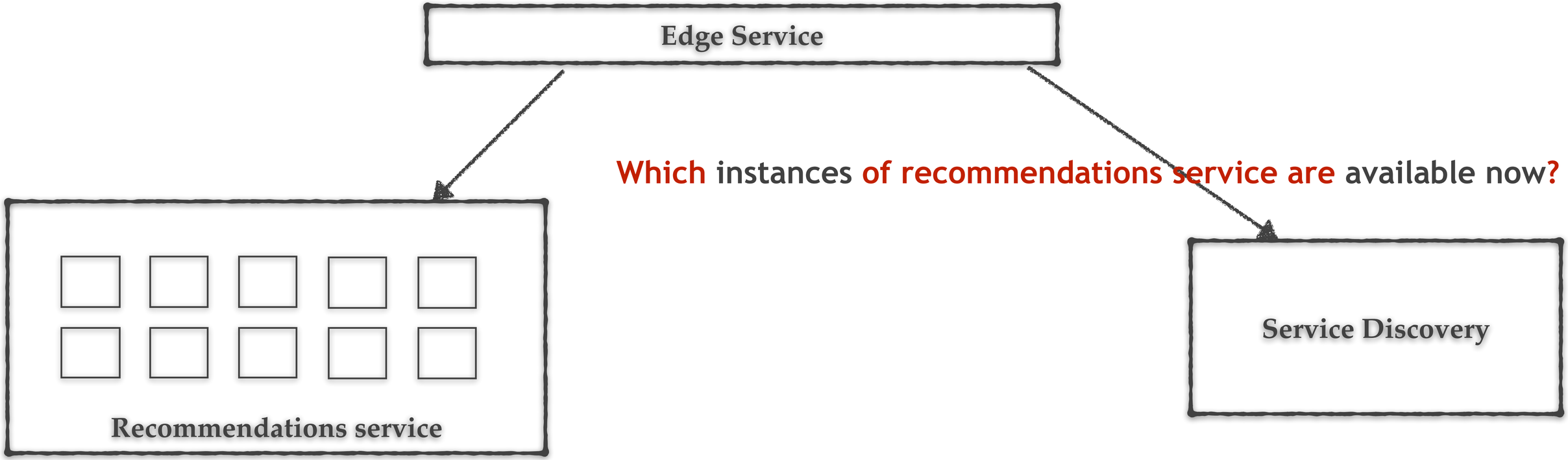
Disclaimer: This is an example and not an exact representation of the processing

Delivering Netflix



Disclaimer: This is an example and not an exact representation of the processing

Delivering Netflix

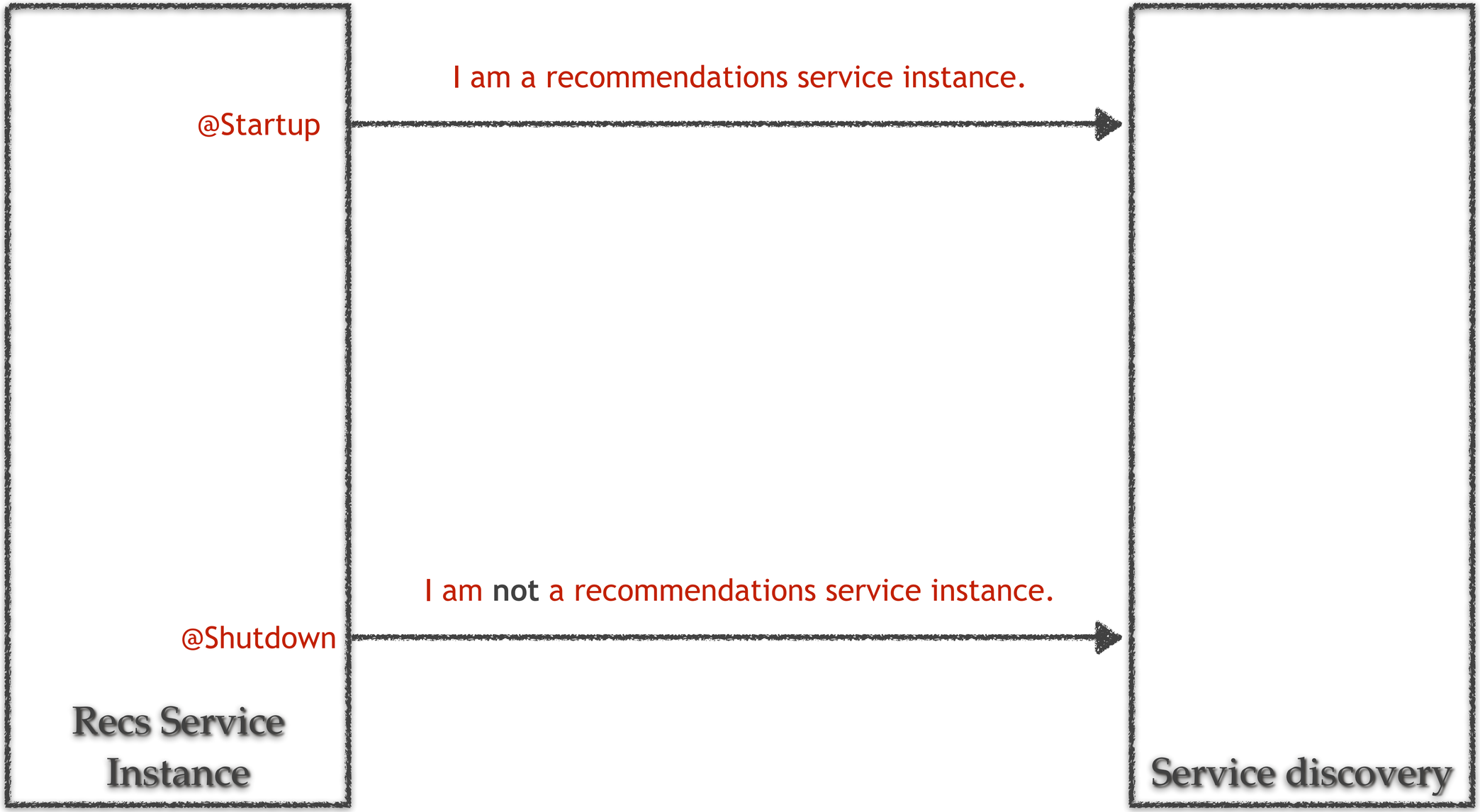


Disclaimer: This is an example and not an exact representation of the processing

Which instances of recommendations service are available now?

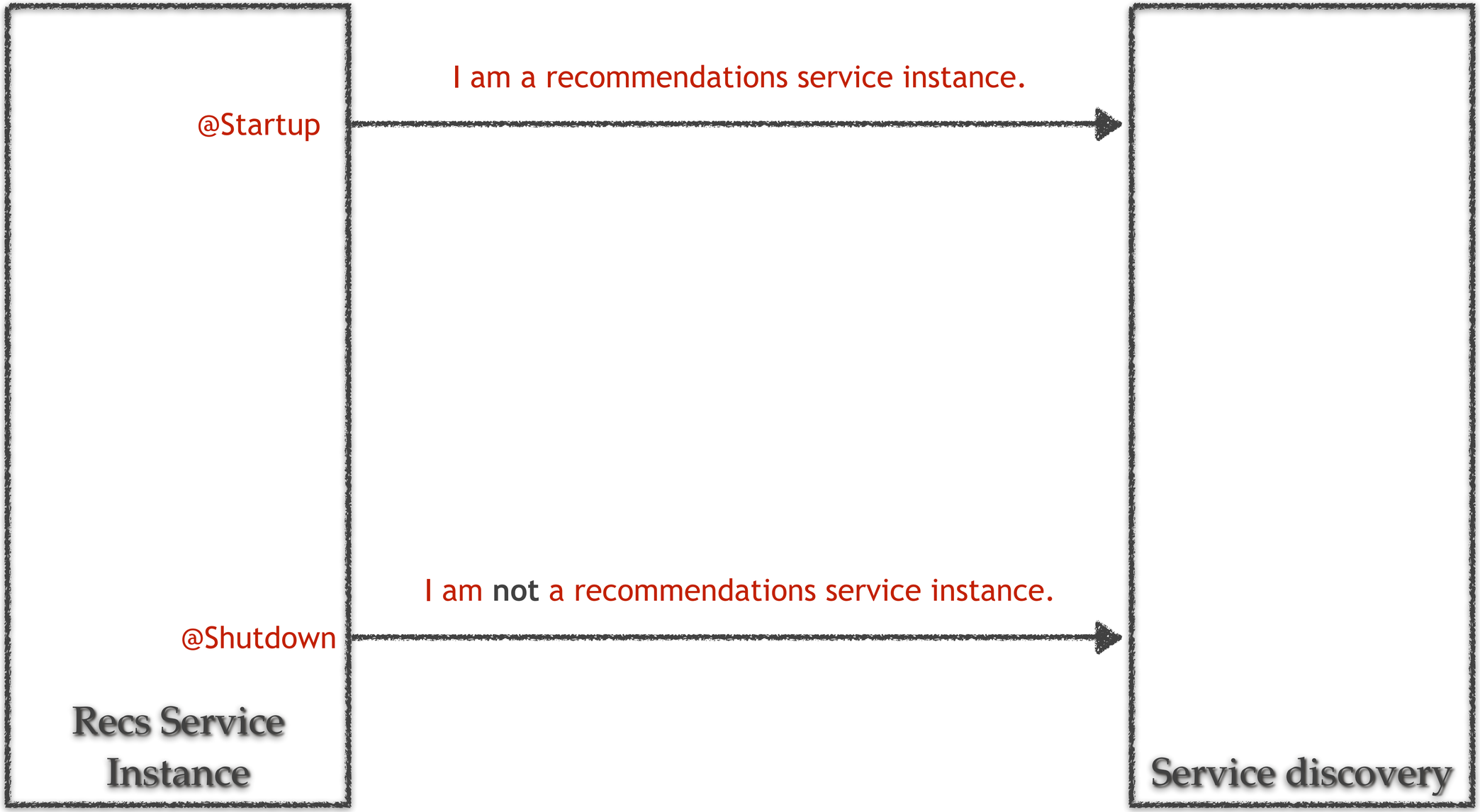
Which of the machines in the datacenter have the recommendations service software deployed?

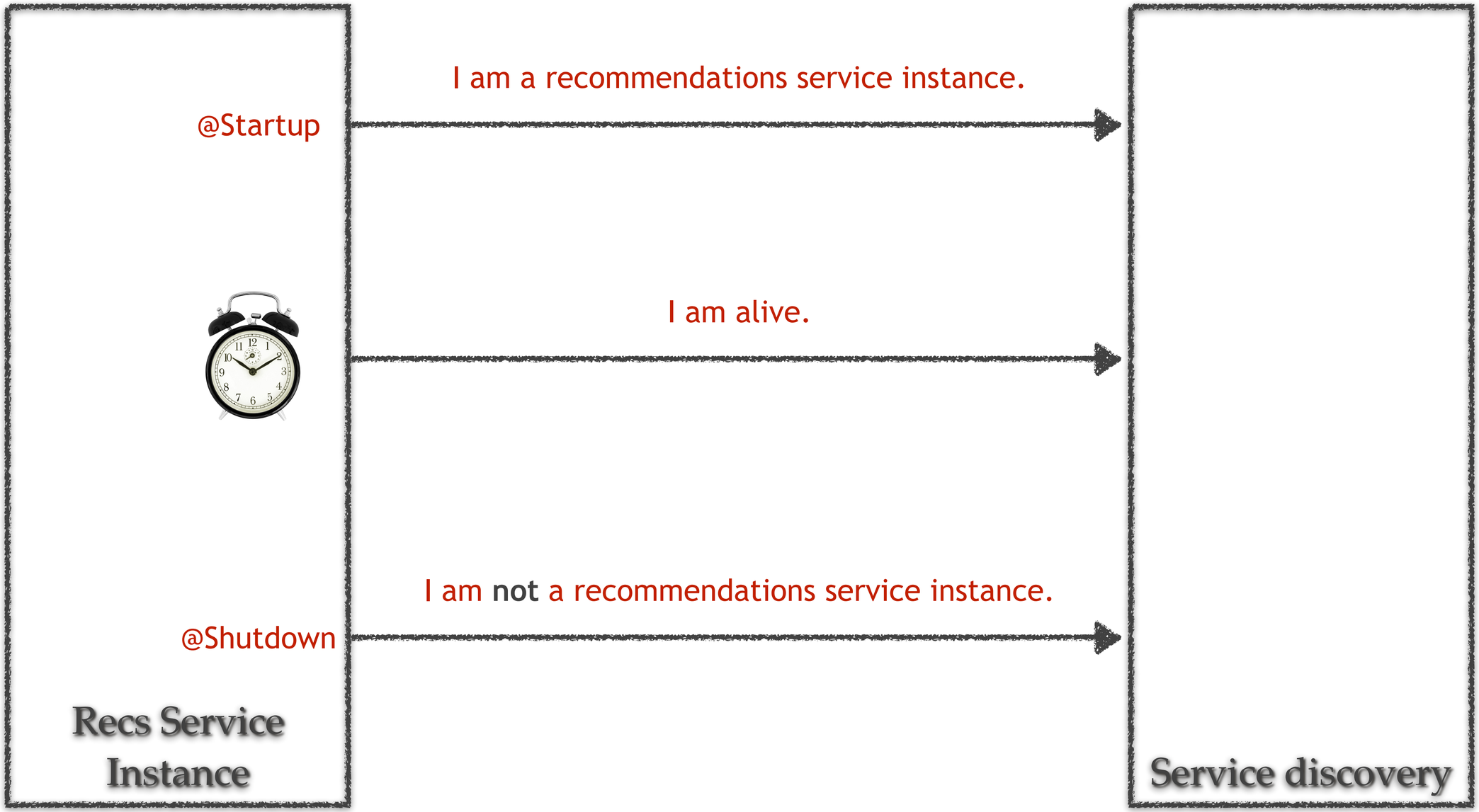
Service Name	Nodes
User Service	10.10.1.1, 10.10.1.2
Metadata Service	10.10.2.1, 10.10.2.2
Recommendations Service	10.10.3.1, 10.10.3.2


I am a recommendations service instance.
@Startup
I am not a recommendations service instance.
@Shutdown
Recs Service Instance
Service discovery

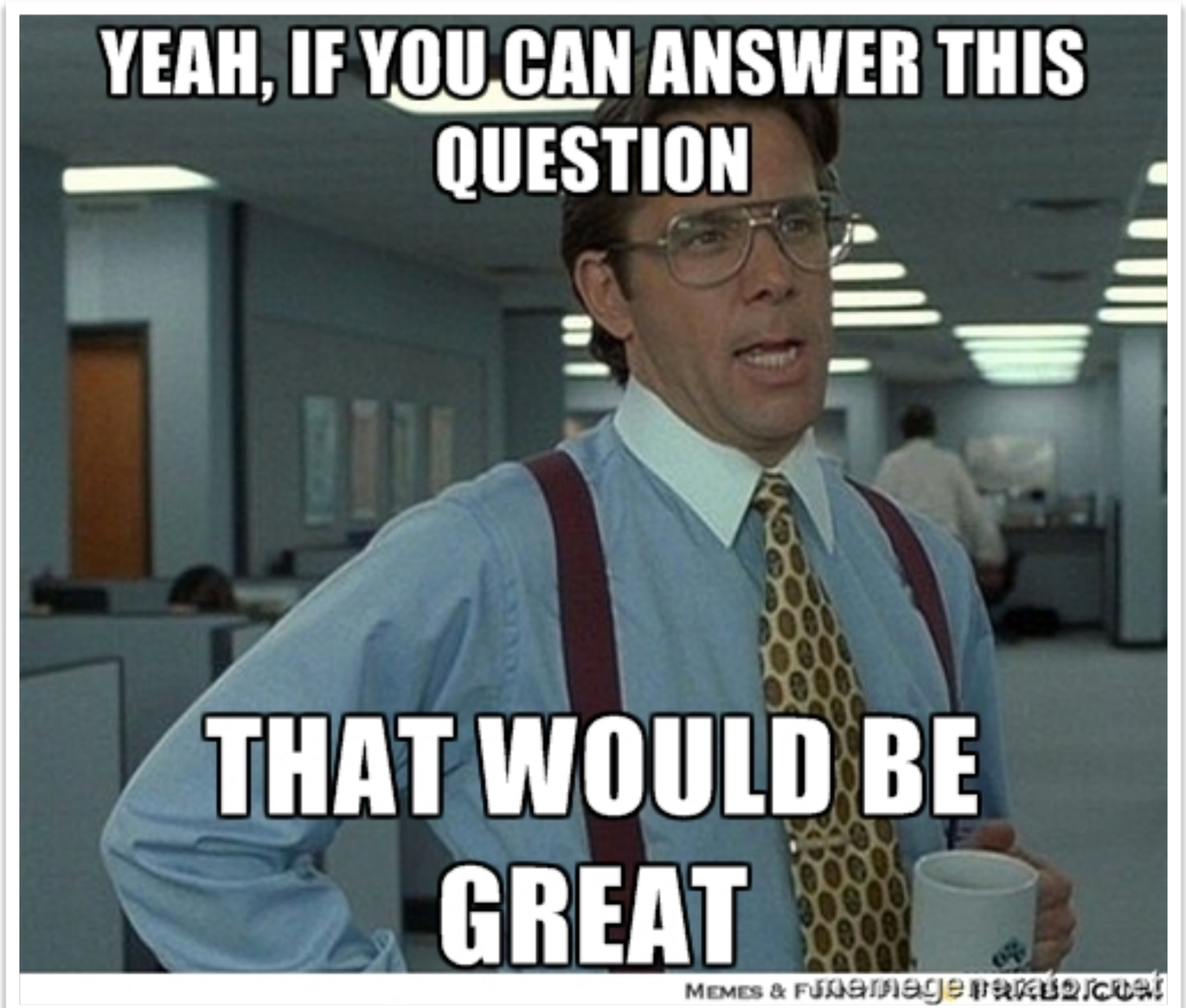
Which instances of recommendations service are available now?

available now?

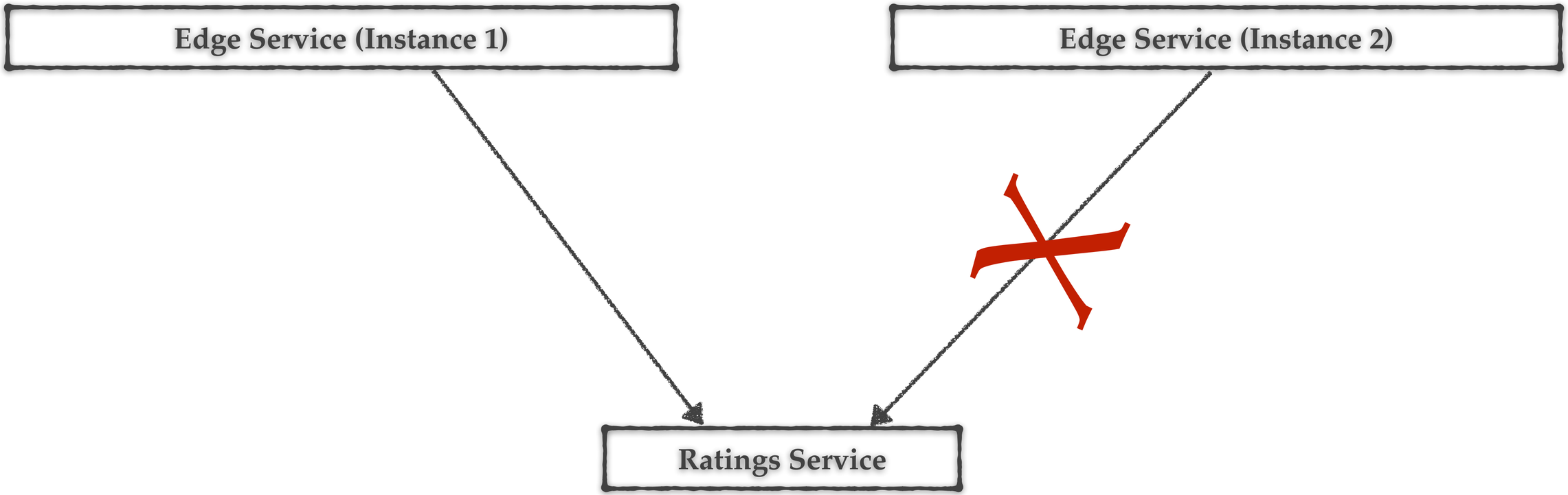

I am a recommendations service instance.
@Startup
I am not a recommendations service instance.
@Shutdown
Recs Service Instance
Service discovery


I am a recommendations service instance.
@Startup
I am alive.
I am not a recommendations service instance.
@Shutdown
Recs Service Instance
Service discovery

available now?



Network Partitions



Network Partitions

Being **available** in a distributed system, is very **subjective**.

Network Partitions

Being **available** in a distributed system, is very **subjective**.

What is **available** to one can be **unavailable** to other.

Network Partitions

Being **available** in a distributed system, is very **subjective**.

What is **available** to one can be **unavailable** to other.

A node's **availability** decision is best when it is **local**.

Service discovery does not guarantee node availability.

Why should Service discovery store node status?

Why should Service discovery store node status?

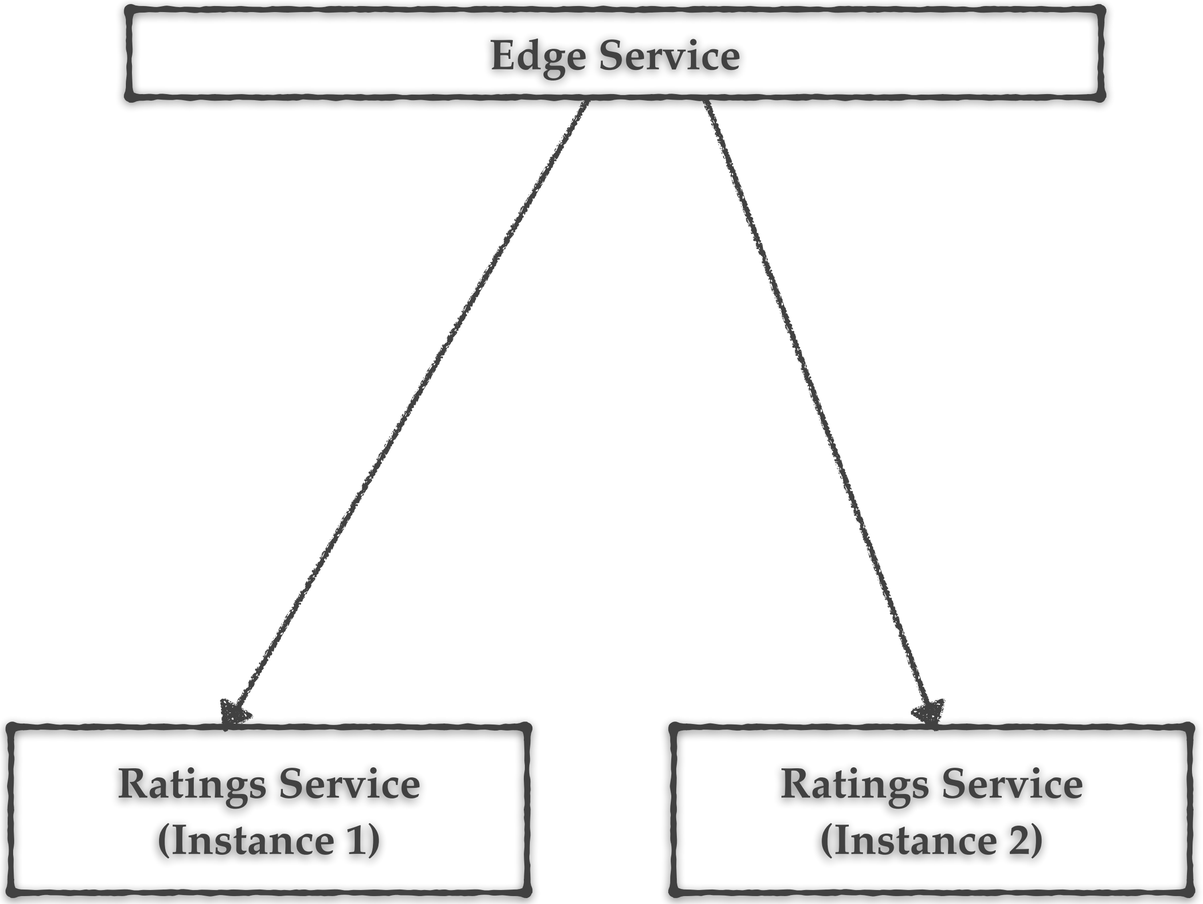
Availability is doubtful but Unavailability can be trusted.

Unavailability can be trusted

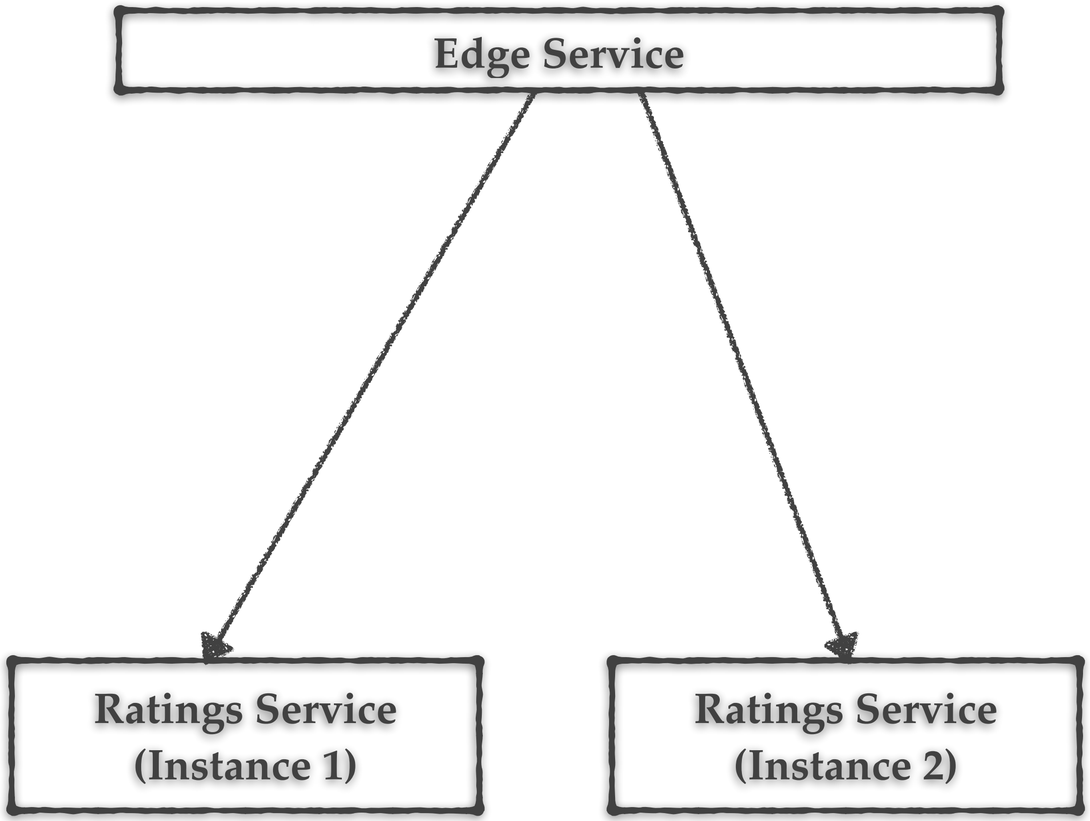
Unavailability can be trusted

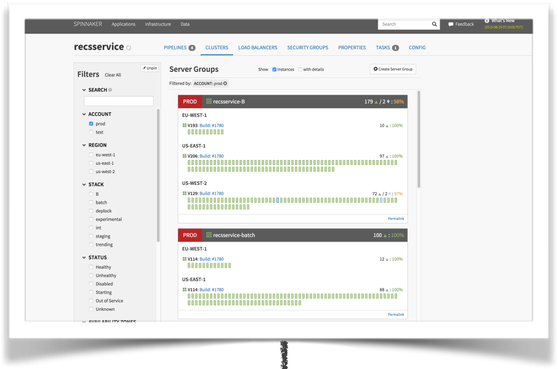
Node status **override**

Node status override



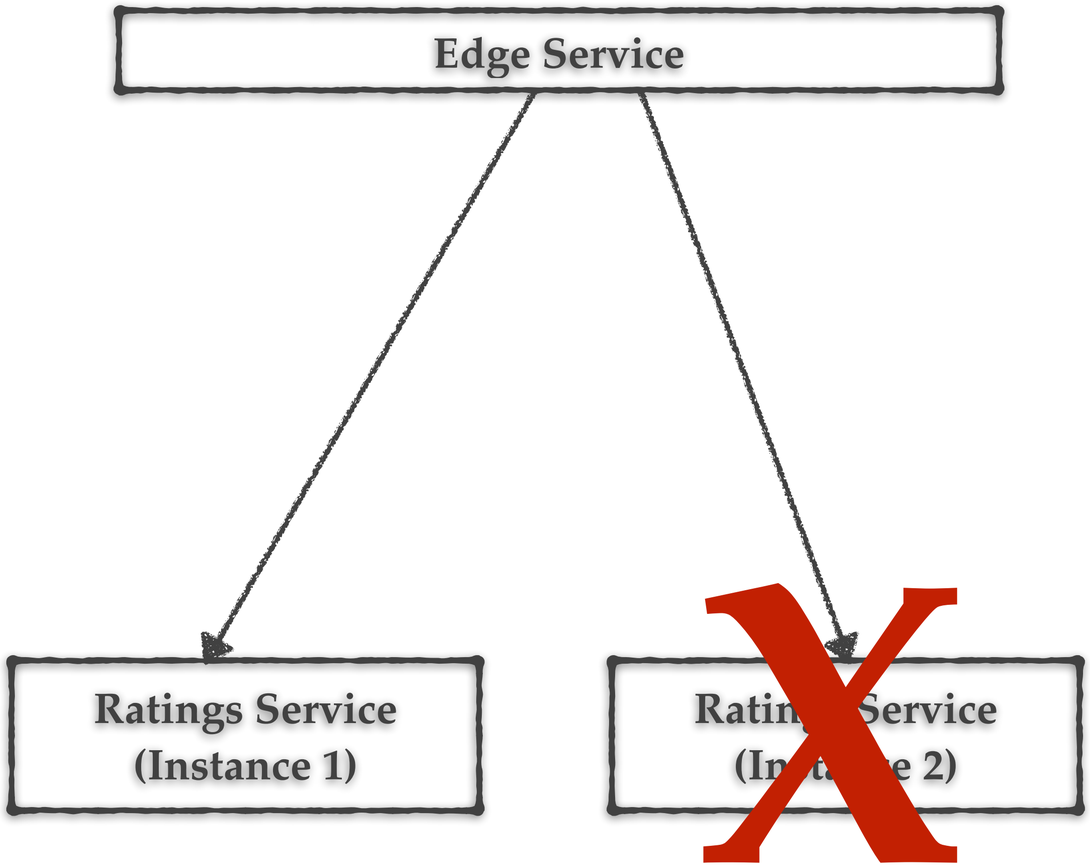
Node status override

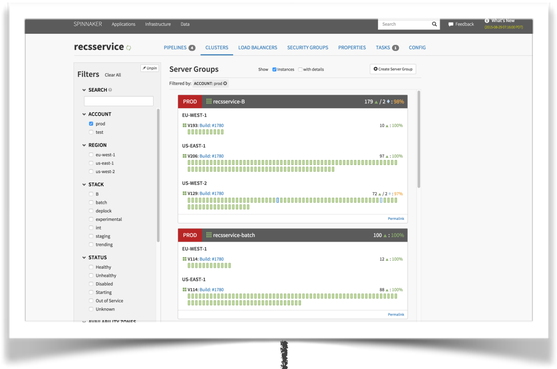




Take Instance 2 out of service

Node status override





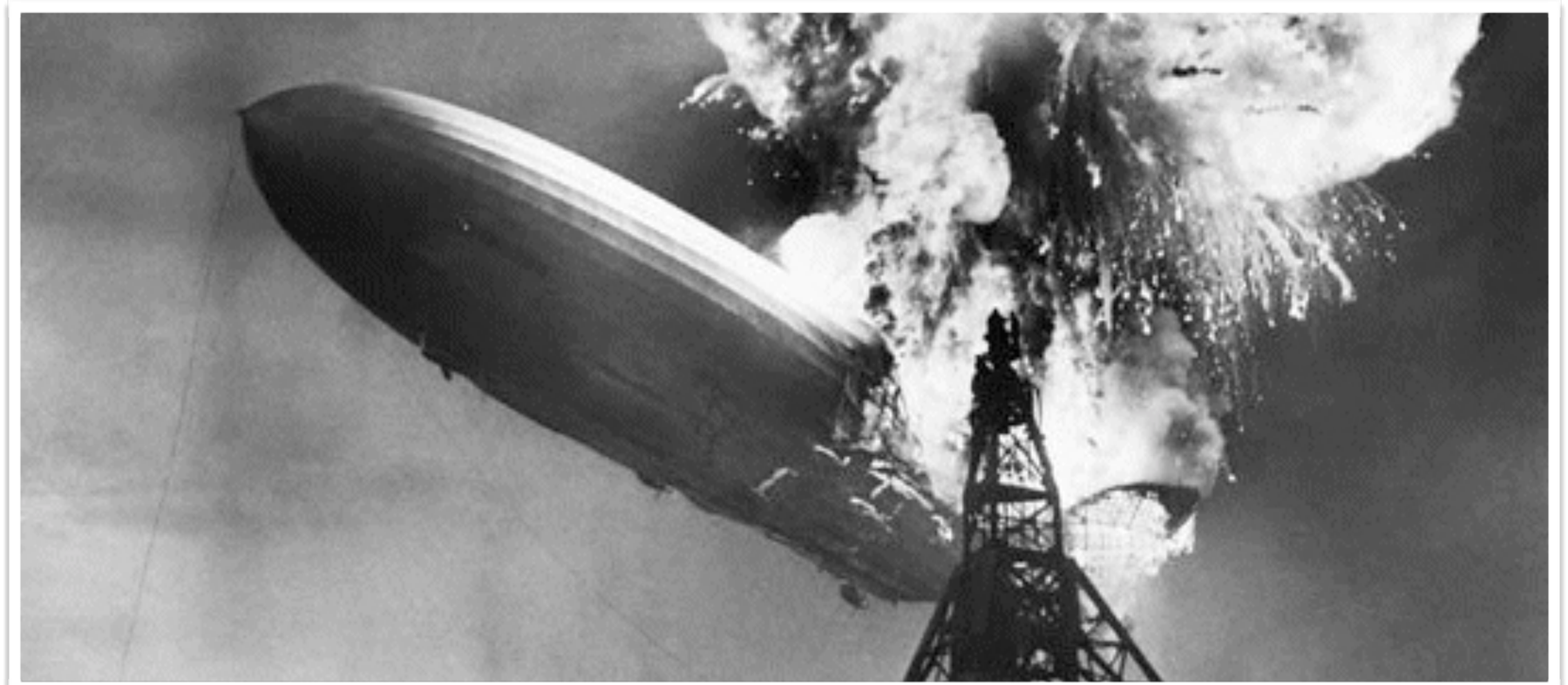
Take Instance 2 out of service

Service Discovery

Nodes can be isolated for debugging.

Nodes can be started in isolation.

Service discovery controls visibility of nodes but does **not** guarantee availability.



Failures

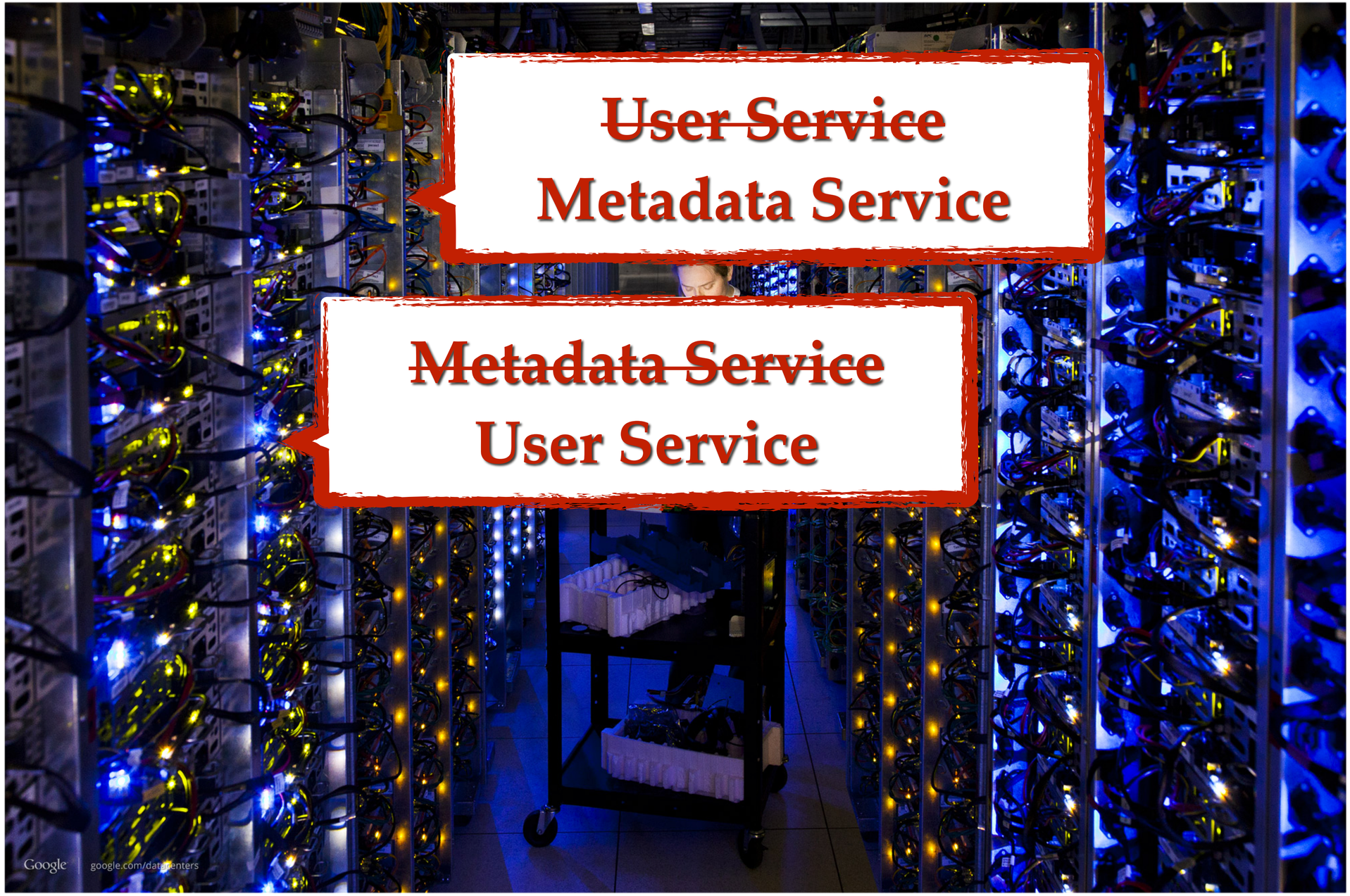
What if?

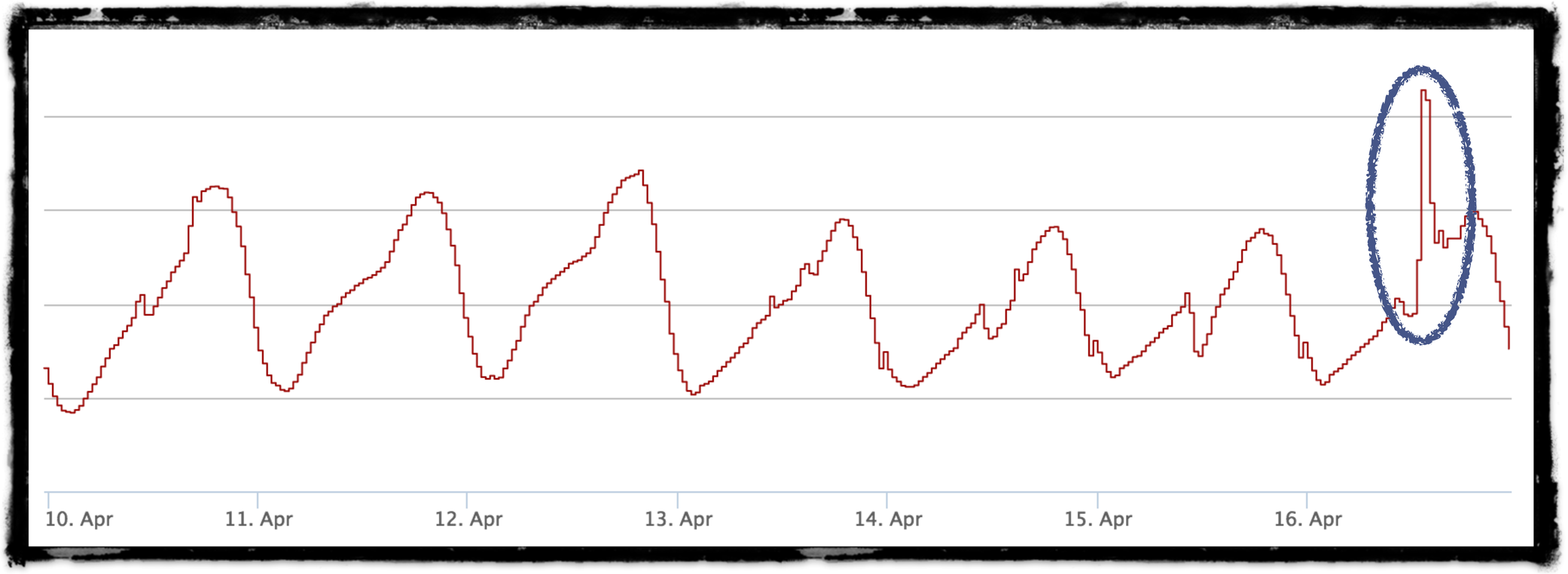
What if …

Service discovery is unavailable?

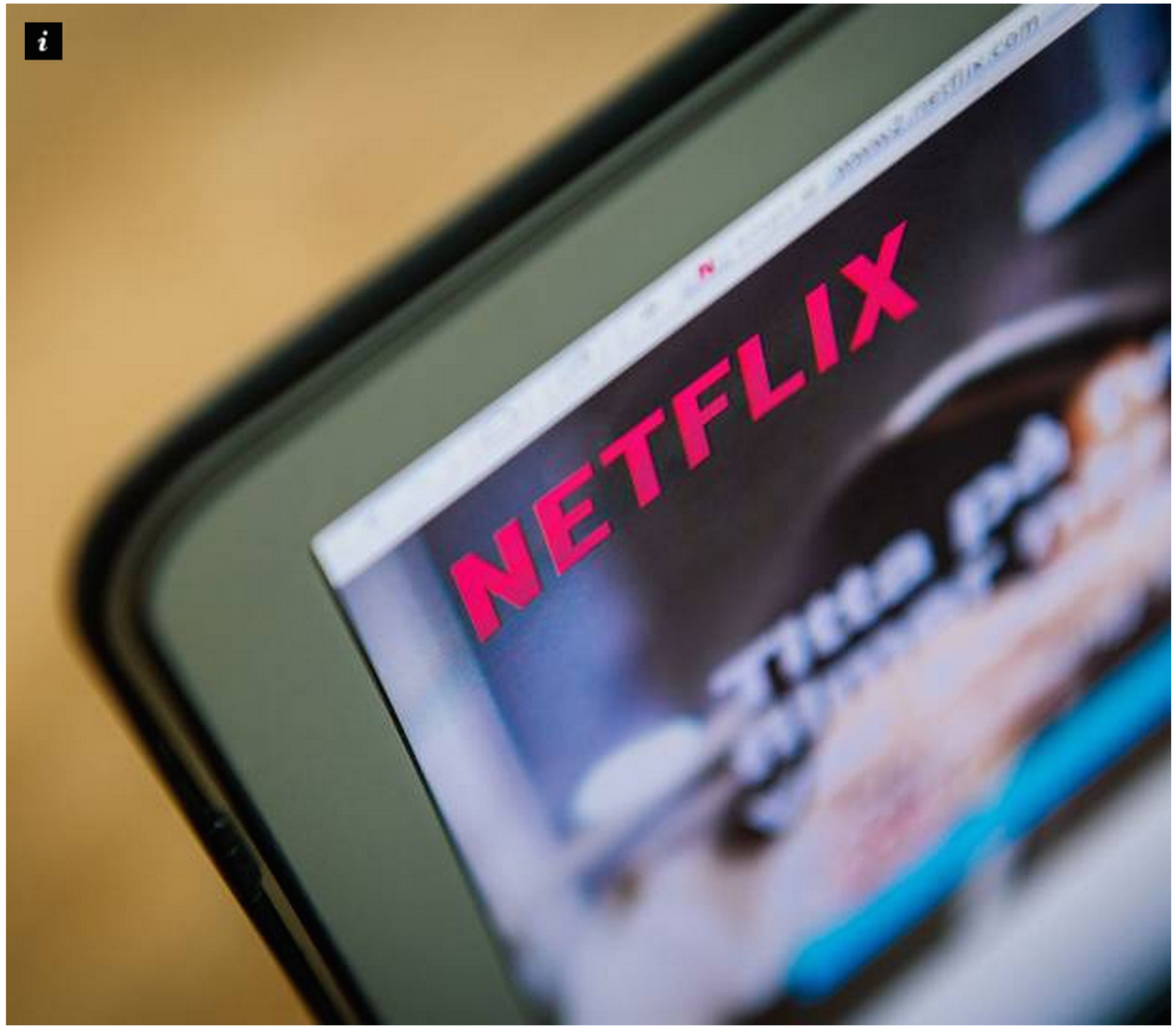
Service discovery controls visibility of nodes but does **not** guarantee availability.

All you lose is the visibility of new nodes.


~~User Service~~
Metadata Service
~~Metadata Service~~
User Service
Google
google.com/datacenters


10. Apr
11. Apr
12. Apr
13. Apr
14. Apr
15. Apr
16. Apr

Netflix 'down': Users complain video streaming site is 'not working'




HE HE HE

High **availability** is a strong **requirement**.

CAP theorem

Visibility

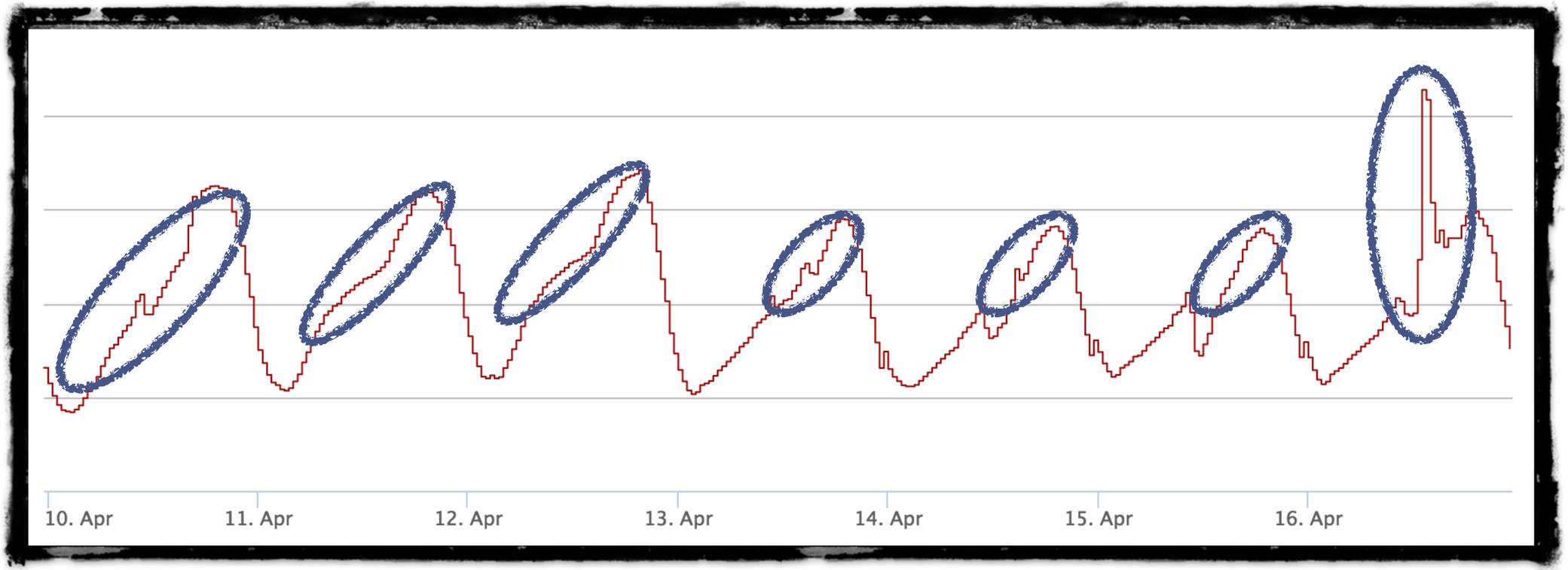
Consistency

Availability

Visibility

Consistency

A



Worst time

Visibility

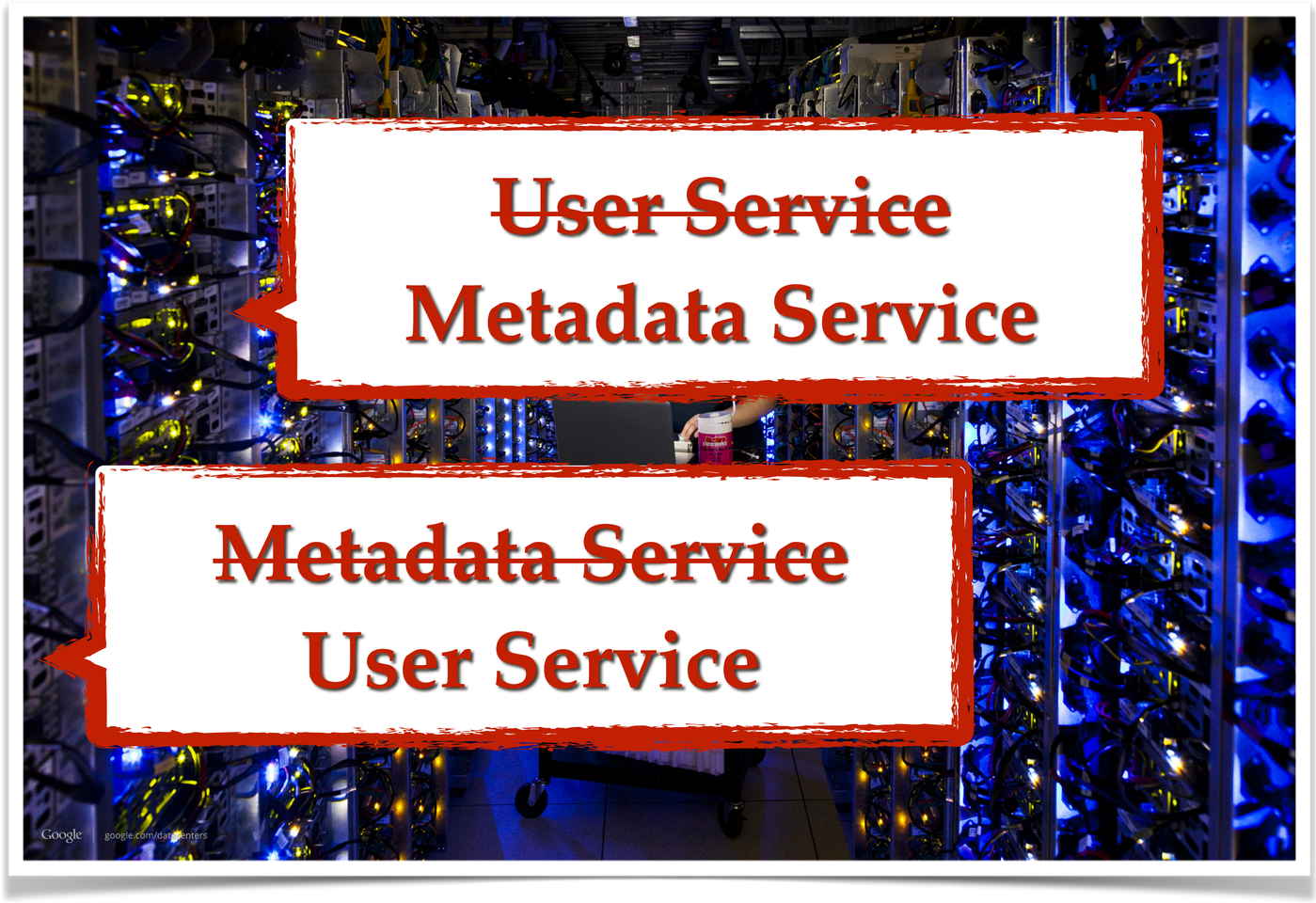
Consistency

Availability

Visibility

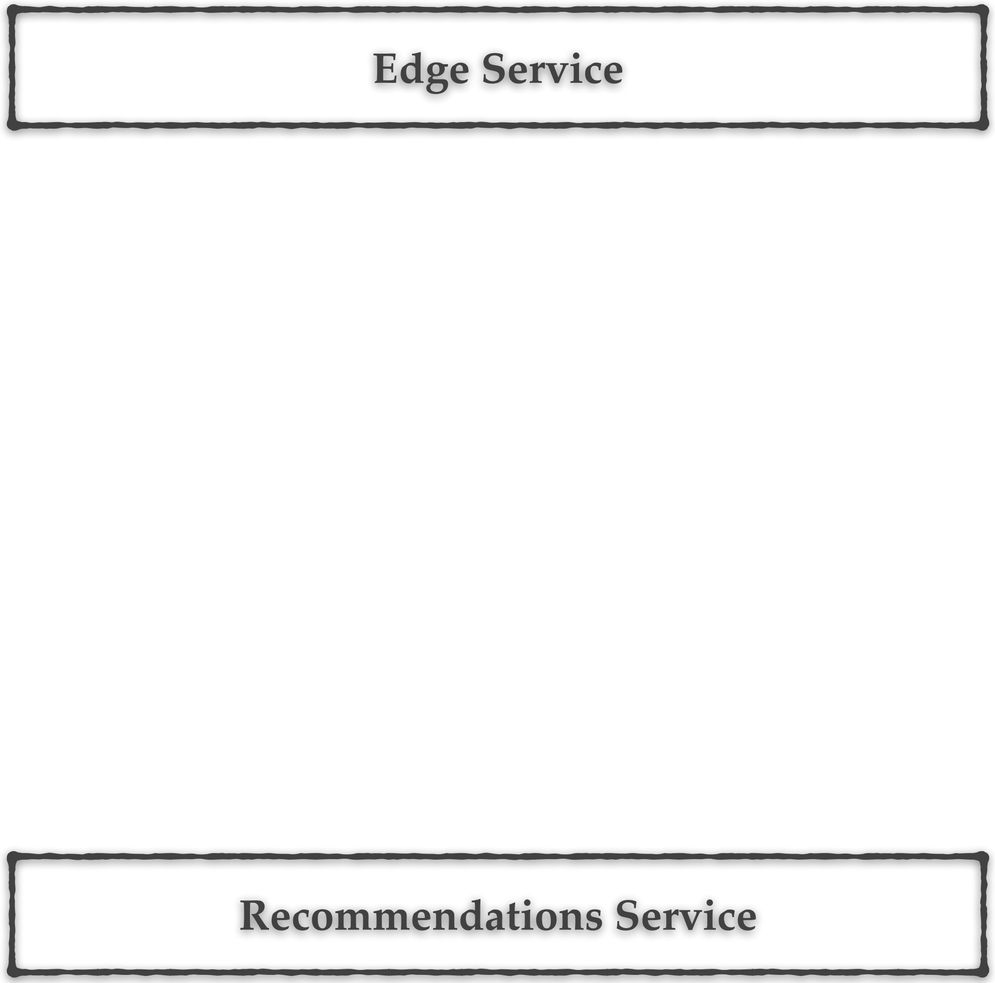
C

Availability


~~User Service~~
Metadata Service
~~Metadata Service~~
User Service
Google
google.com/datacenters

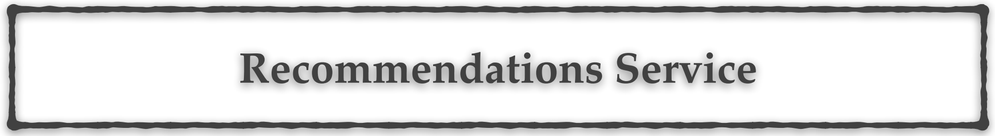
Worst case

A persistent, verified connection based interaction solves this issue.


Edge Service
Connect
Recommendations Service

Connect

Verify
Are you a Recs service instance?

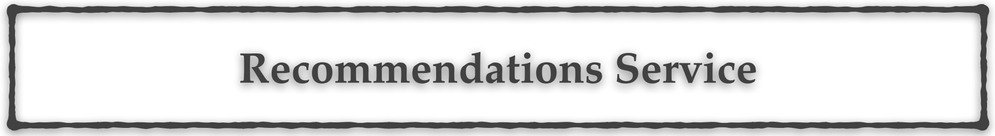


Connect

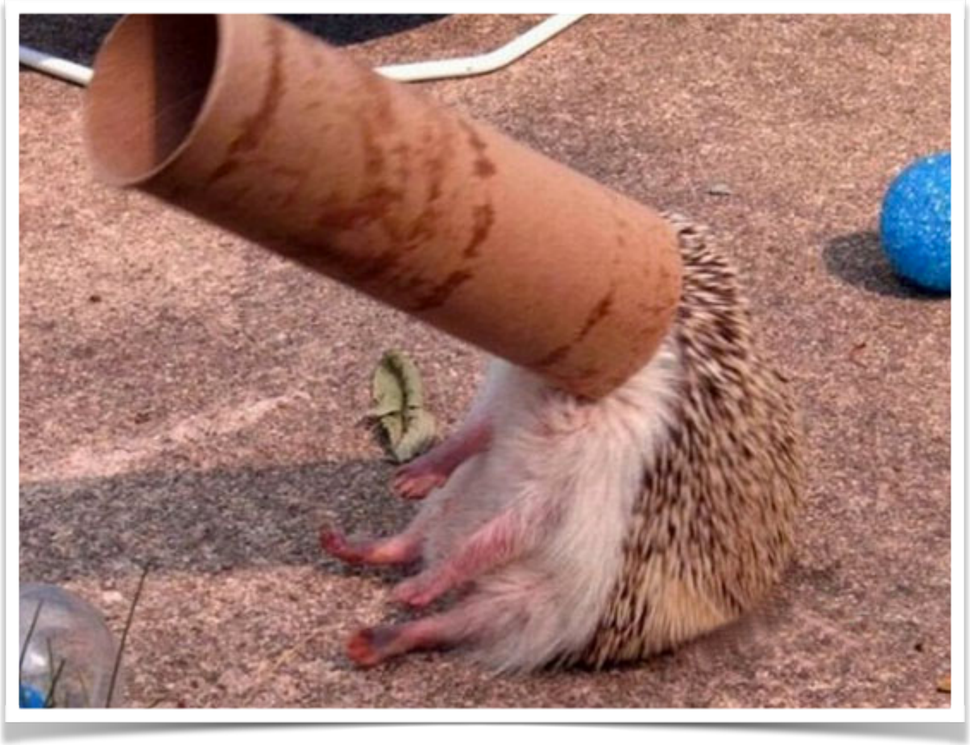
Verify
Are you a Recs service instance?

Requests

.
.
.



In most cases choosing availability over consistency for Service Discovery is the correct thing to do.



Durability

Service discovery data is
ephemeral and **re-generatable**.

Ephemeral


I am a recommendations service
@Start
In dynamic environments, this duration is in order of hours or days.
I am not a recommendations service
@Shutdo
Recs Service Instance
Service discovery

Re-generatable data

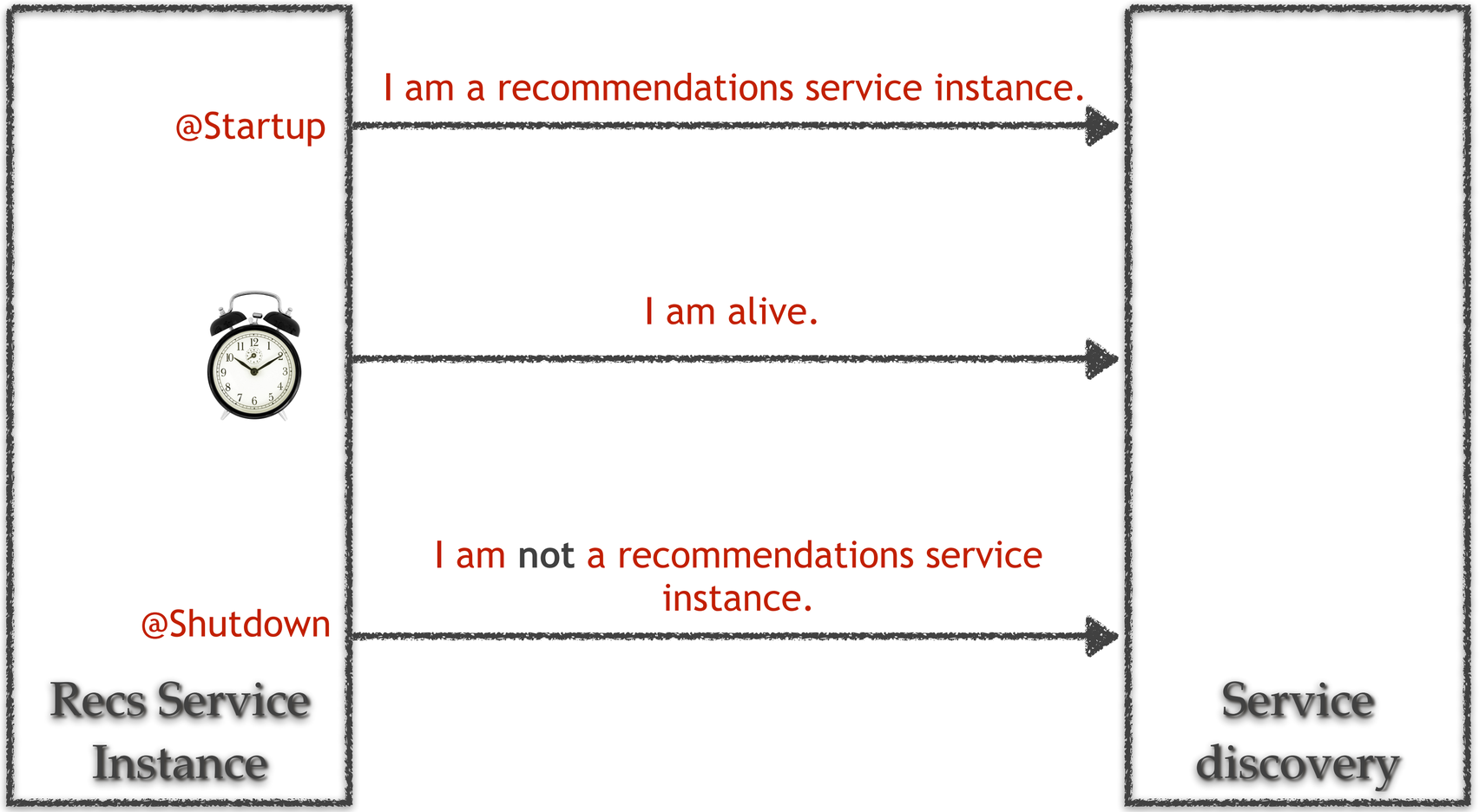
Service Discovery data is the **instance information**.
It is **always available** to the instance.


node-categorization
re-generatable
availability
heartbeats
ephemeral
node-availability-out-of-scope
WordItOut

Can we create a

co-ordination free service discovery?

Stateful Client-Server interaction



Stateful Client-Server interaction

Needs a stateful protocol.


I am a recommendations service instance.
@Startup
I am alive.
A connection oriented, ordered and reliable protocol.
I am not a recommendations service instance.
@Shutdown
Recs Service Instance
Service discovery

Nice Benefits

Causal ordering

Nice Benefits

Causal ordering

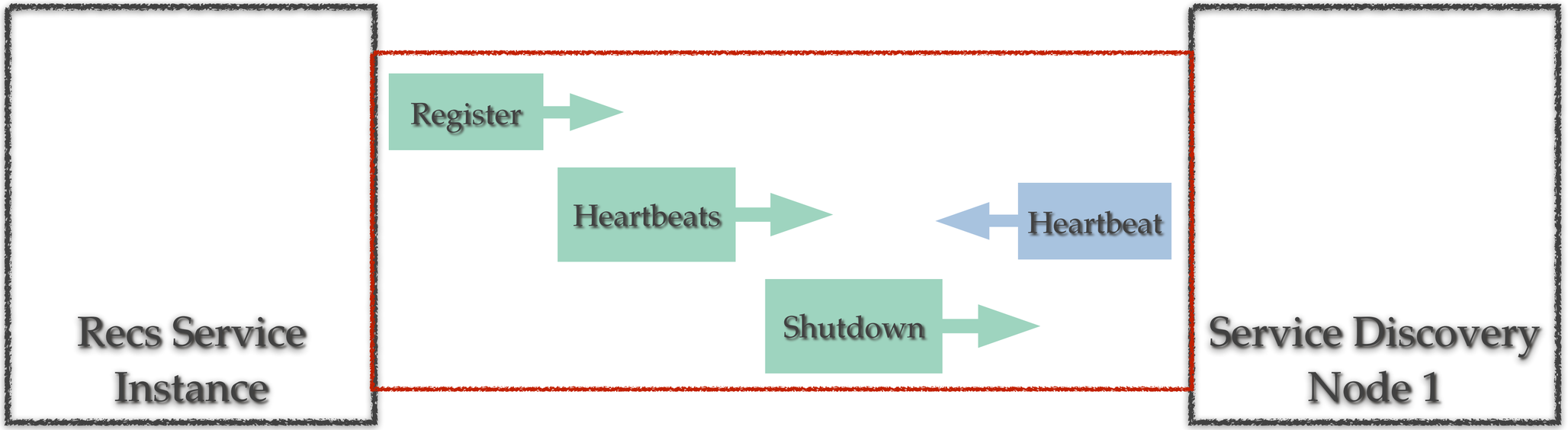
Natural lifecycle (connection)

Nice Benefits

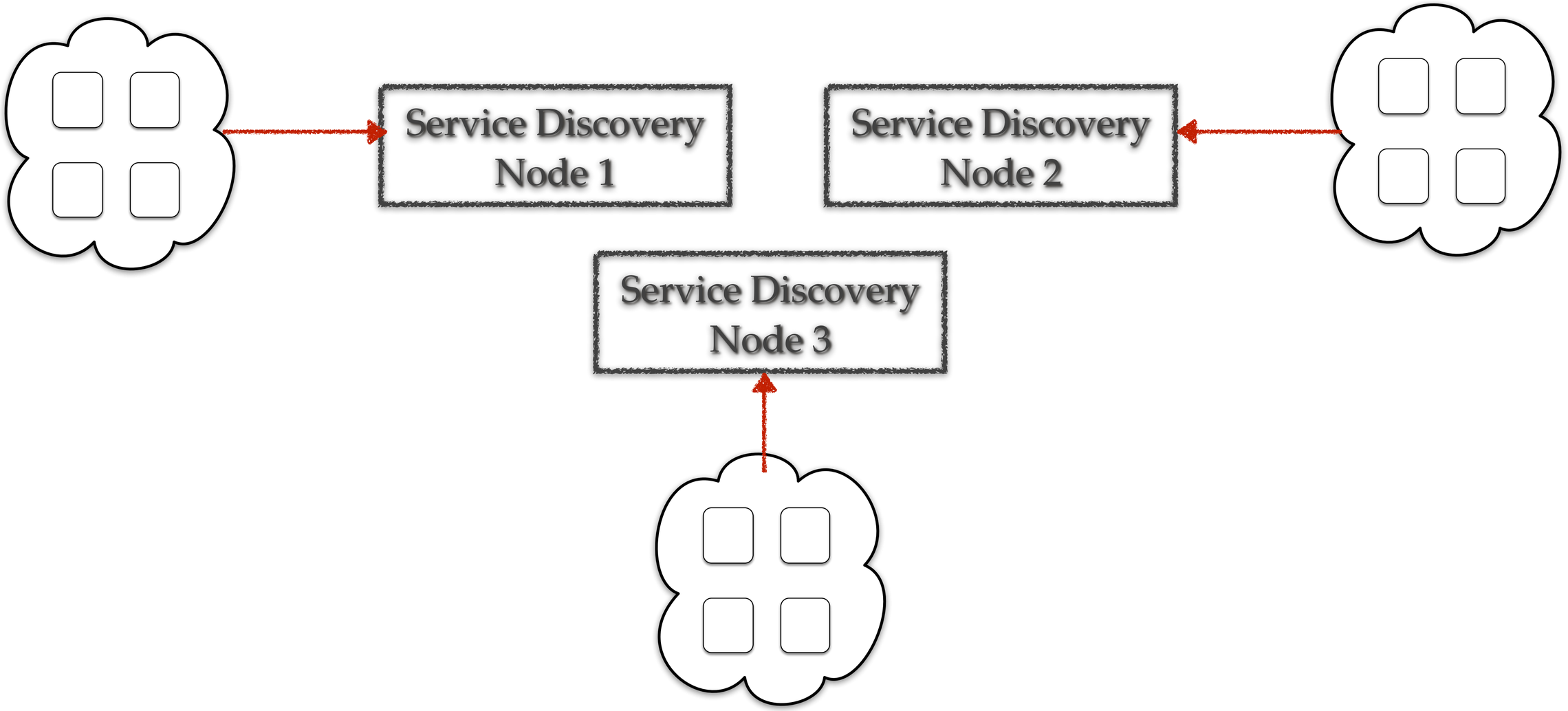
Causal ordering

Natural lifecycle (connection)

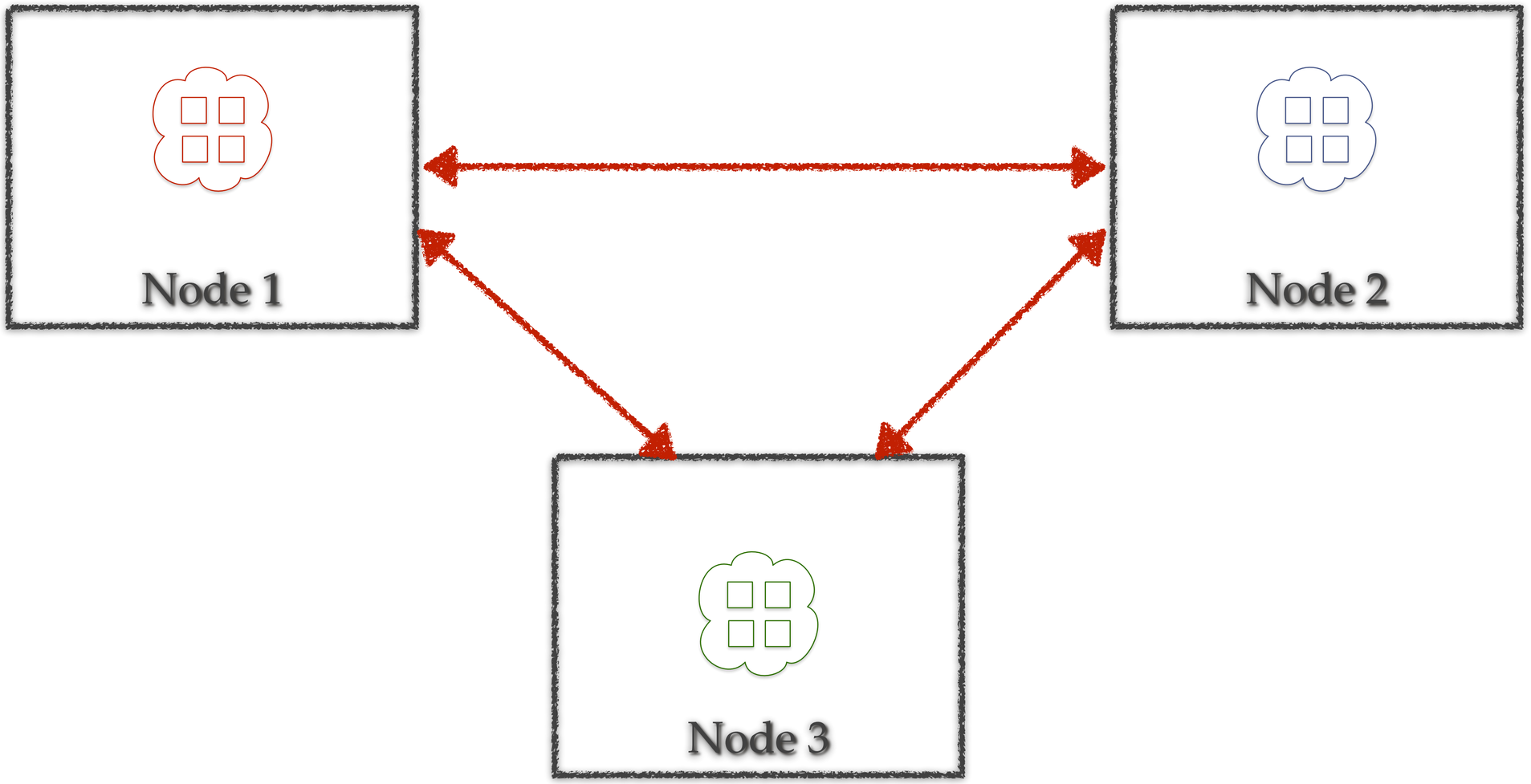
Ability to send data“diff”.


Recs Service Instance
Register
Heartbeats
Heartbeat
Shutdown
Service Discovery Node 1

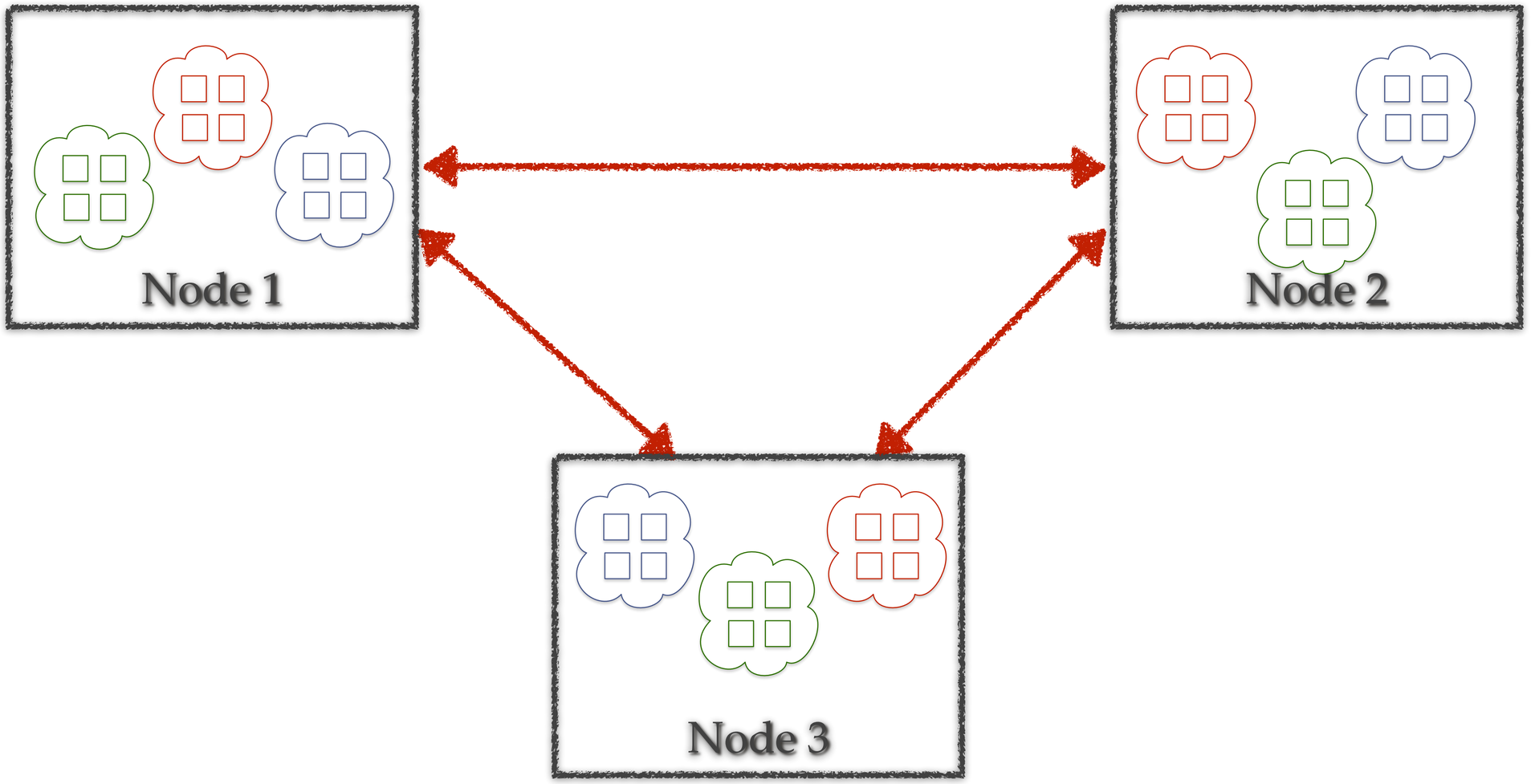
Bigger Picture

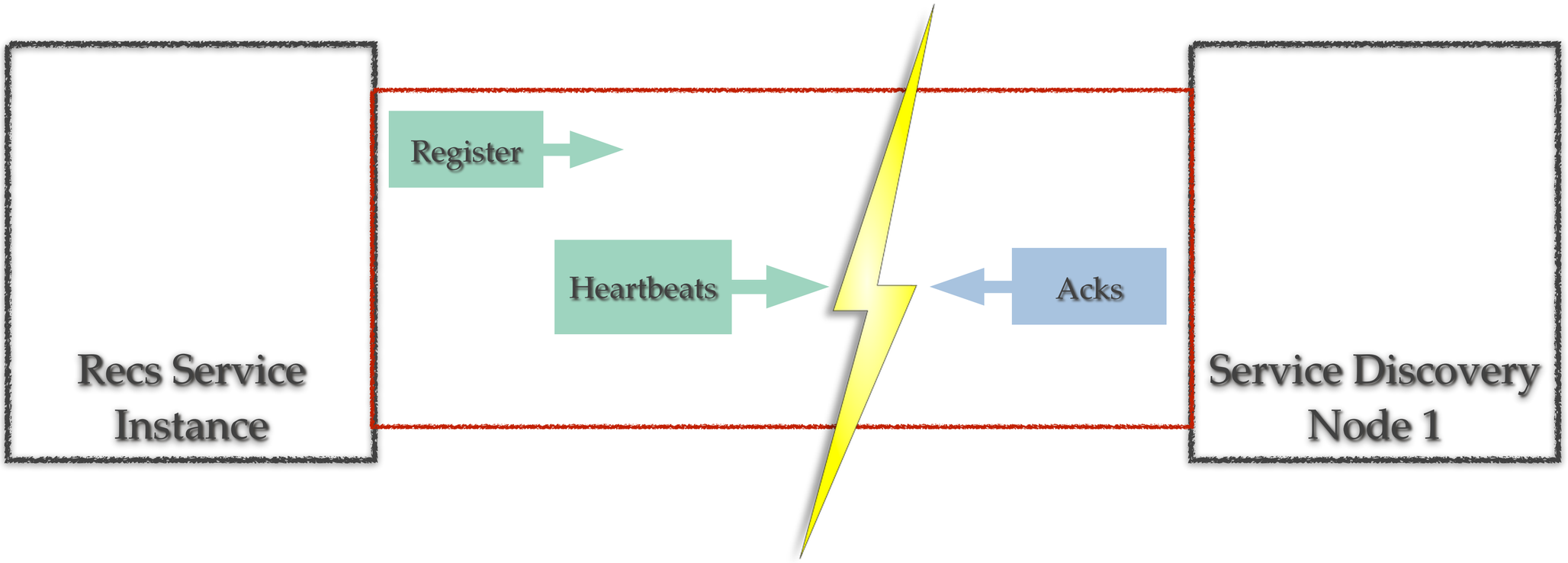


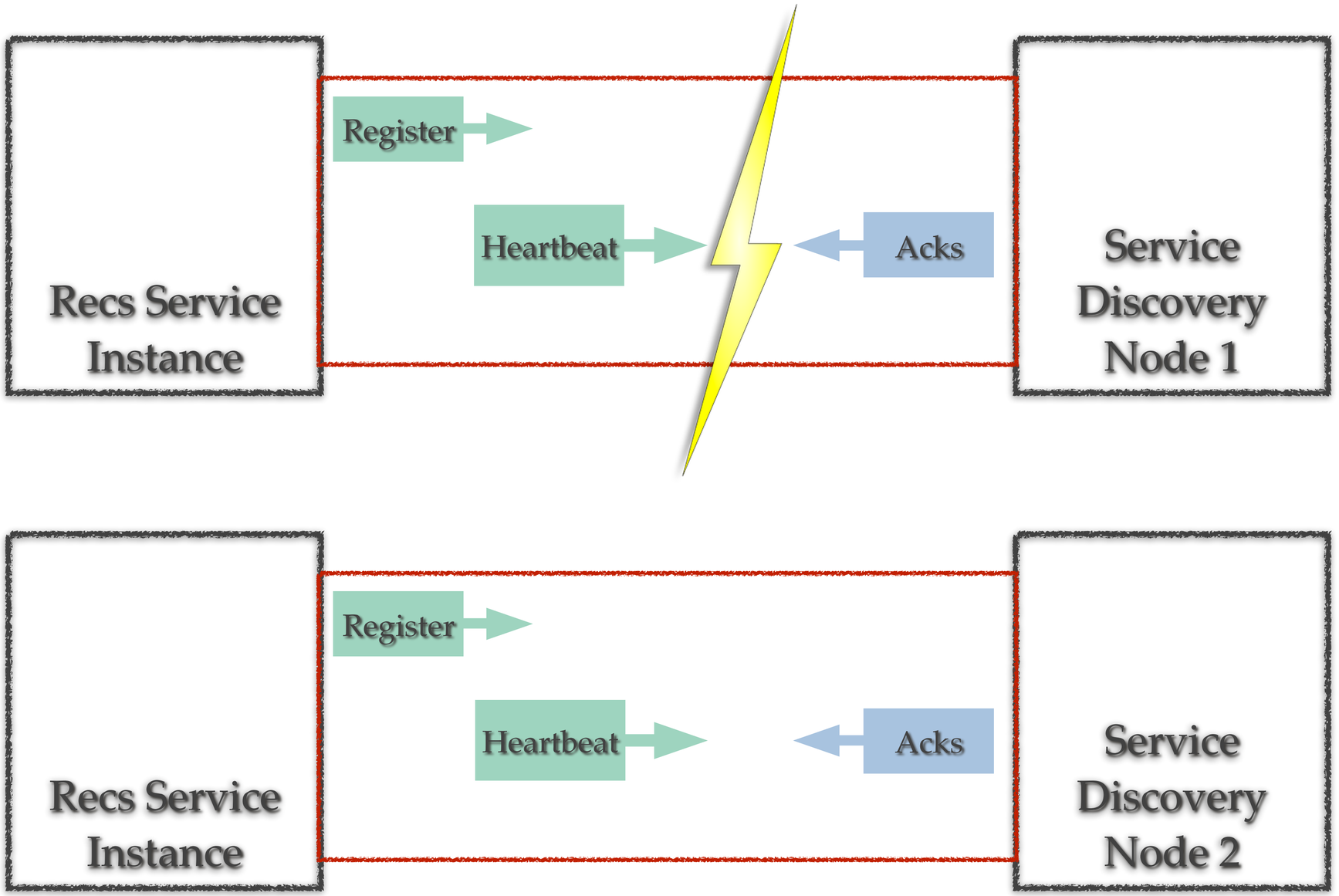
Replication

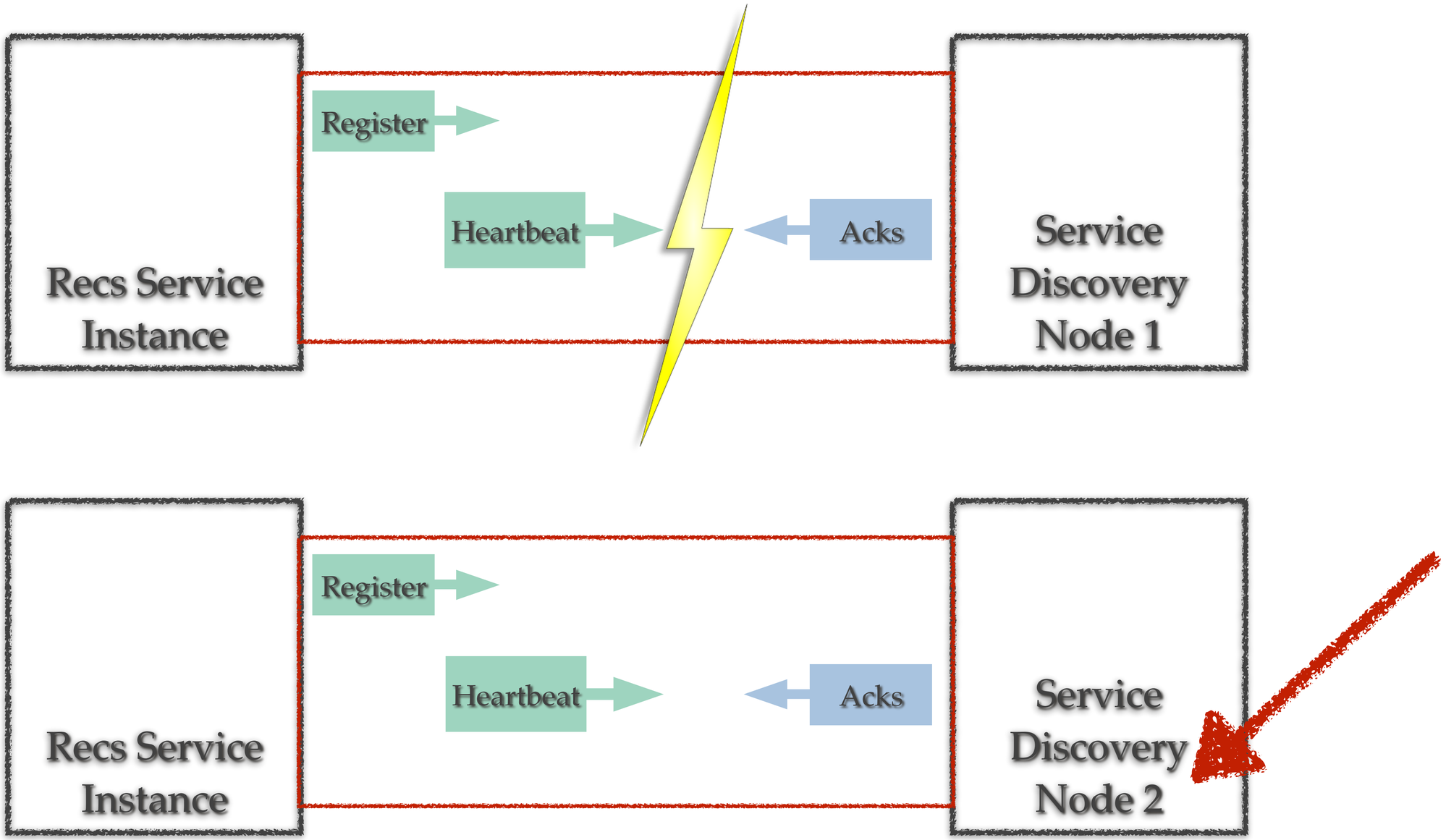


Replication



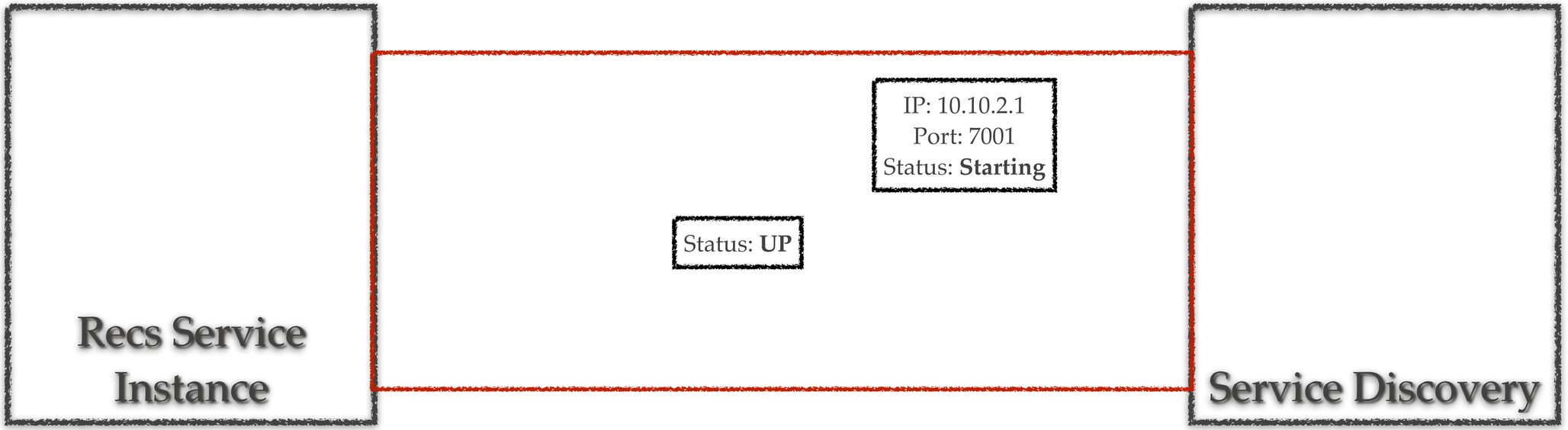

Recs Service Instance
Register
Heartbeats
Acks
Service Discovery Node 1


Recs Service Instance
Register
Heartbeat
Acks
Service Discovery Node 1
Recs Service Instance
Register
Heartbeat
Acks
Service Discovery Node 2

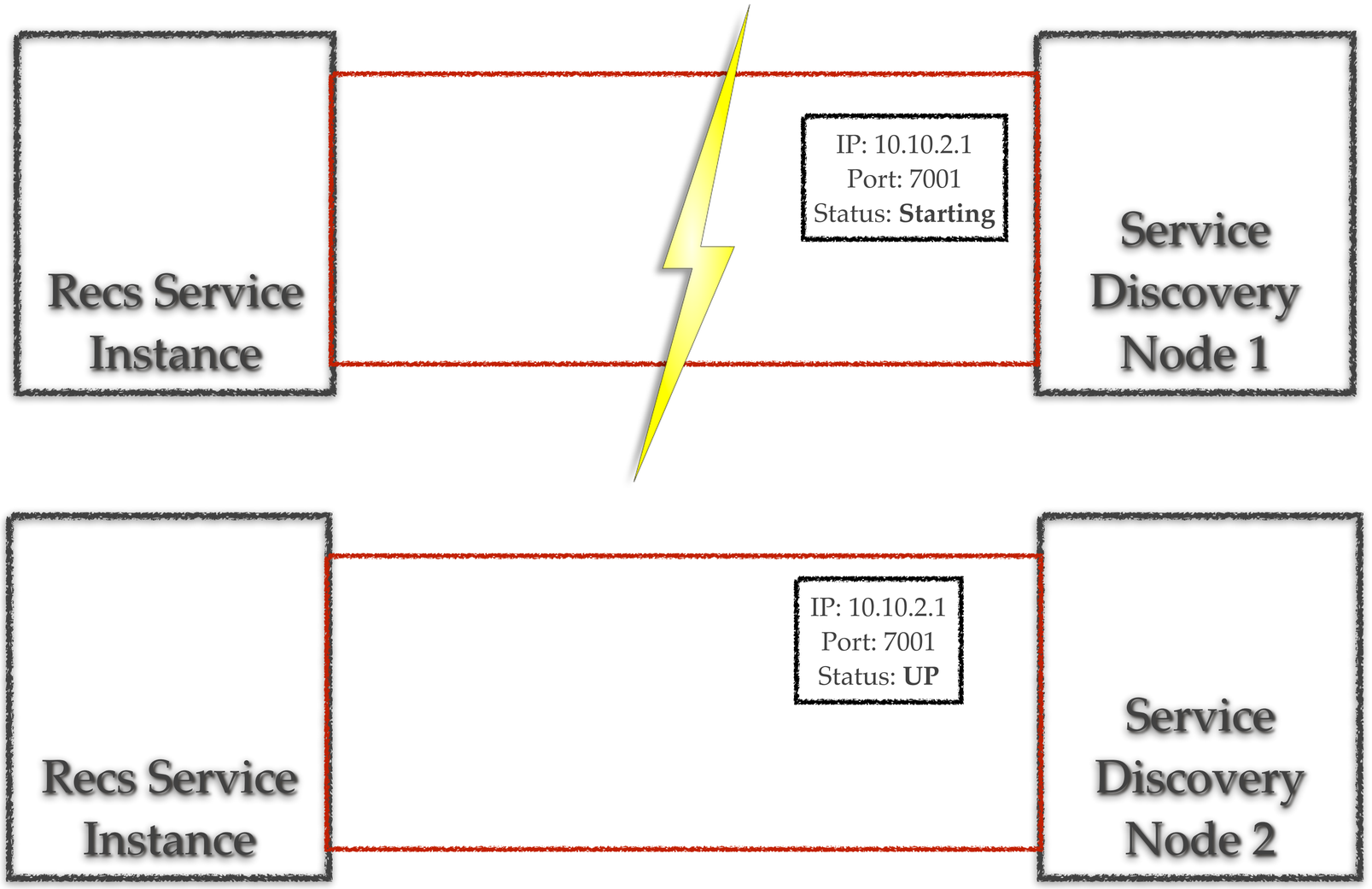

Recs Service Instance
Register
Heartbeat
Acks
Service Discovery Node 1
Recs Service Instance
Register
Heartbeat
Acks
Service Discovery Node 2

Data conflicts

Conflicts


IP: 10.10.2.1
Port: 7001
Status: **Starting**
Status: **UP**
Recs Service Instance
Service Discovery

Conflicts


Recs Service Instance
IP: 10.10.2.1
Port: 7001
Status: Starting
Service Discovery Node 1
Recs Service Instance
IP: 10.10.2.1
Port: 7001
Status: UP
Service Discovery Node 2

Conflicts

IP: 10.10.2.1
Port: 7001
Status: **Starting**

From Service Discovery Node 1

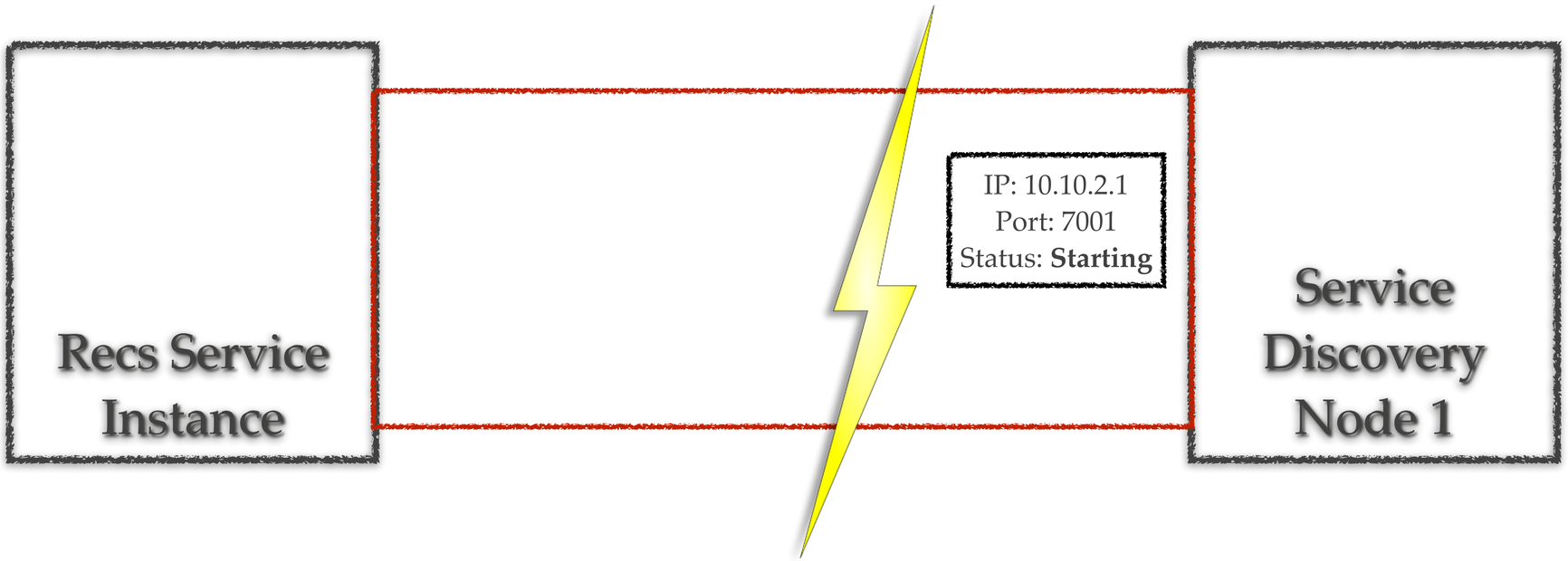
IP: 10.10.2.1
Port: 7001
Status: **UP**

From Service Discovery Node 2

Service Discovery Node 3

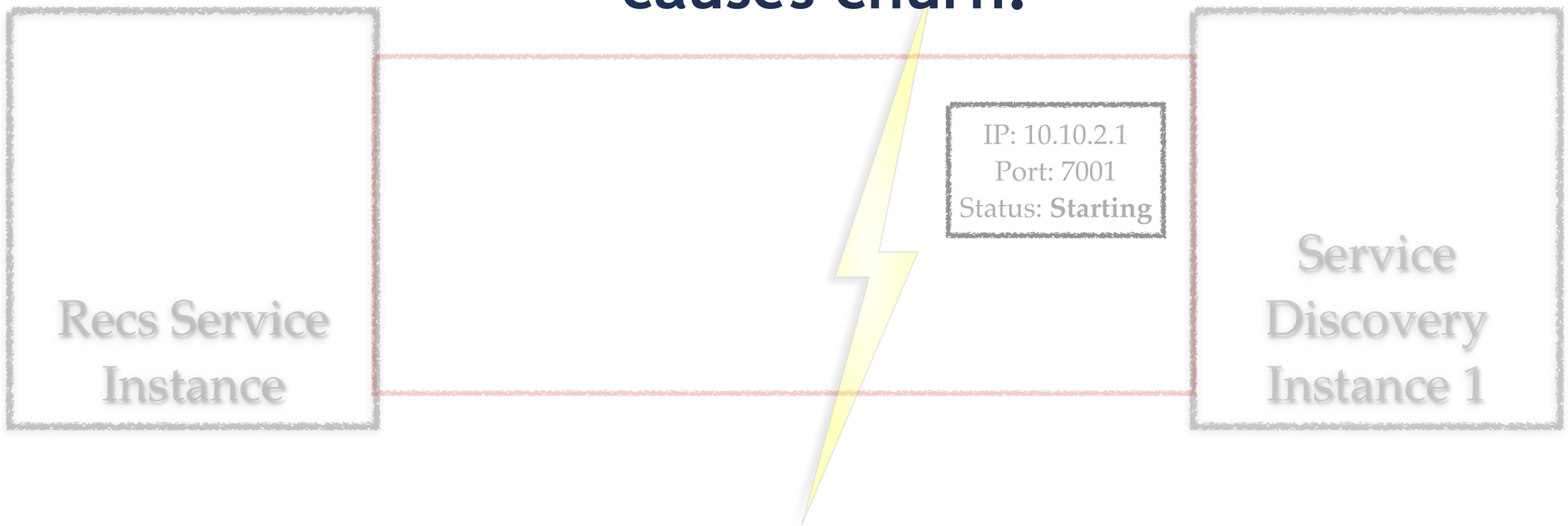
Conflict resolution

What should be the state of Recs Service Instance data now?



What should be the state of Recs Service Instance data now?

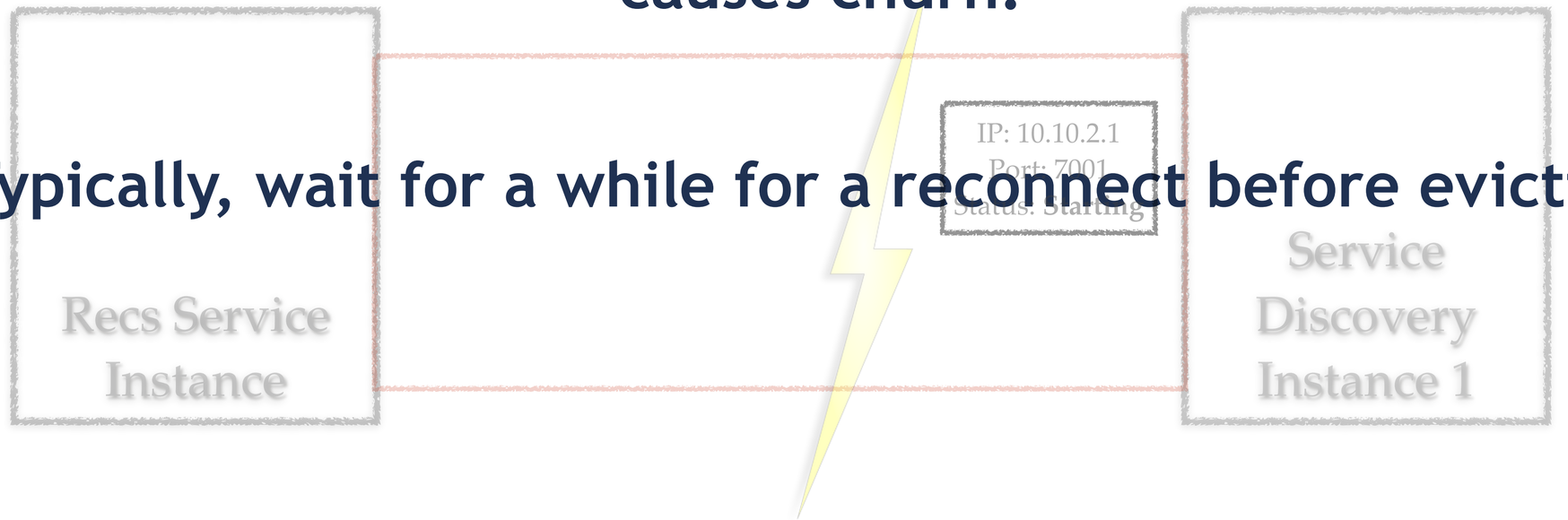
Show tolerance towards broken connections, evicting an instance too early causes churn.



What should be the state of Recs Service Instance data now?

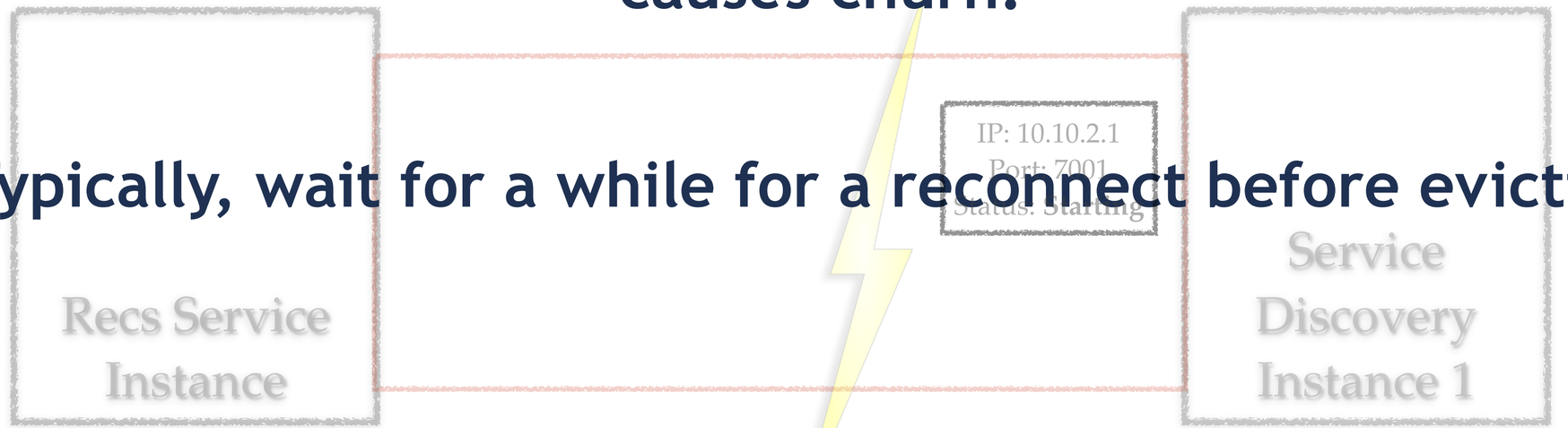
Show tolerance towards broken connections, evicting an instance too early causes churn.

Typically, wait for a while for a reconnect before eviction.



What should be the state of Recs Service Instance data now?

Show tolerance towards broken connections, evicting an instance too early causes churn.

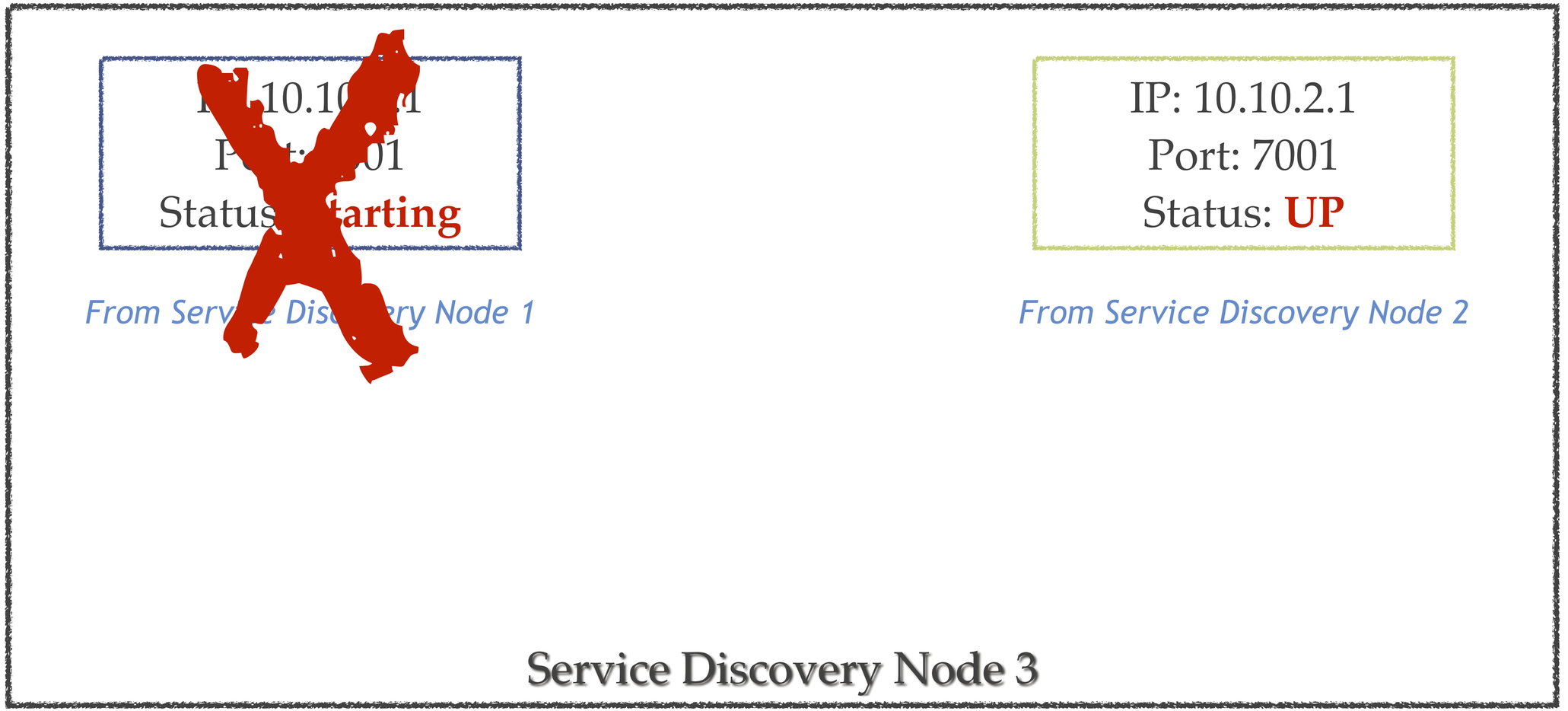


Typically, wait for a while for a reconnect before eviction.

Evict in absence of reconnects after a while.

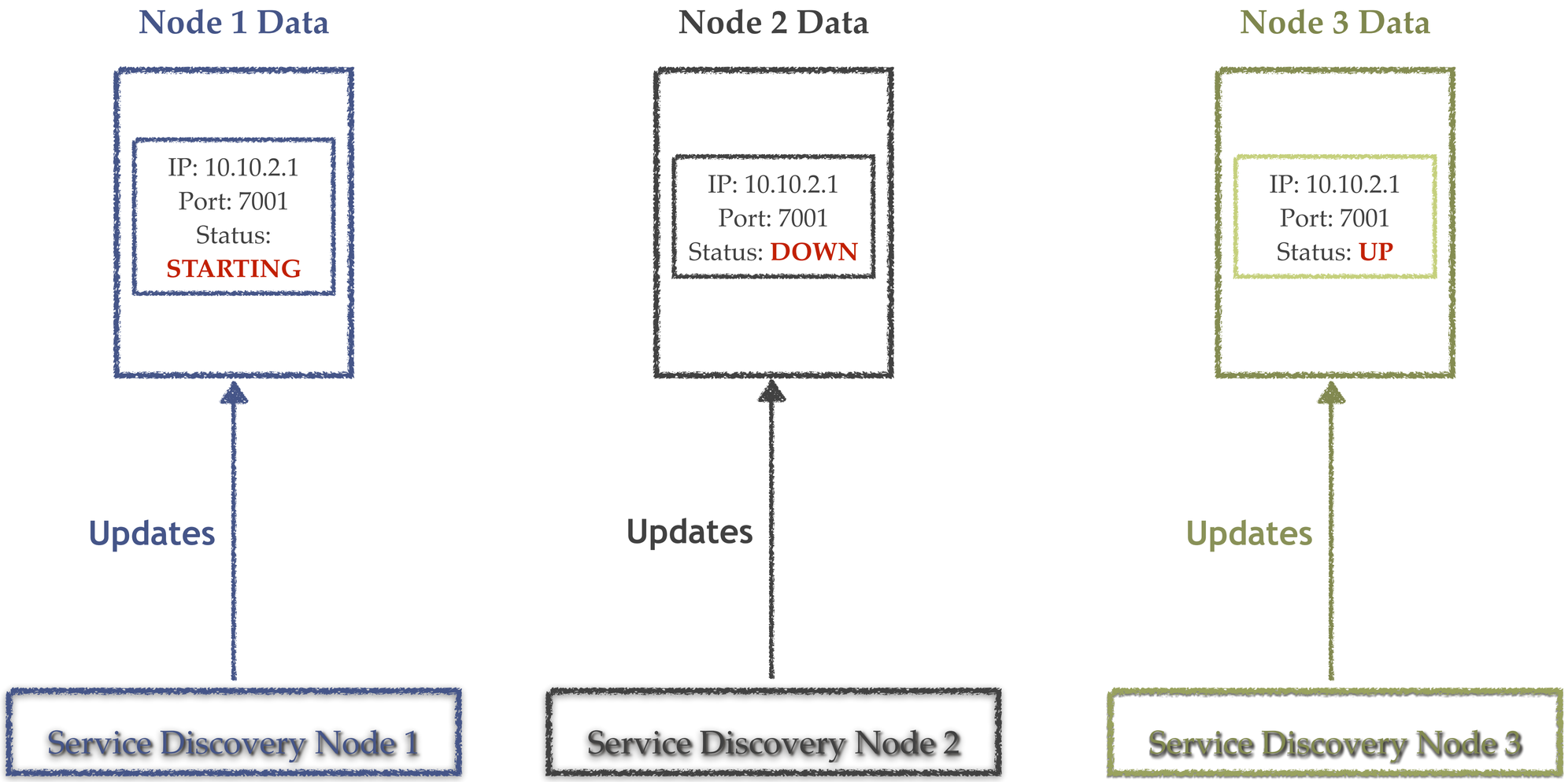
Well behaved clients at **steady state** connects to a **single** Service Discovery **node**.

Conflicts

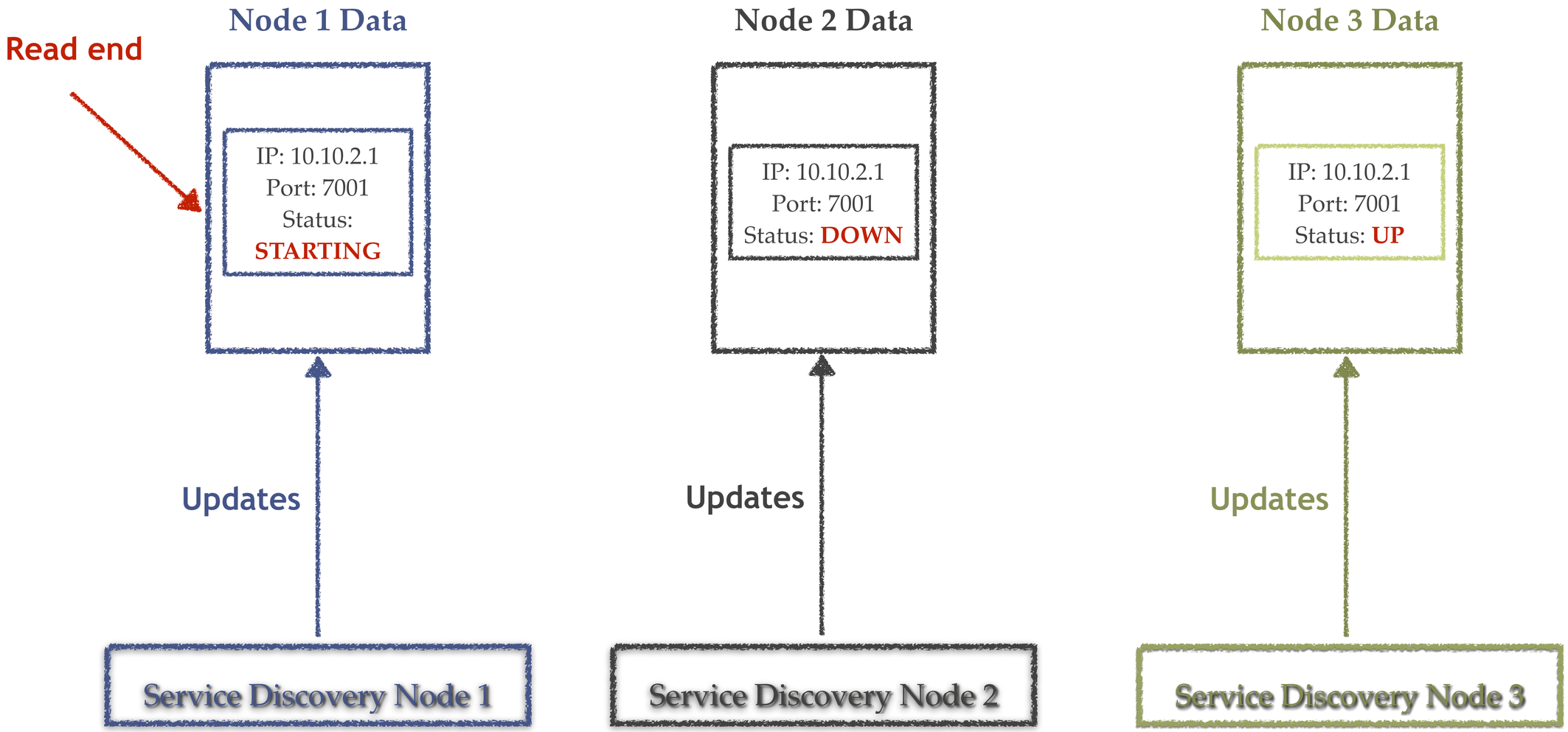

IP: 10.10.[illegible].1
Port: [illegible]7001
Status: Starting
From Service Discovery Node 1
IP: 10.10.2.1
Port: 7001
Status: UP
From Service Discovery Node 2
Service Discovery Node 3

Data conflicts are resolved naturally for service discovery.

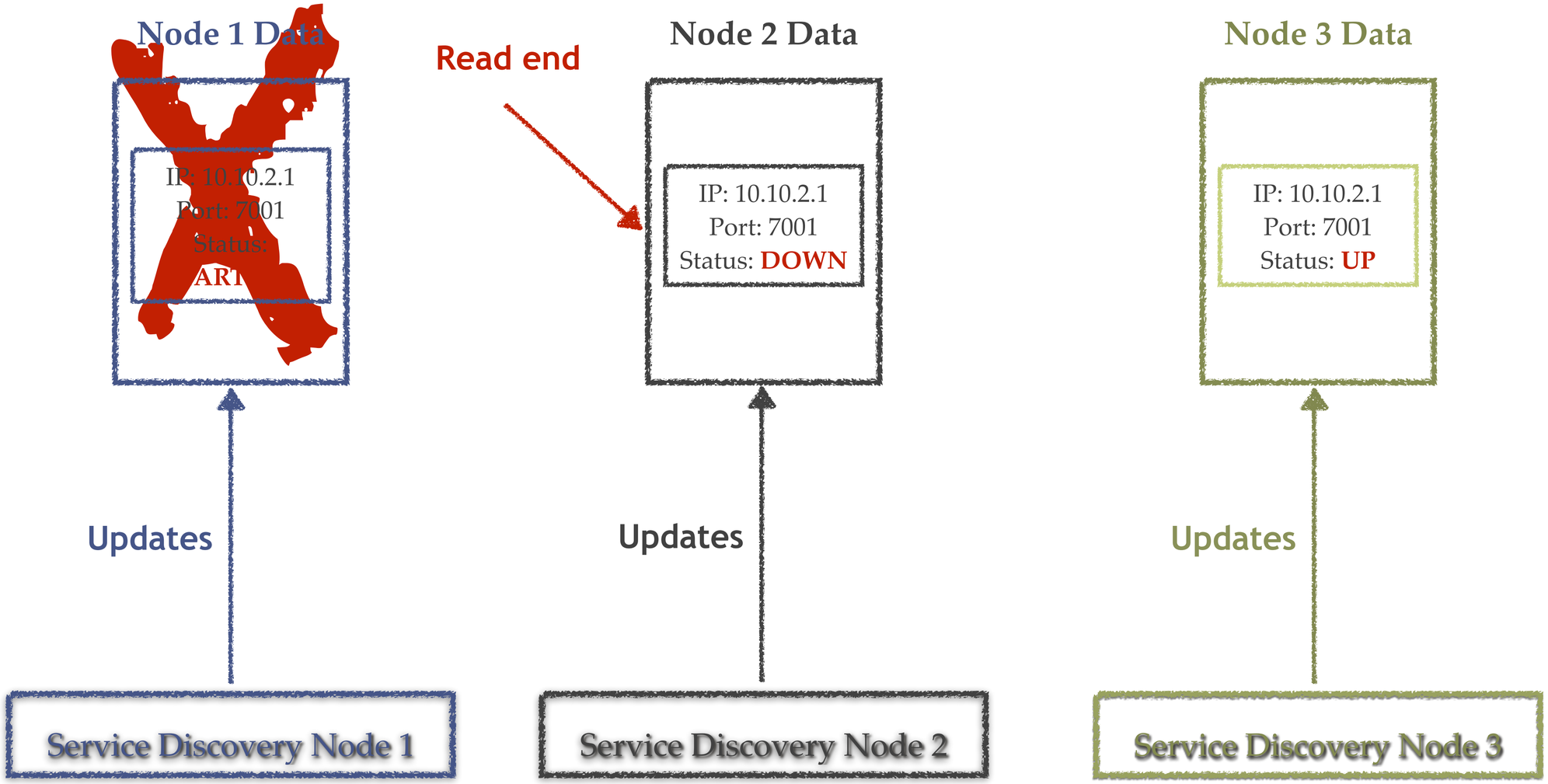
Tolerate temporary conflicts



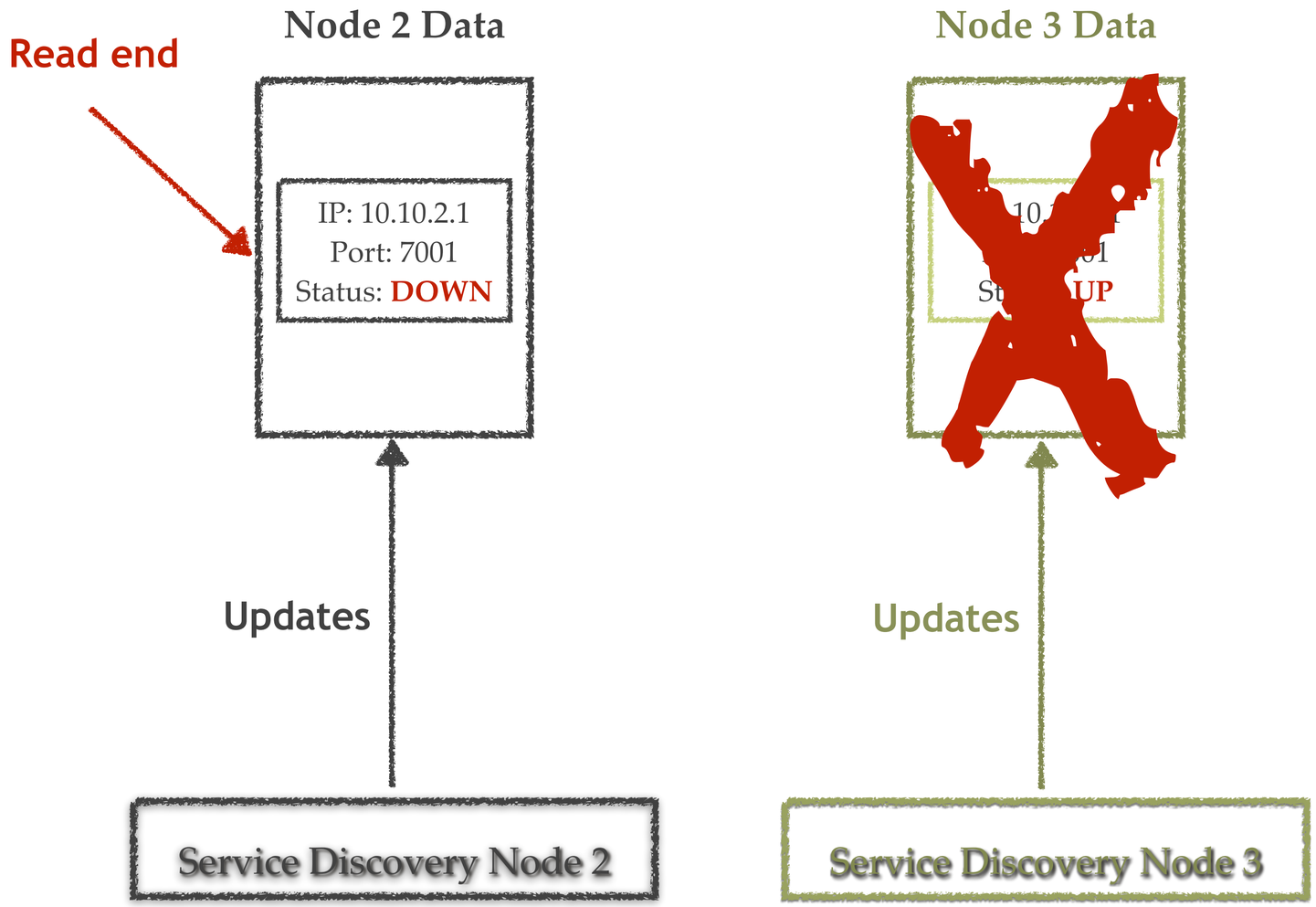
Read from a version (till it is gone)



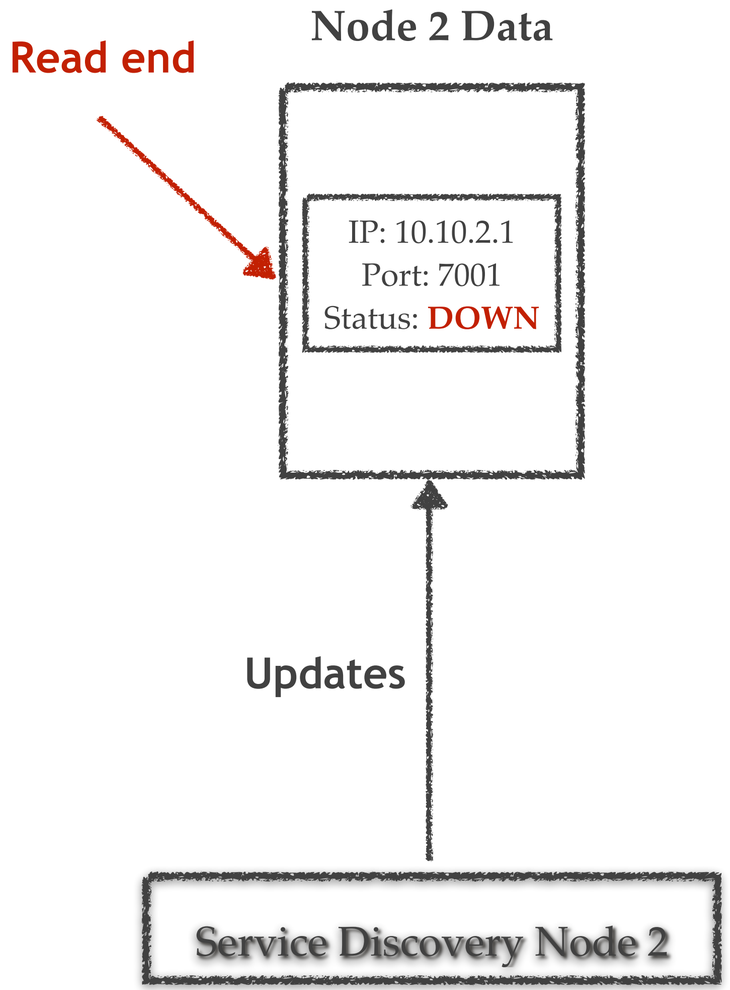
Read from a version (till it is gone)



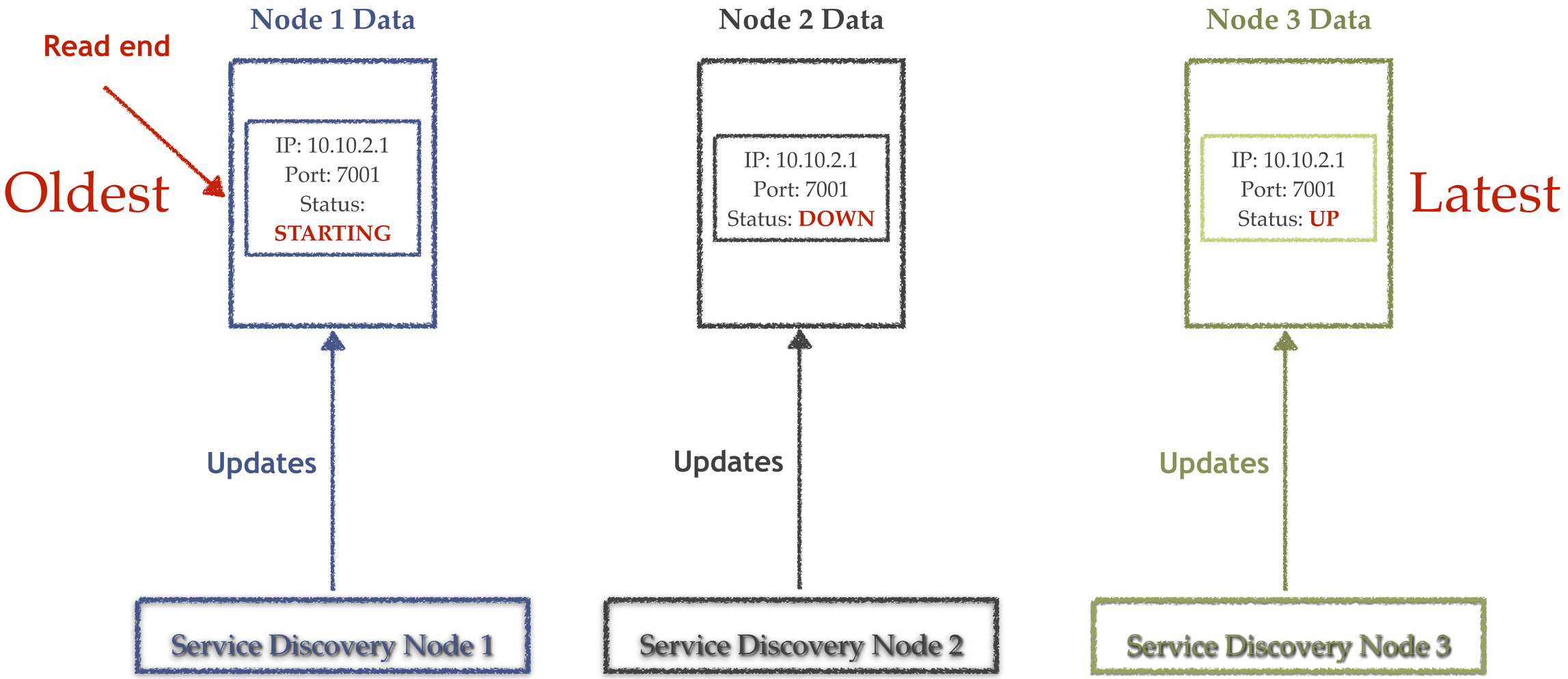
Read from a version (till it is gone)



Steady State



Time to converge (worst case)



Time to converge (worst case)

Time to evict stale copies. *(Constant)*

+

Time to replicate from the owner node.

Time to evict stale copy

Heartbeat interval

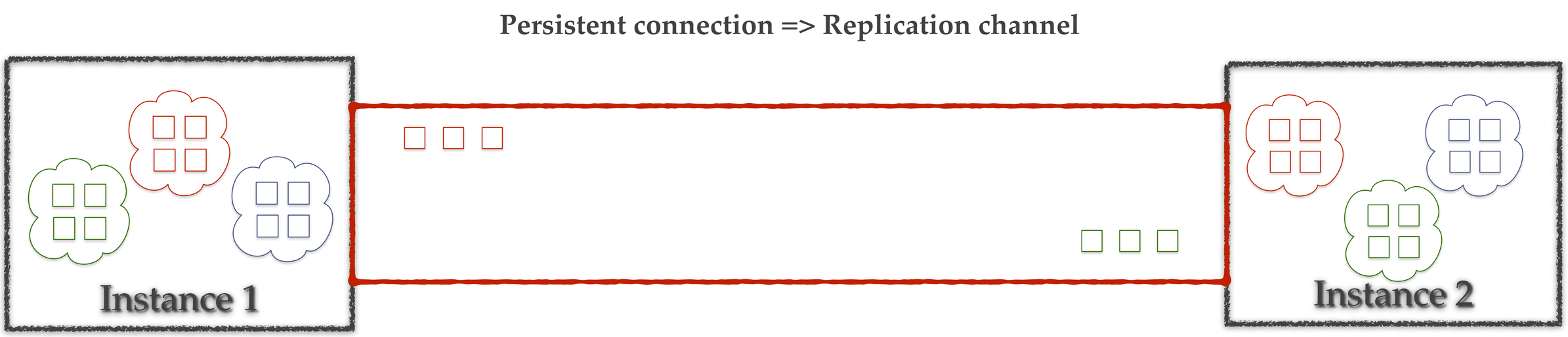
*****

Number of tolerated missing heartbeats

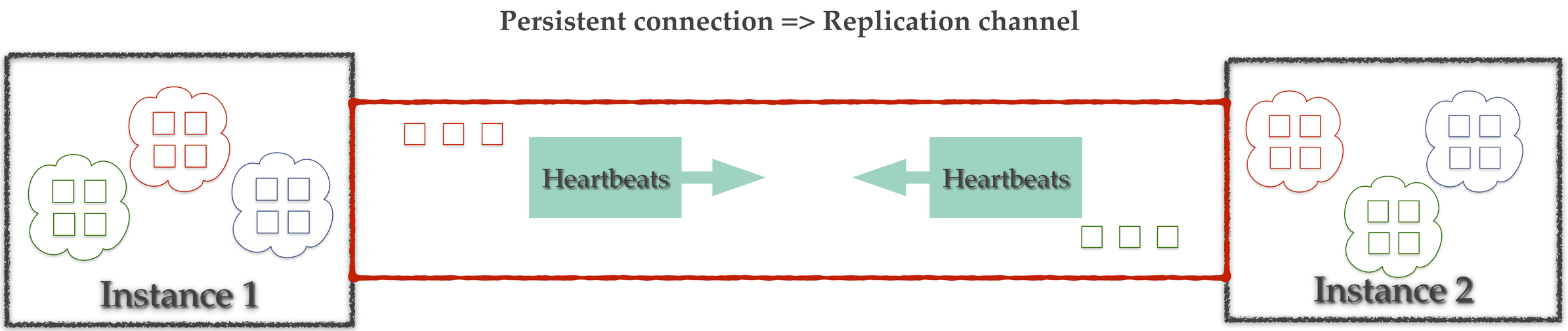
Time to replicate from the owner node

Theoretically unbounded

Adding bounds



Adding bounds



Time to replicate from the owner node

Somewhat bounded

Heartbeat interval

*****

Number of tolerated missing heartbeats

Time to converge (worst case)

Cost of divergence?

Cost of divergence?

Instance data is hardly changing!

In most cases,
there isn't a divergence
and when it happens,
the **impact** is **low**.

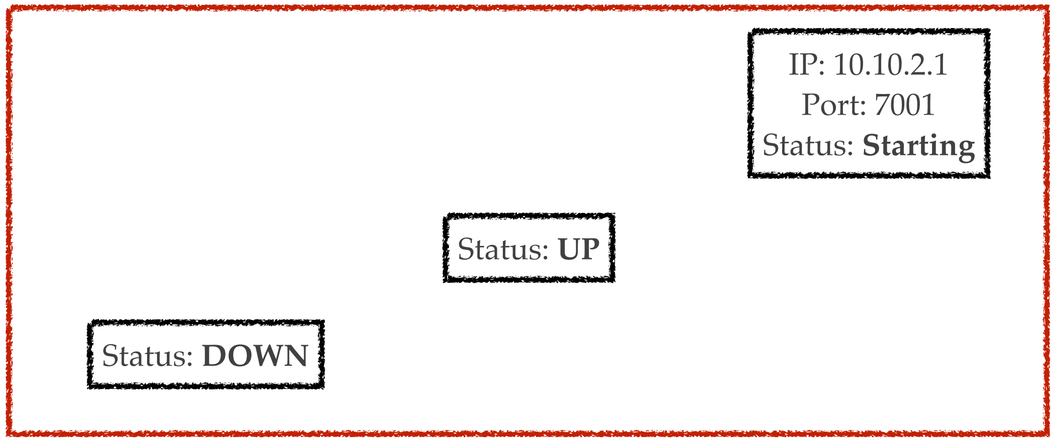
Cost of divergence?

Service discovery controls visibility of nodes but does **not** guarantee availability.

Reads

How to implement reads on service discovery data?

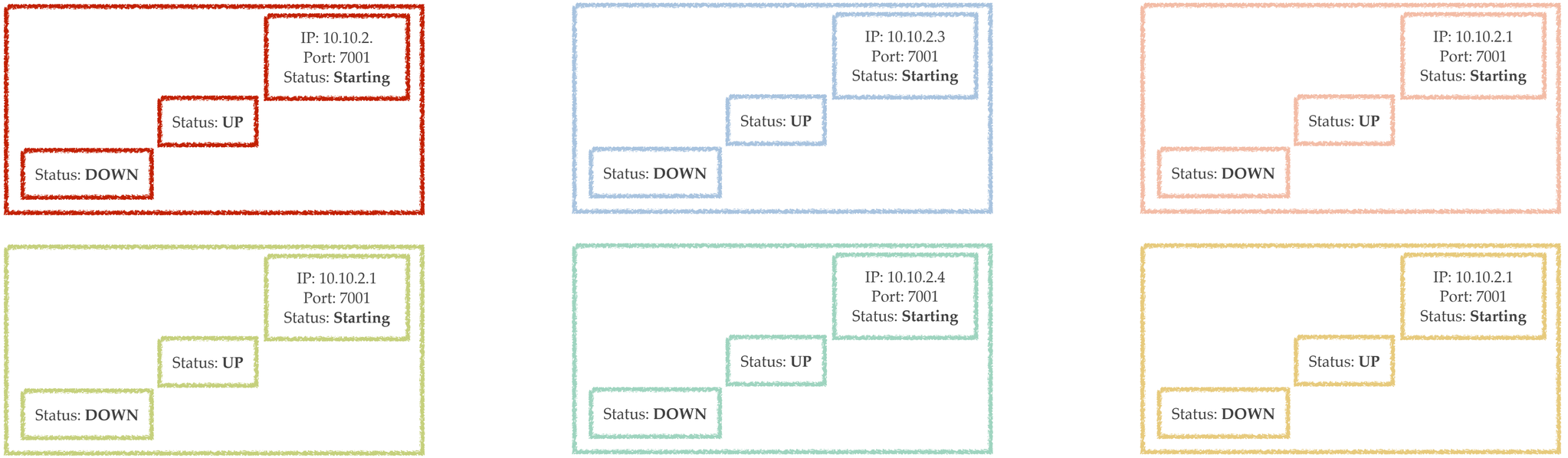
Service discovery data is an **ordered stream**.



Service discovery data is a

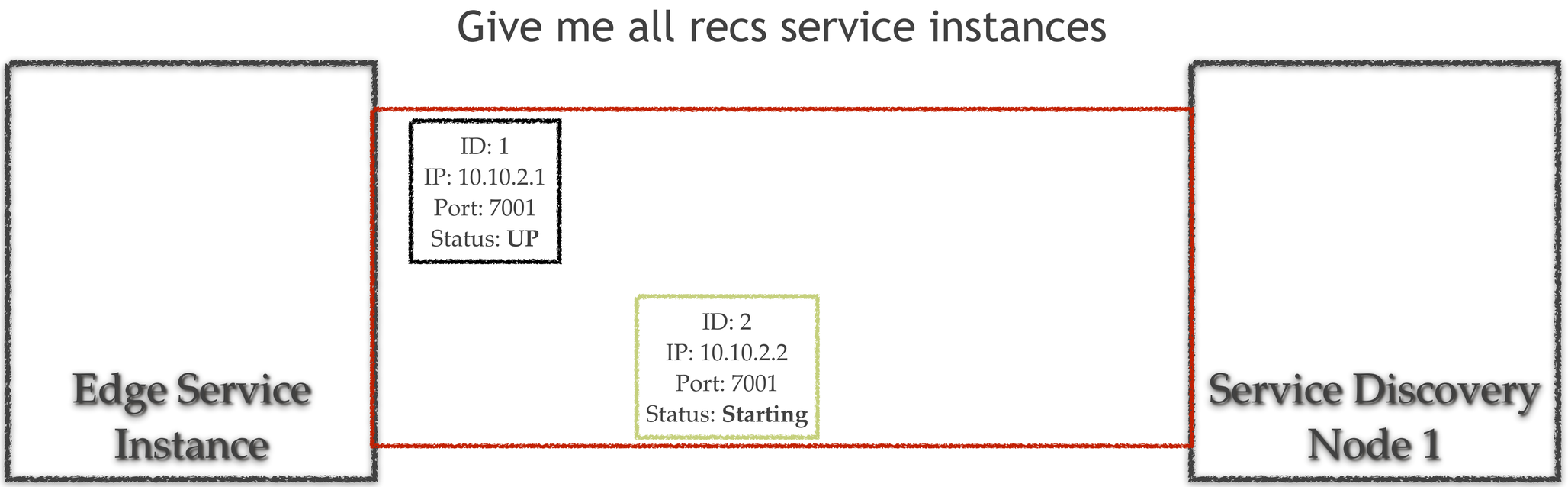
~~ordered stream~~

merged ordered stream

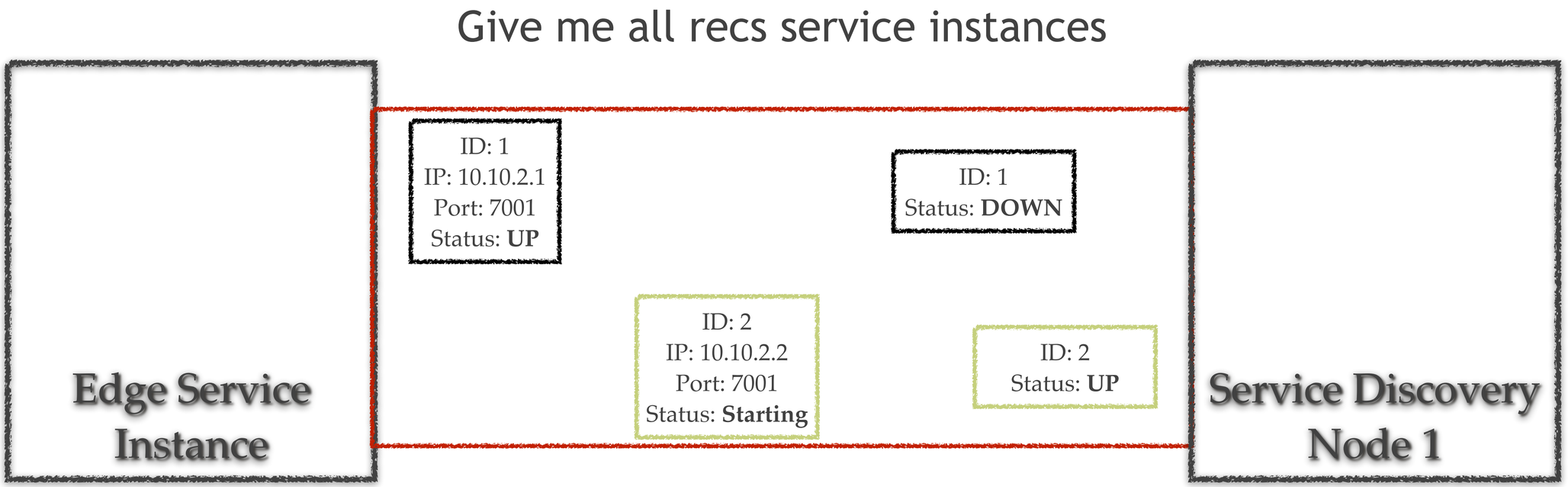


Data as a stream (Lookup)

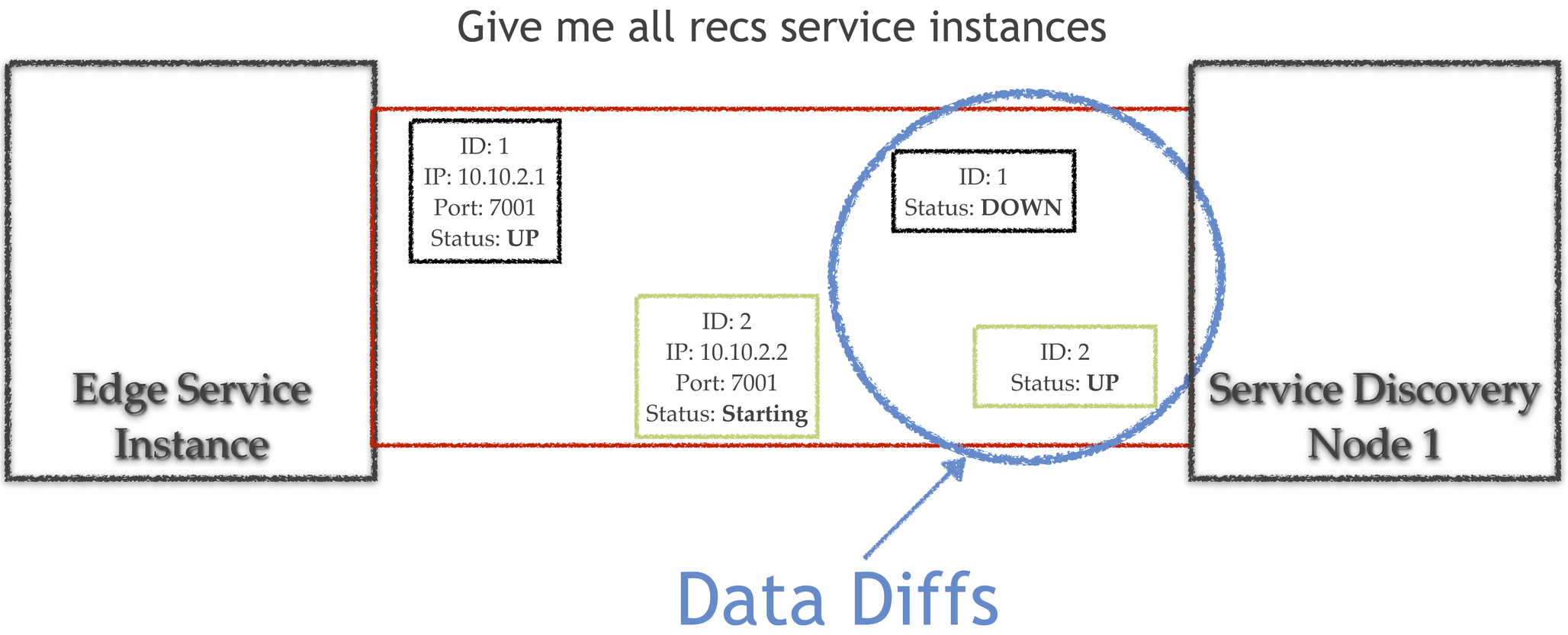
Data as a stream (Lookup)



Data as a stream (Lookup)

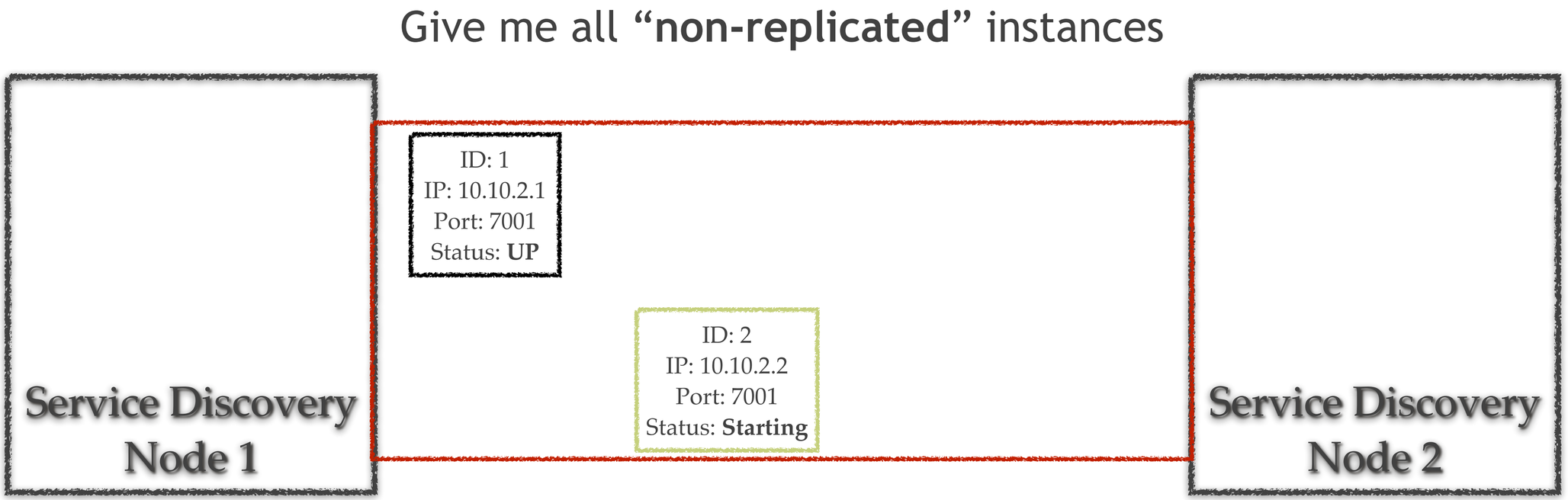


Data as a stream (Lookup)



Data as a stream (Replication)

Data as a stream (Replication)

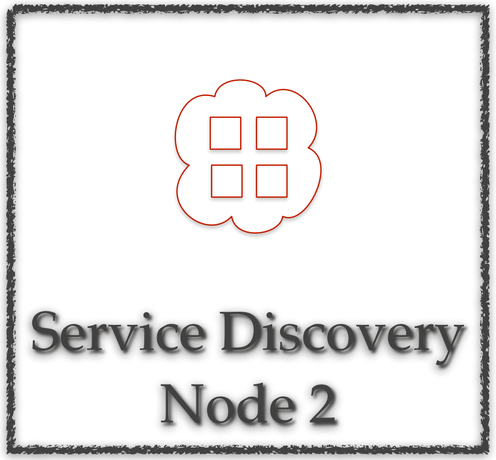


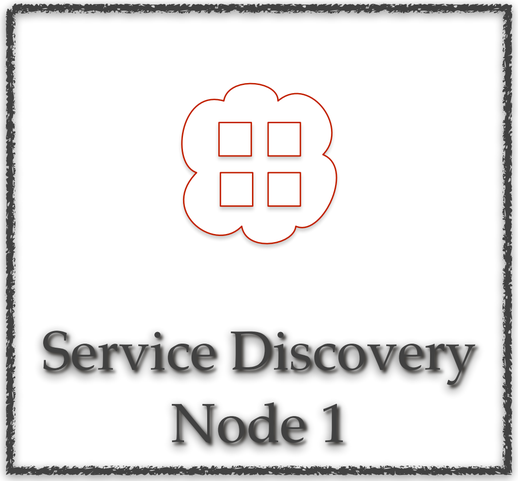
Data as a stream (Replication)

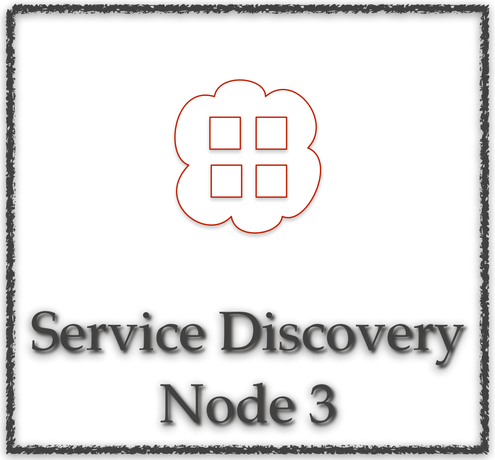

KEEP CALM
AND EMBRACE STREAMS

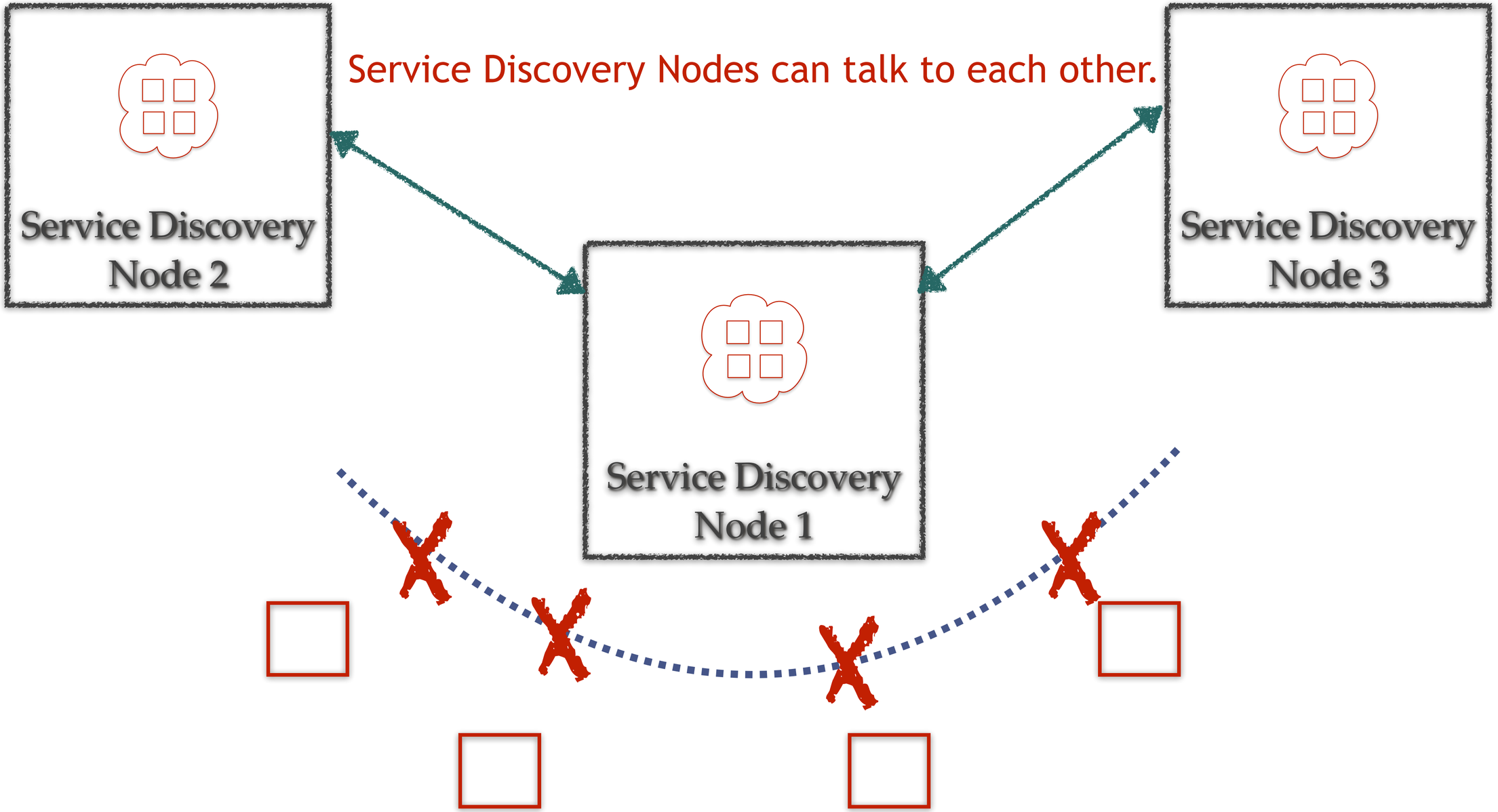
Learnt the hard-way

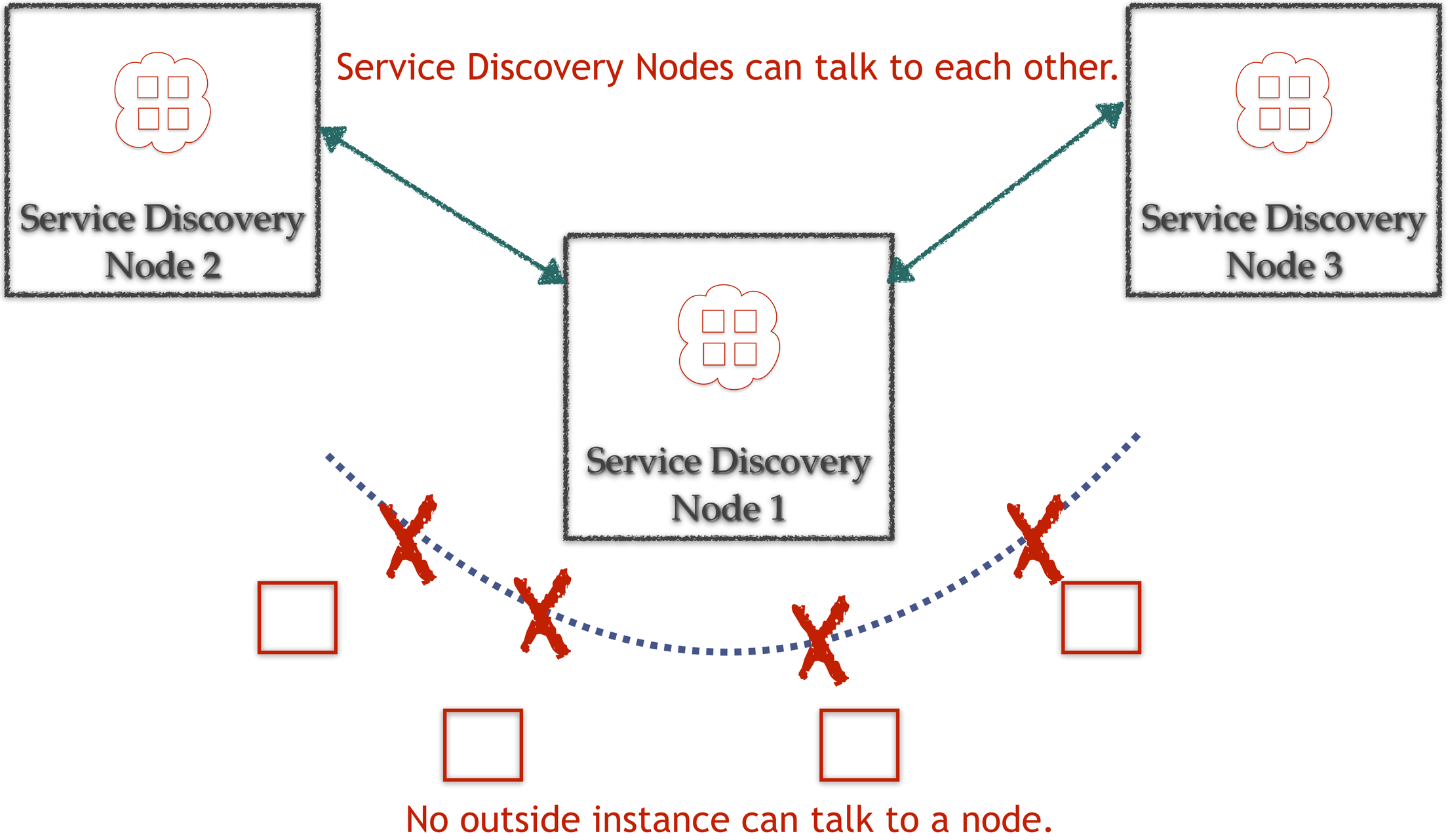
Network partitions gone wild!

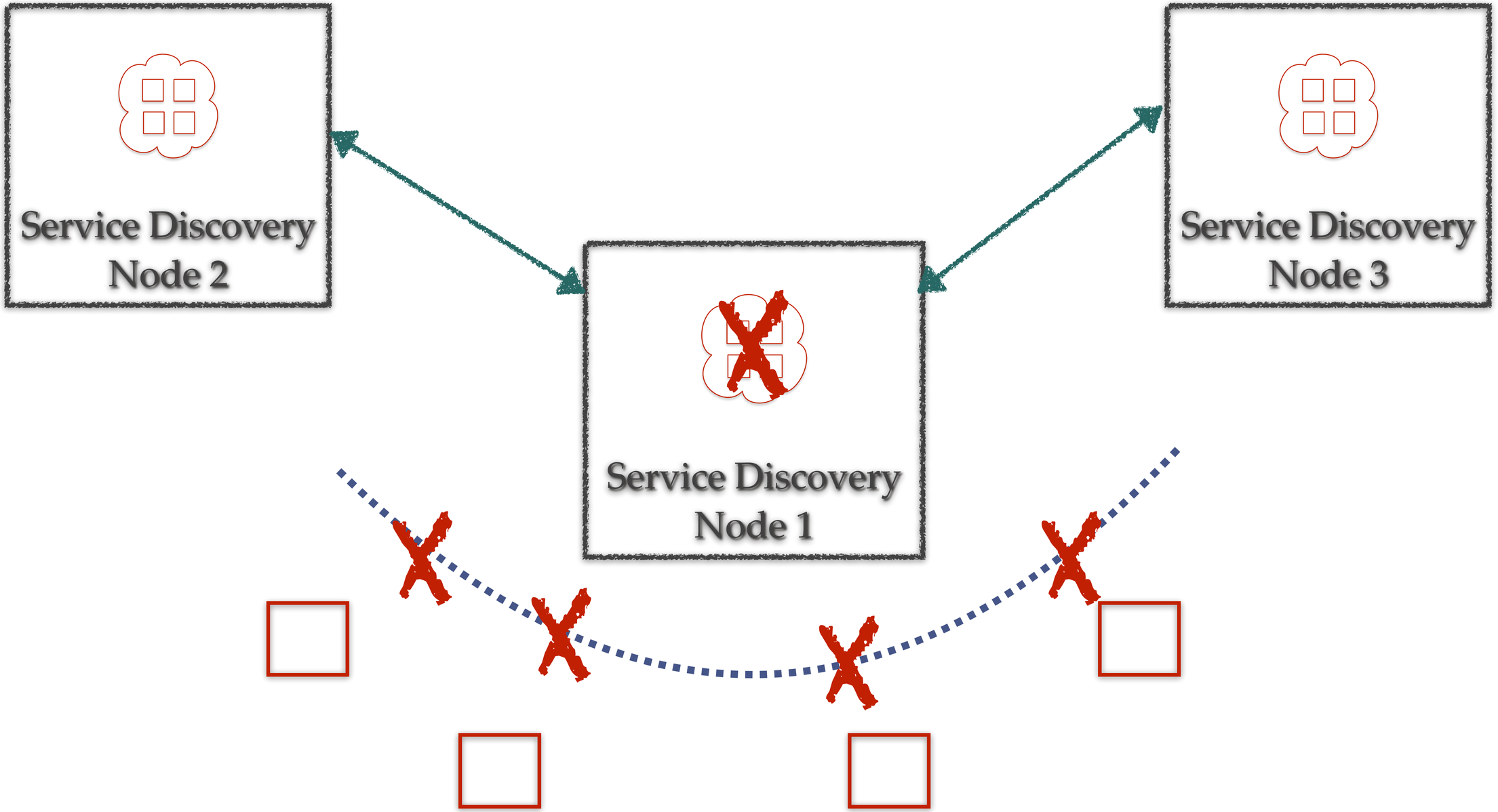

Service Discovery
Node 2

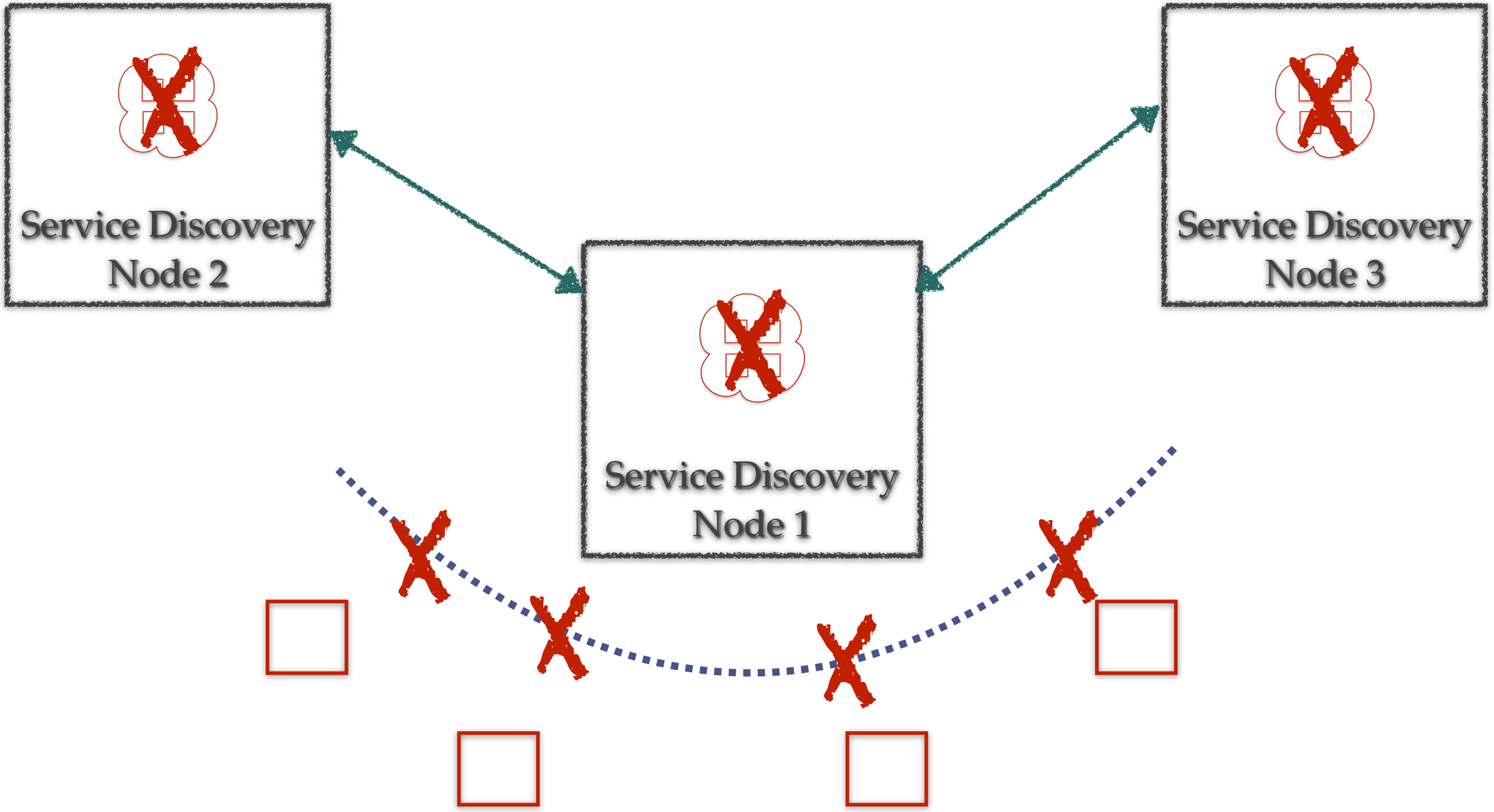

Service Discovery
Node 1


Service Discovery
Node 3


Service Discovery Nodes can talk to each other.
Service Discovery Node 2
Service Discovery Node 1
Service Discovery Node 3


Service Discovery Nodes can talk to each other.
Service Discovery Node 2
Service Discovery Node 3
Service Discovery Node 1
No outside instance can talk to a node.


Service Discovery
Node 2
Service Discovery
Node 1
Service Discovery
Node 3


Service Discovery
Node 2
Service Discovery
Node 3
Service Discovery
Node 1

One fine day ....

... you lost most of your service instances ...

One fine day ....

... you lost most of your service instances ...

because ...

one node of service discovery was partitioned

Taming evictions
a.k.a
Self preservation

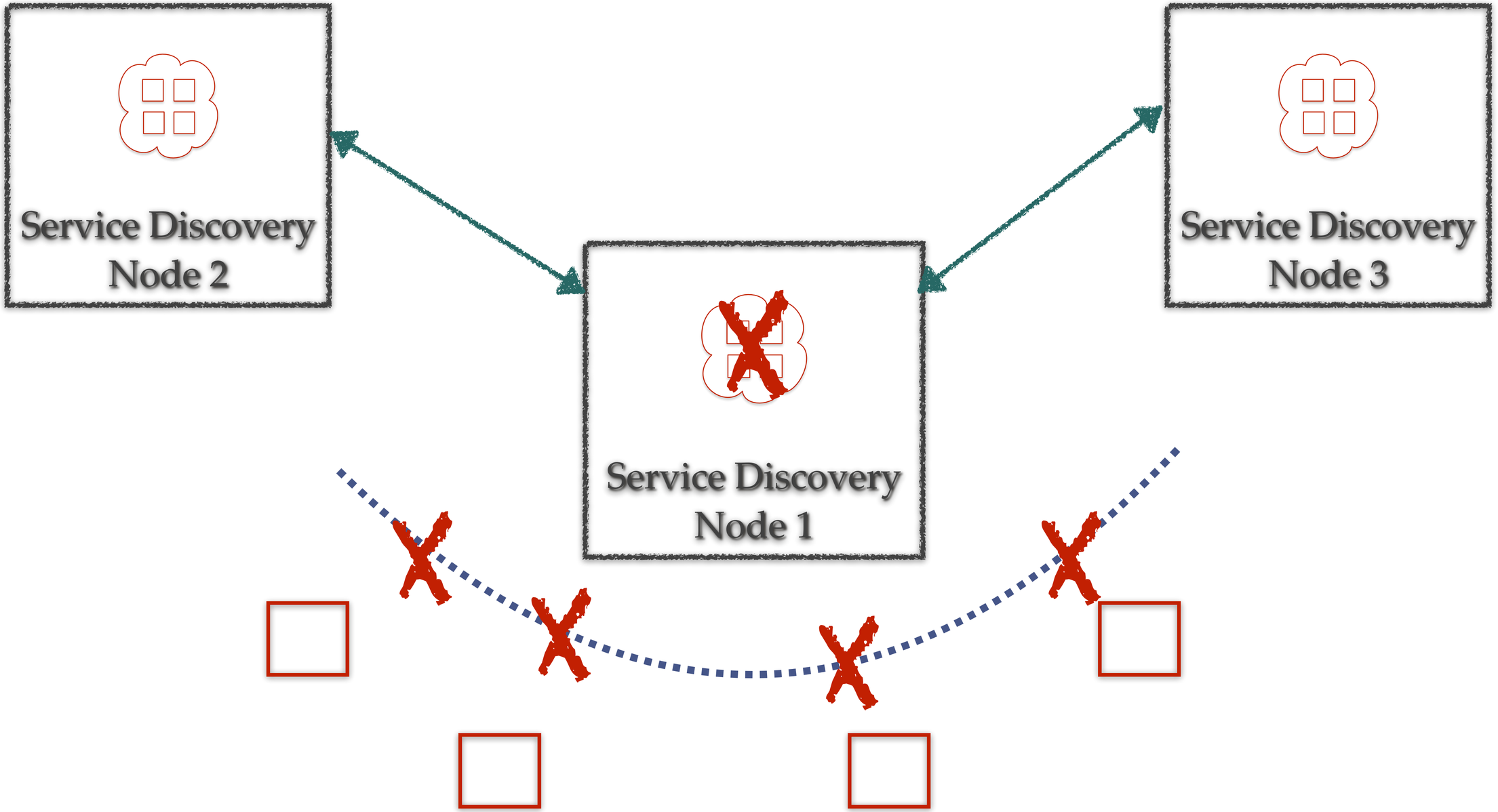
All instances “generally” do not vanish

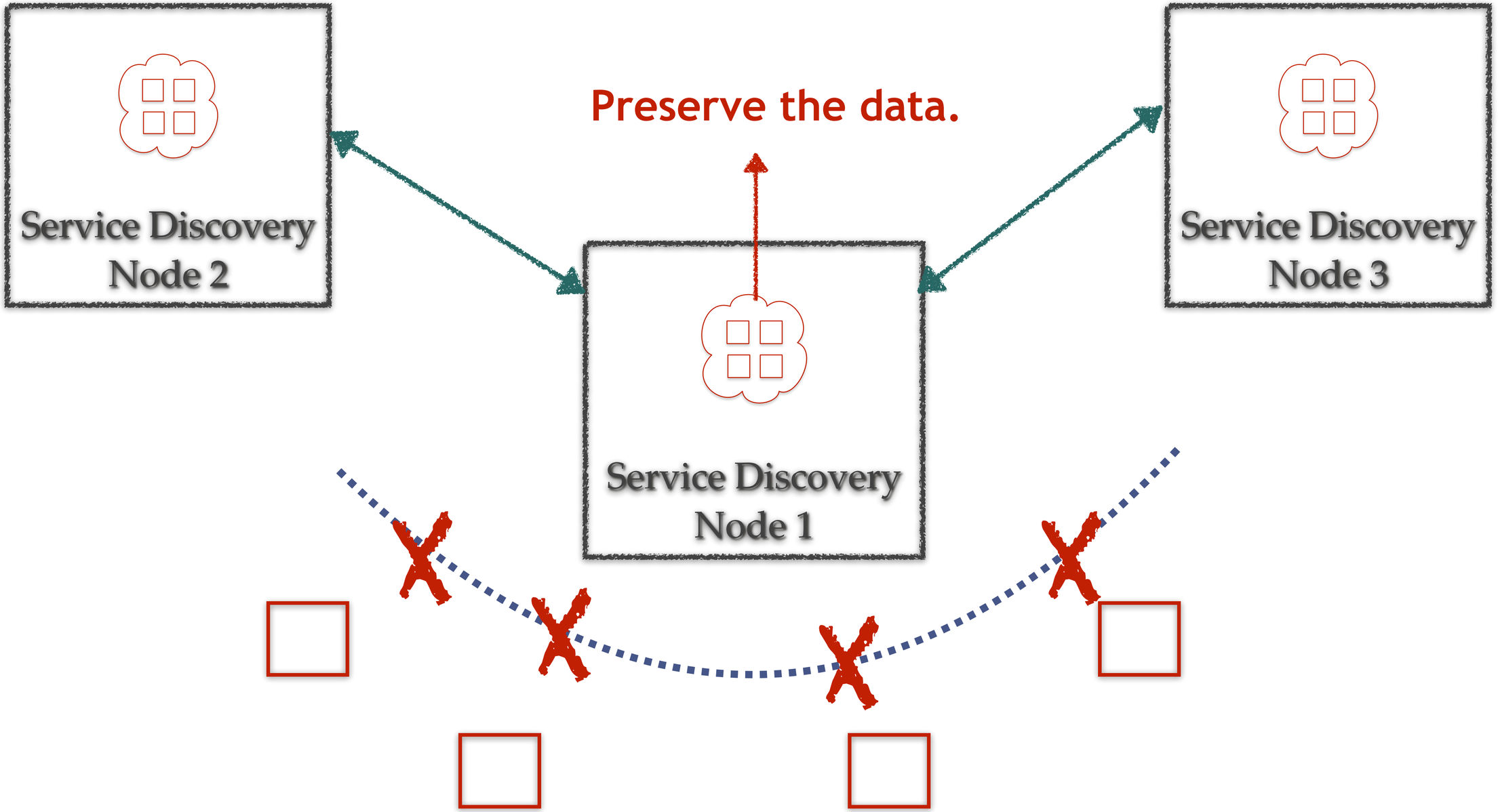
All instances “generally” do not vanish

If they do, let local decisions prevail

Do NOT evict

If > X % of instances vanish


Service Discovery
Node 2
Service Discovery
Node 1
Service Discovery
Node 3


Service Discovery
Node 2
Preserve the data.
Service Discovery
Node 3
Service Discovery
Node 1

Clients talking to the “in-doubt” instance will detect failure, if any.

Total unavailability
a.ka.
Apocalypse

Impact

Fatal

New nodes can not be started.

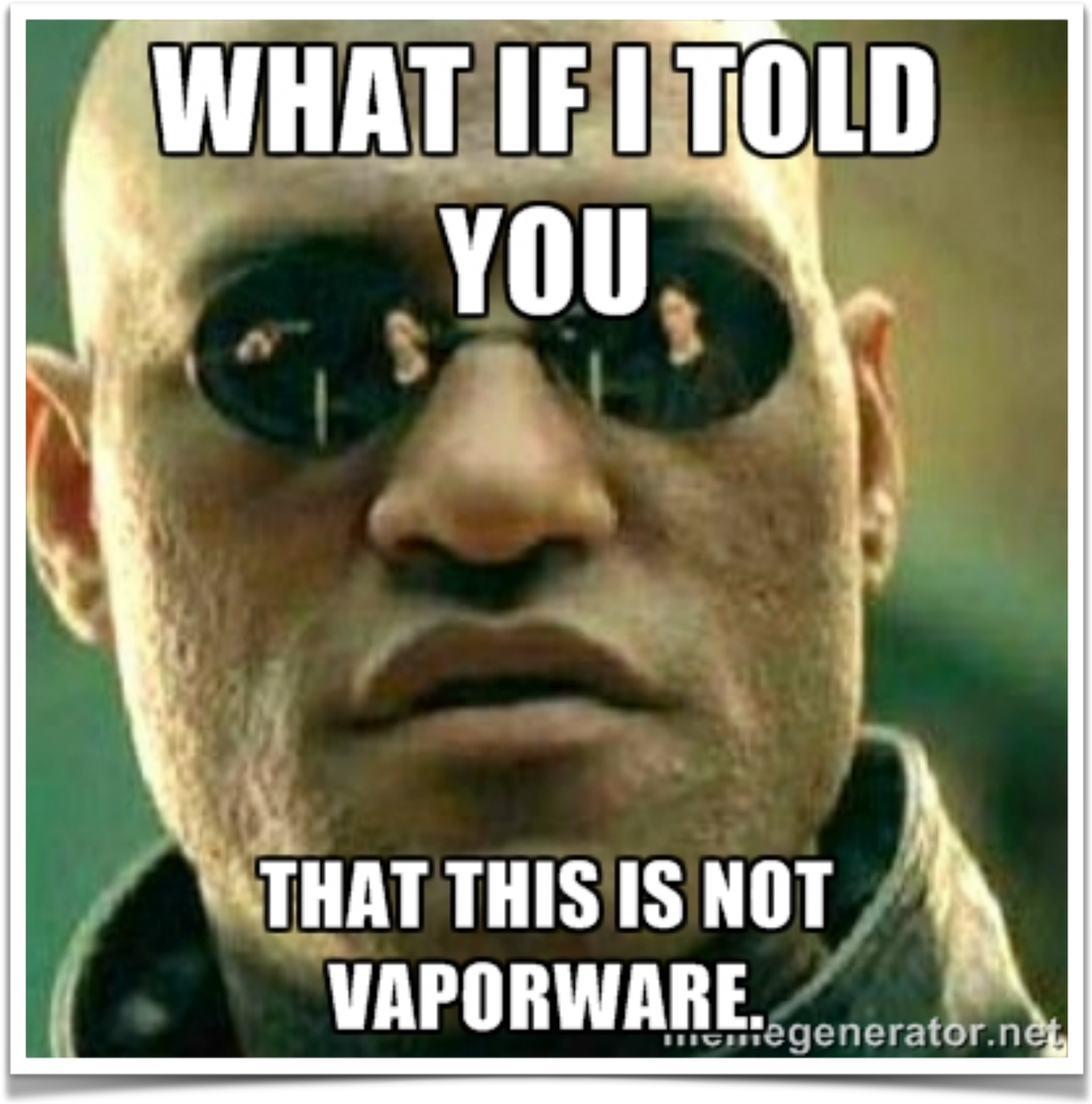
Impact

Fatal	Degraded
New nodes can not be started.	Existing nodes uses cached data.

NETFLIX | OSS

Netflix Eureka V2

https://github.com/Netflix/eureka



Don’t make Service Discovery your Achilles heel.

Service discovery controls visibility of nodes but does **not** guarantee availability.


thank you
danke
謝謝
ngiyabonga
teşekkür ederim
рахмат
Баярлалаа
спасибо
gracias
merci
obrigado
dziękuję
bedankt
dank je
sukriya
kop khun krap
terima kasih
go raibh maith agat
mochchakkeram
arigatô
dakujem
grazie
takk
tapadh leat
мерси
ありがとう
감사합니다
ευχαριστώ
sagolun
mauruuru
hvala
köszönöm
enkosi
spas
tack
mersi
barka
welalin
vinaka
blagodaram
misaotra
matondo
paldies
grazzi
mahalo
хвала
asante
manana
obrigada
murakoze
tenki
djiere dieuf
дякую
mamnun
chokrane
trugarez
shukriya
mercé
dhanyavadagalu
diolch
xièxie
tanemirt
rahmet
najis tuke
kam sah hamnida
didi madloba
mèsi
sobodi
dèkuji
chnorakaloutioun
gratias ago
gràcies
sulpáy
taiku
akun
dankon
kiitos
dankie
dhanyavad
nandri
nanni
bayarlalaa
gràcie
faafetai lava
спасибі
kia ora