Introduction à

Cassandra

CampusPlex - Ajaccio

6 janvier 2012

Nicolas Romanetti

jaxio

A propos

- Nicolas Romanetti
- Co-fondateur JAXIO / Architecte / Développeur
- Utilisation de Cassandra lors du Challenge USI 2011

jaxio

Mon objectif

- Vous faire passer de «c'est quoi ce truc!» à «je pige!»
- Vous donner envie d'approfondir le sujet

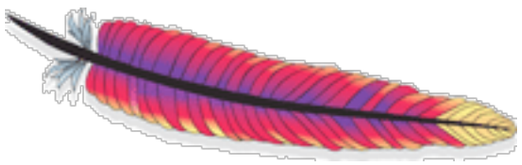
jaxio

Cassandra

- Base de données distribuée conçue pour
  - gérer une grande quantité de données
  - résister aux pannes
  - être performante

Historique

- Créée par Facebook (Inbox search)
- Open Source en 2008
- Top Level Apache Project en 2010
- Abandonné par Facebook en 2010
- Utilisée par Netflix, Twitter, etc...



Data Model

☞ vocabulaire déroutant...

Column

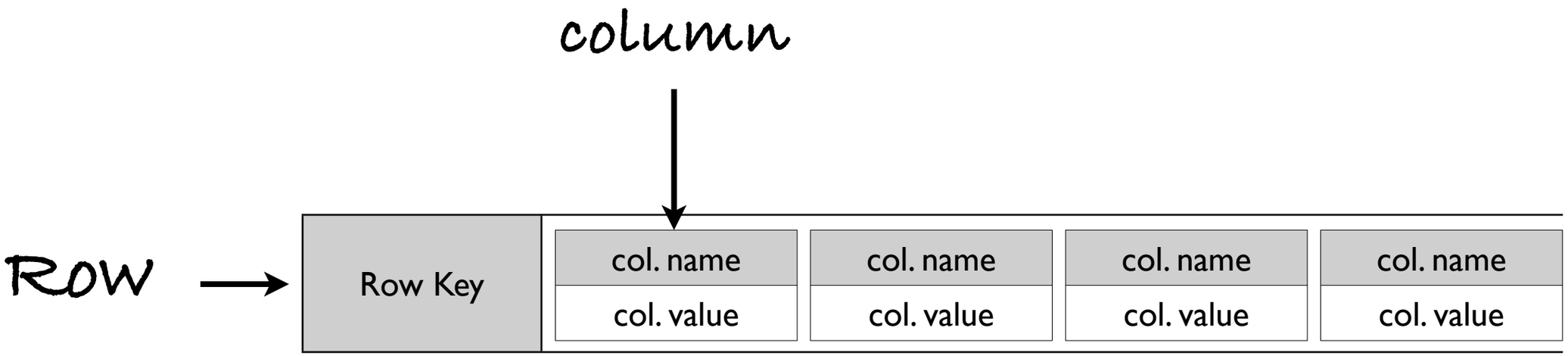
name
value
timestamp

La colonne est la plus petite unité de donnée

name: 64Ko max
value: 2Go max

jaxio

Row

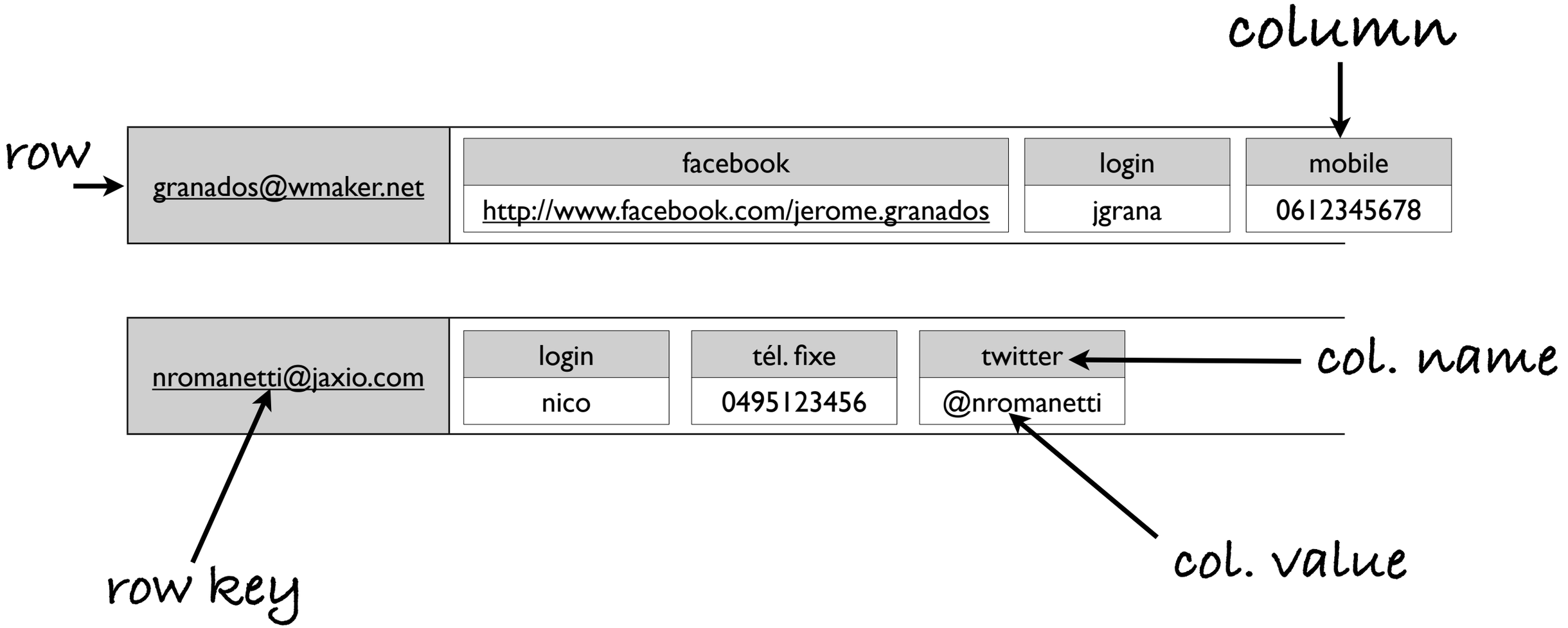


Une row: un ensemble de «columns»

row key: 64ko max
nb max de colonnes: 2 milliards

jaxio

Exemple



Pas de contraintes sur les colonnes...

Column Family

≃Table en relationnel

User

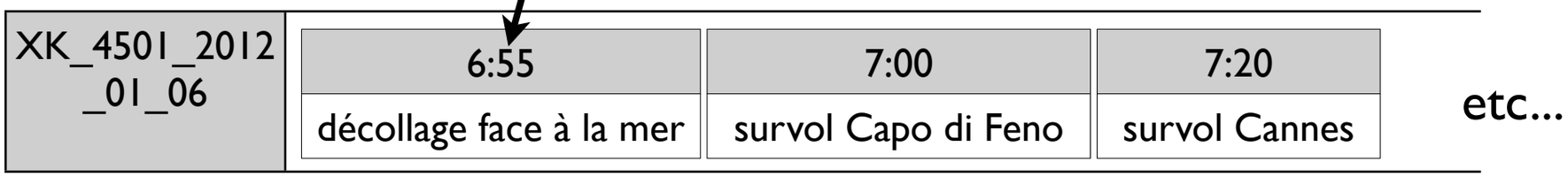
granados@wmaker.net	login	facebook	mobile
	jgrana	http://www.facebook.com/jerome.granados	0612345678

nromanetti@jaxio.com	twitter	login	tél. fixe
	@nromanetti	nico	0495123456

```
User[granados@wmaker.net][mobile] = 0612345678
```

col. Family — row key — col. name — col. value

Le nom est une valeur



- Le nom de la colonne est aussi une valeur!
- Les colonnes sont triées par leur nom
- Choix de la stratégie de tri (comparator)
  Long, UTF8, Byte, etc...

***XK_4501_2012_01_06**: vol Air Corsica AJA -> ORY du 6 janvier 2012 à 06:55*

jaxio

Liens

‘BlogEntries’

wmaker	2011/12/31 10:52	2012/01/01 23:12	2012/01/06 11:11	etc...
	rowkeyA	rowkeyB	rowkeyC	

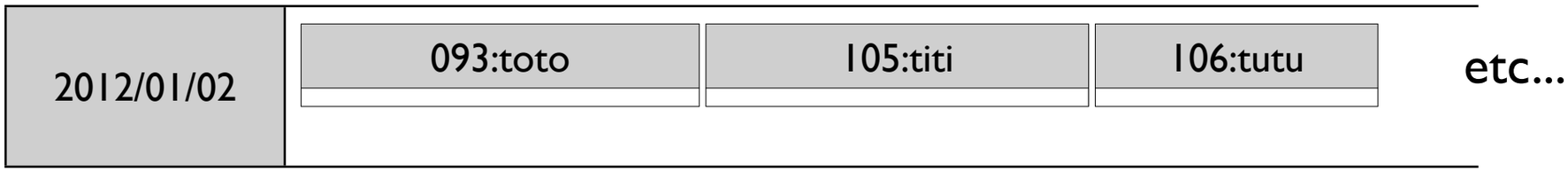
‘BlogEntry’

rowkeyA	Content	date	Title	etc...
	En cette fin d’année....	2011/12/31 10:52	Bye bye 2011	

jaxio

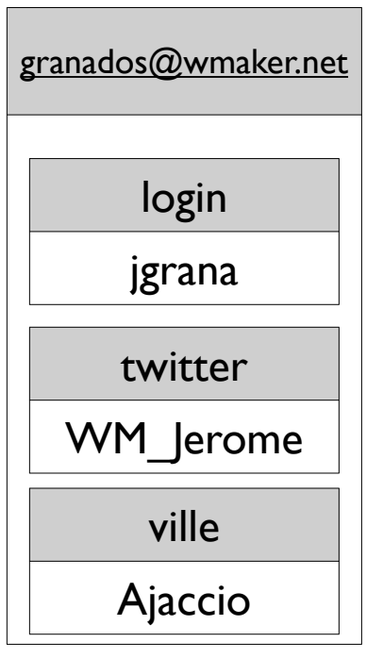
Valueless column

‘TopScores’



- Le nom de la colonne porte à lui seul l’information
  - ici: le score suivi du username
- Intérêt? performance...

Super Column



- Une column contenant des columns
  - grouper des informations dépendantes

```
User[allusers][granados@wmaker.net][ville] = Ajaccio
```

Super col. Family	row key	super col. name	sub col. name	sub col. value

Column avec plusieurs données

contactInfo
tel:04..., email: nr@..., city: Ajaccio, etc...

Expiring Column

- Une colonne avec un TTL (Time To Live)

Counter Column

- Fonctionnalité récente
- Permet d'incrémenter un compteur distribué

Tombstone

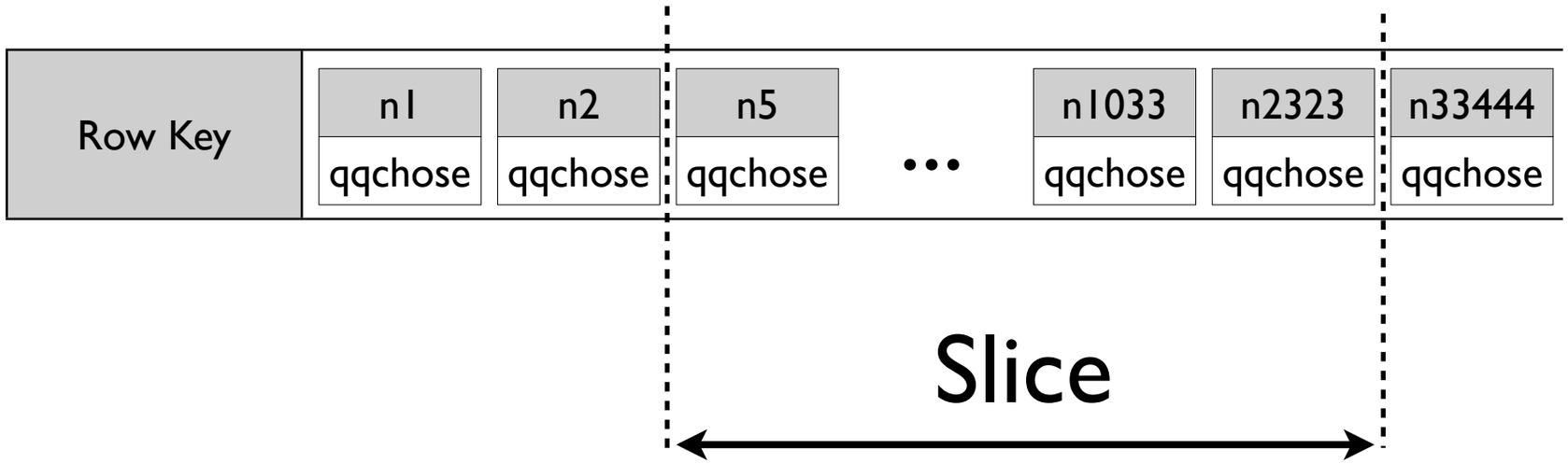
- Delete se fait en écrivant une colonne spéciale!
  - Timestamp fait foi

2 usages pour les rows

- **Wide Row**: Contient des milliers, millions de colonnes...
  - données temporelles (timeline twitter, logs, etc...)

- **Skinny Row**: En contient moins

Requêtes sur les colonnes



- Pas de join!
- On récupère une ‘Slice’ de column
  - ex: toutes les colonnes entre ‘n5’ et ‘n2323’

Concevoir son modèle de données en fonction des requêtes que l’on veut faire

jaxio

Secondary Index

nromanetti@jaxio.com	twitter	country	tél. fixe
	@nromanetti	FR	0495123456

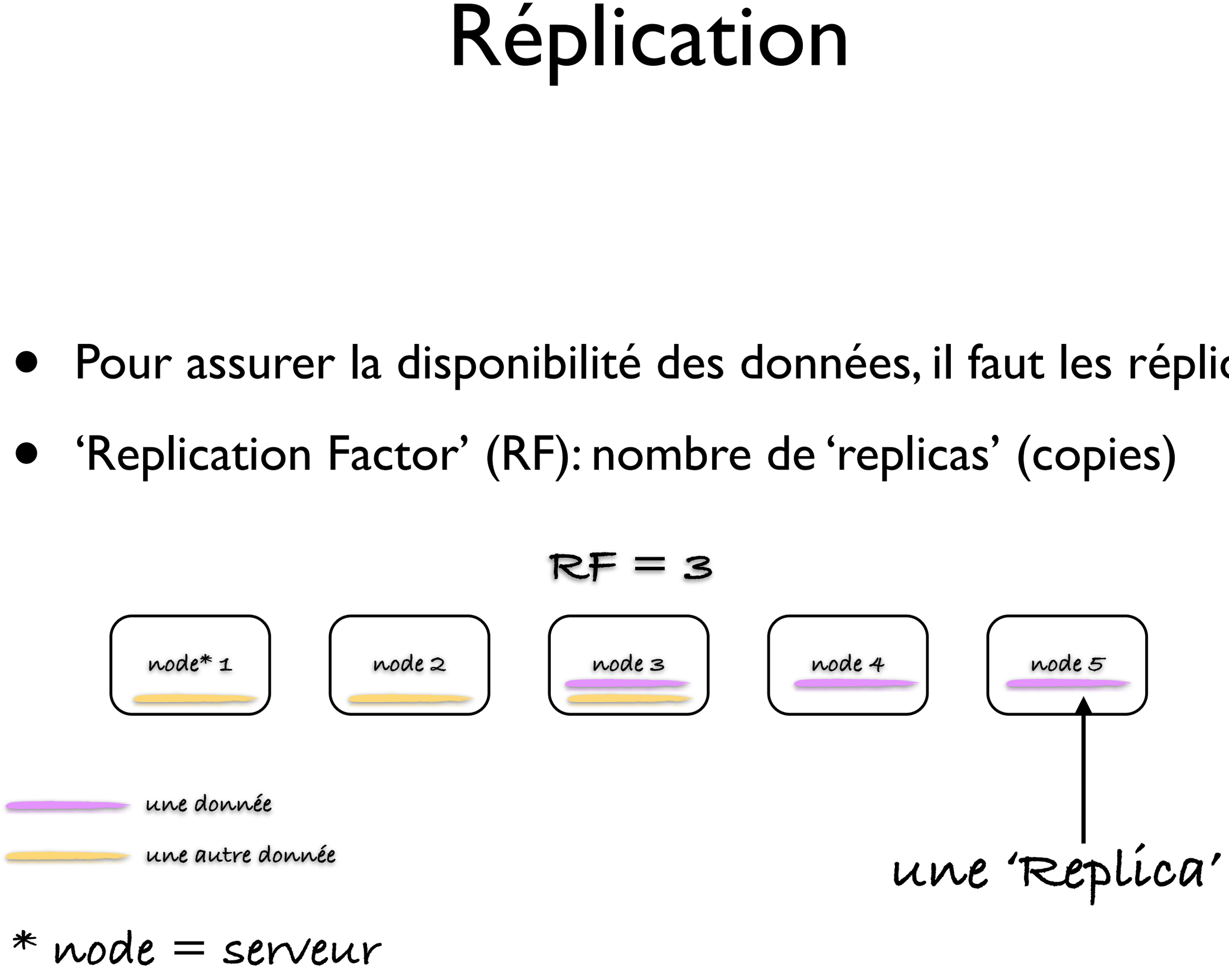
- Index sur la valeur d'une colonne
  - Exemple: index sur la valeur de la column 'country'
  - `get users where country = 'FR';`

Révisions :)

- column (normale ou super)
- wide row, skinny row
- column family
- requêtes (column slice) / secondary index
- valueless column
- liens
- tombstone

Réplication

- Pour assurer la disponibilité des données, il faut les répliquer
- 'Replication Factor' (RF): nombre de 'replicas' (copies)



jaxio

Répliquer, mais où?

- Tous les serveurs sont identiques
- Postulats:
  - 1 row doit tenir sur 1 serveur
  - On veut répartir les rows uniformément

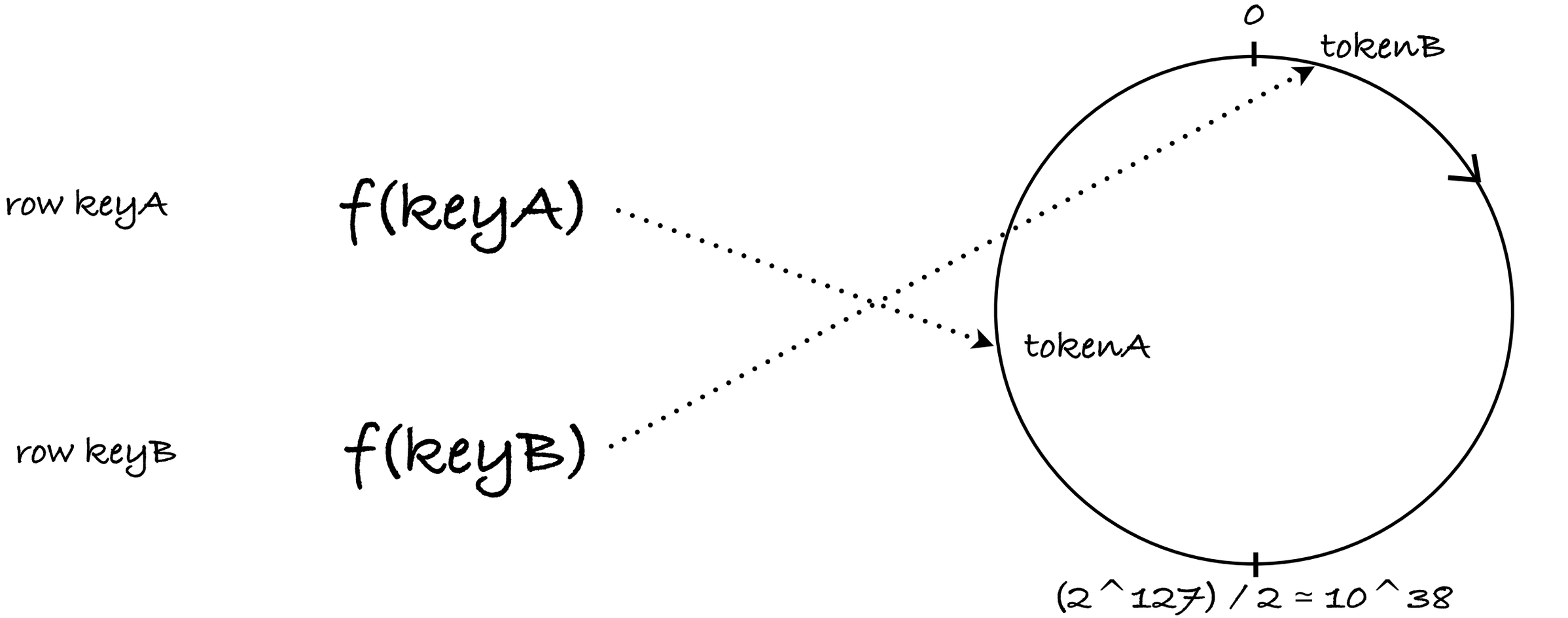
jaxio

Consistent Hashing

bytes
e.g. String

Consistent
Hash

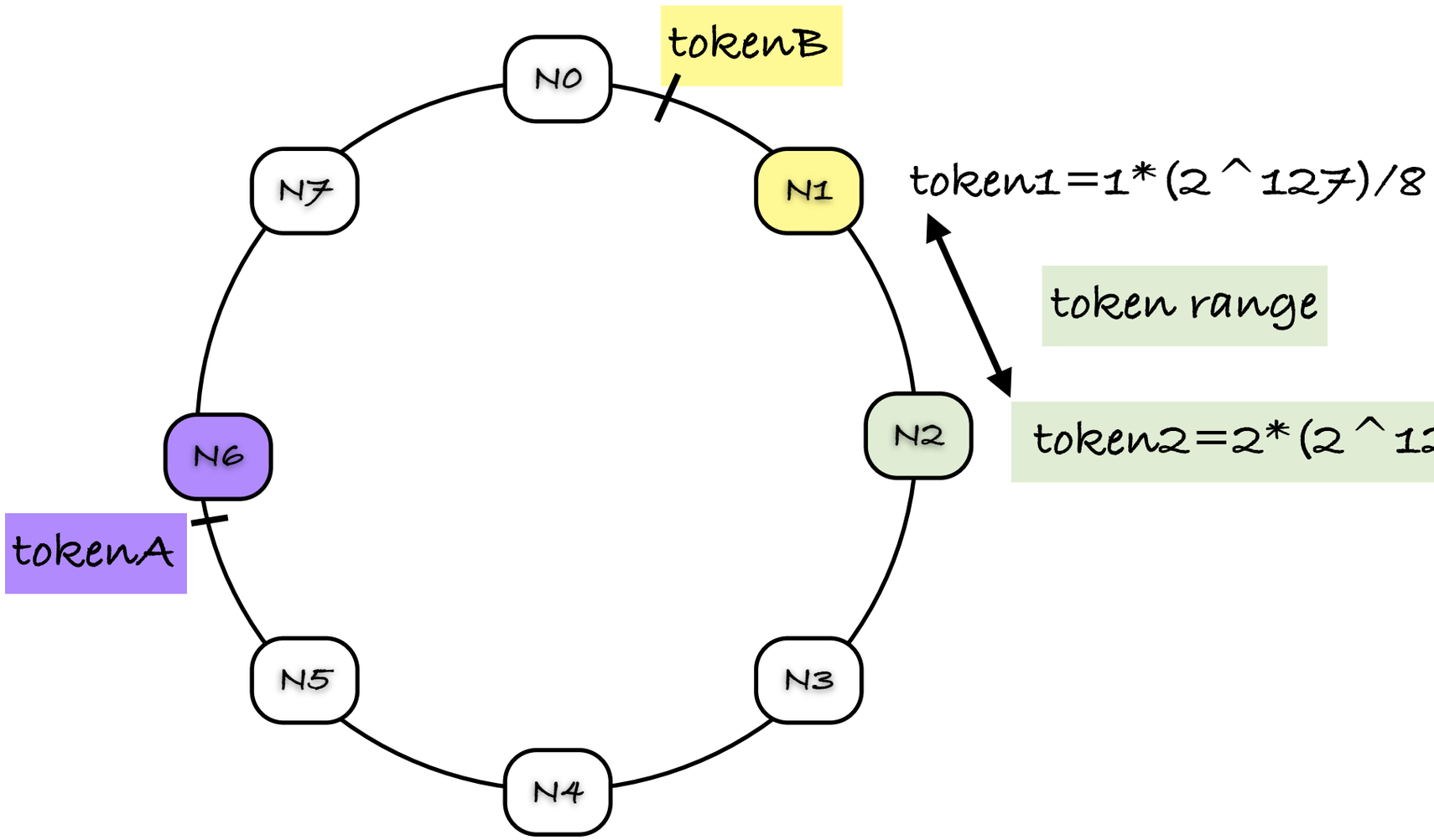
Un ‘token’ entre
0 et 2^127



jaxio

Cassandra Ring

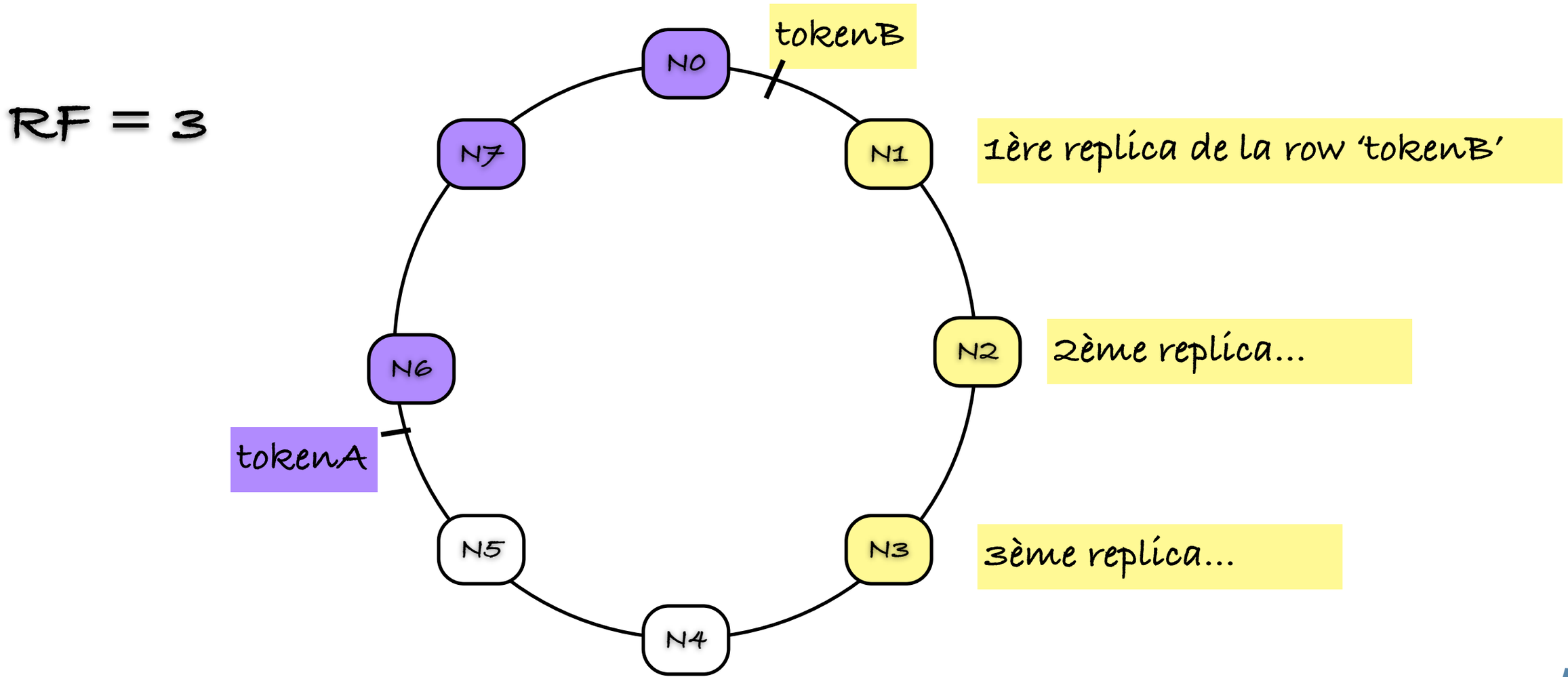
- A chaque noeud on attribut un numéro de token
  - #token = node number * (2^127) / nb of nodes



node 2 = contient 1ère replica des rows ayant leur token compris entre token1 et token2

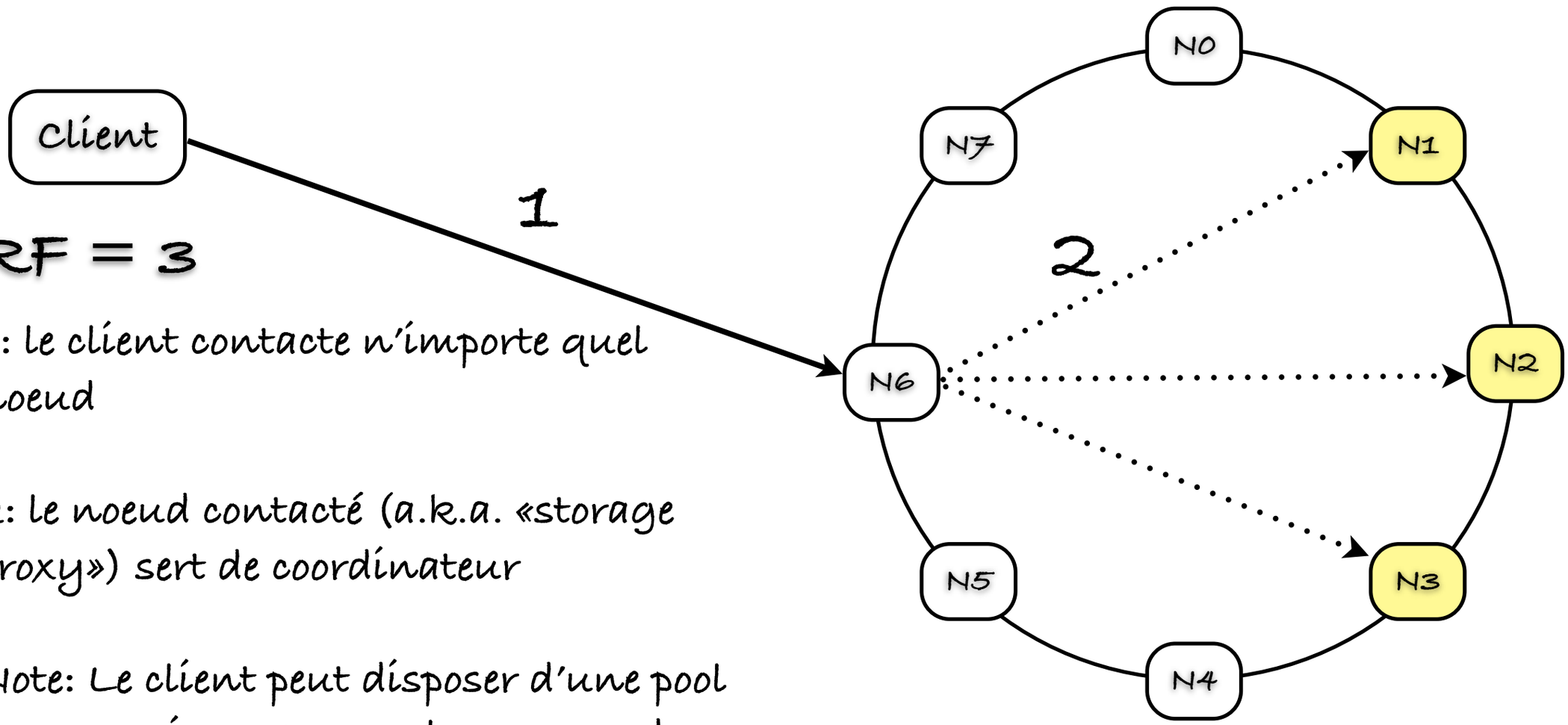
Stratégie de placement

- RandomPartioner: Répartir uniformément les différentes ‘rows’ entre les serveurs
- Placement Strategy: par défaut, sur les noeuds suivants



jaxio

Ecriture



RF = 3

1: le client contacte n'importe quel noeud

2: le noeud contacté (a.k.a. «storage proxy») sert de coordinateur

Note: Le client peut disposer d'une pool de connexions vers quelques noeuds.

Le client ne sait pas où la donnée sera écrite

Ecriture

- Update
  - On crée!
- Delete
  - Crée la column sous forme de Tombstone

create

password
xyz
10000

update

password
abc
20000

delete

password
30000

3 columns!

Le timestamp le plus récent fait foi

Ecriture

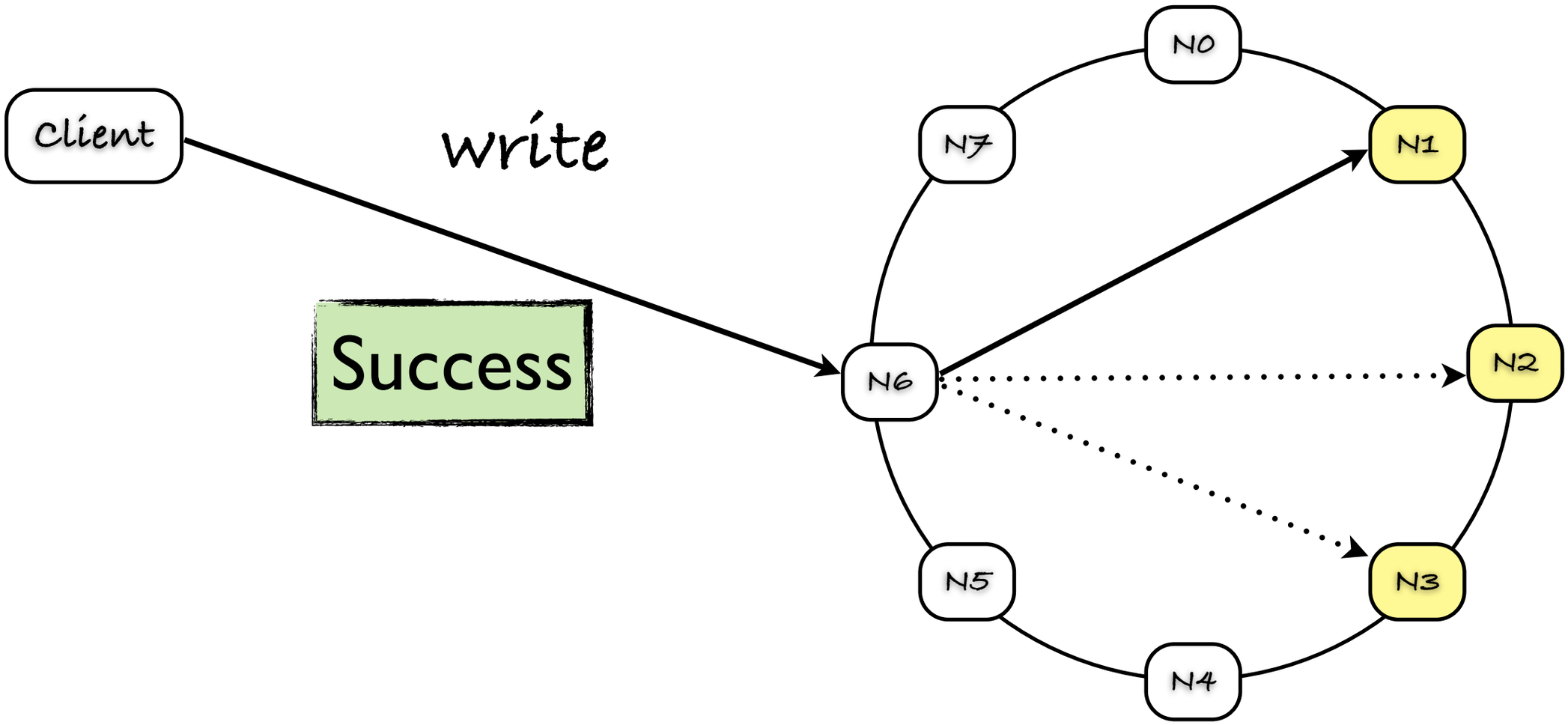
- Comment le client s'assure-t-il que la donnée est effectivement écrite et répliquée?
  - Attendre une réponse de tous les serveurs impliqués?
  - Ne pas attendre?
- En pratique, cela dépend du use-case
  - Solution: Tuneable Consistency

jaxio

Tuneable Consistency

- En écriture, le client indique le Consistency Level désiré
  - **ONE**: Je veux être certain qu'au moins une replica a été persistée durablement
  - **QUORUM**: Je veux être certain que la majorité des replicas ont été persistées durablement
  - **ALL**: Je veux être certain que toutes les replicas ont été persistées durablement

Consistency Level ONE

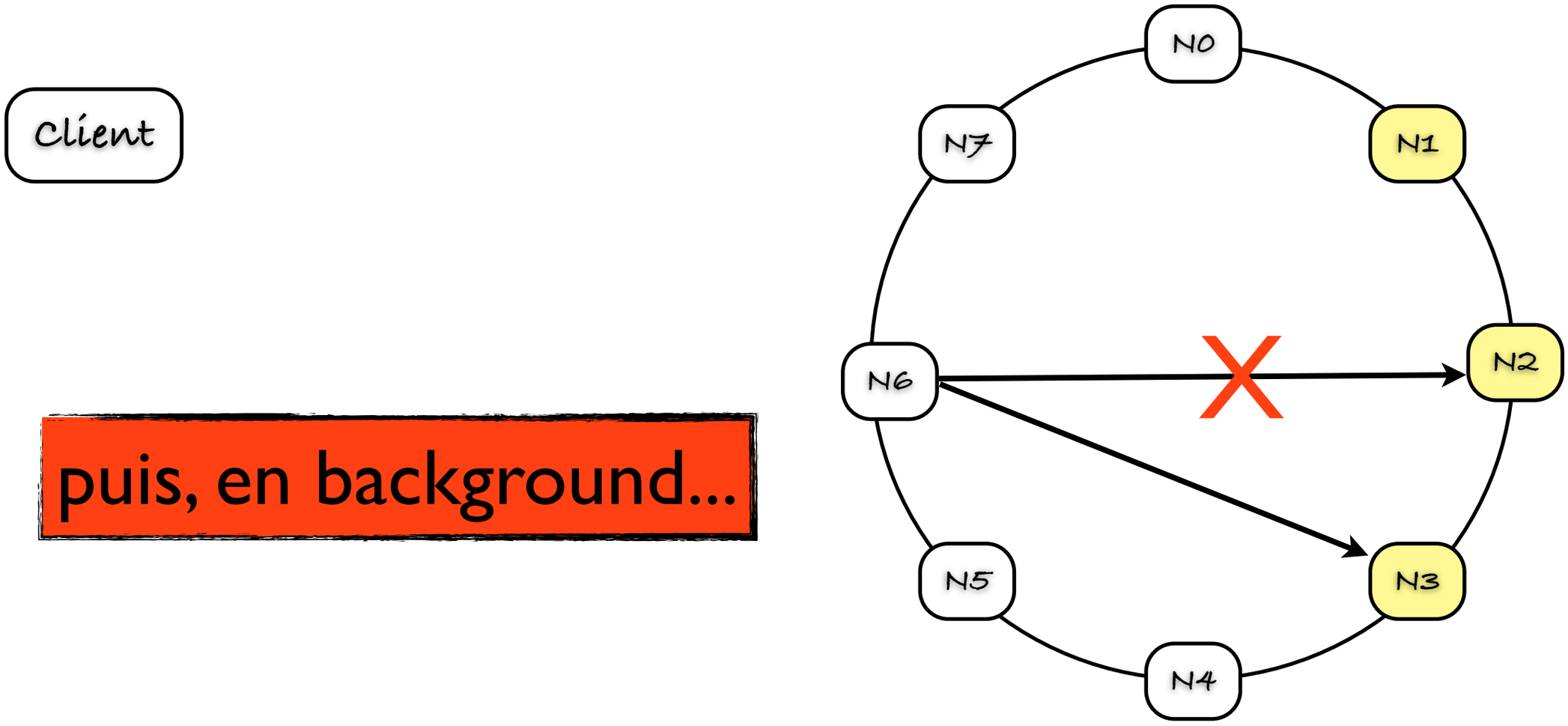


RF = 3

·····➤ write en asynchrone

jaxio

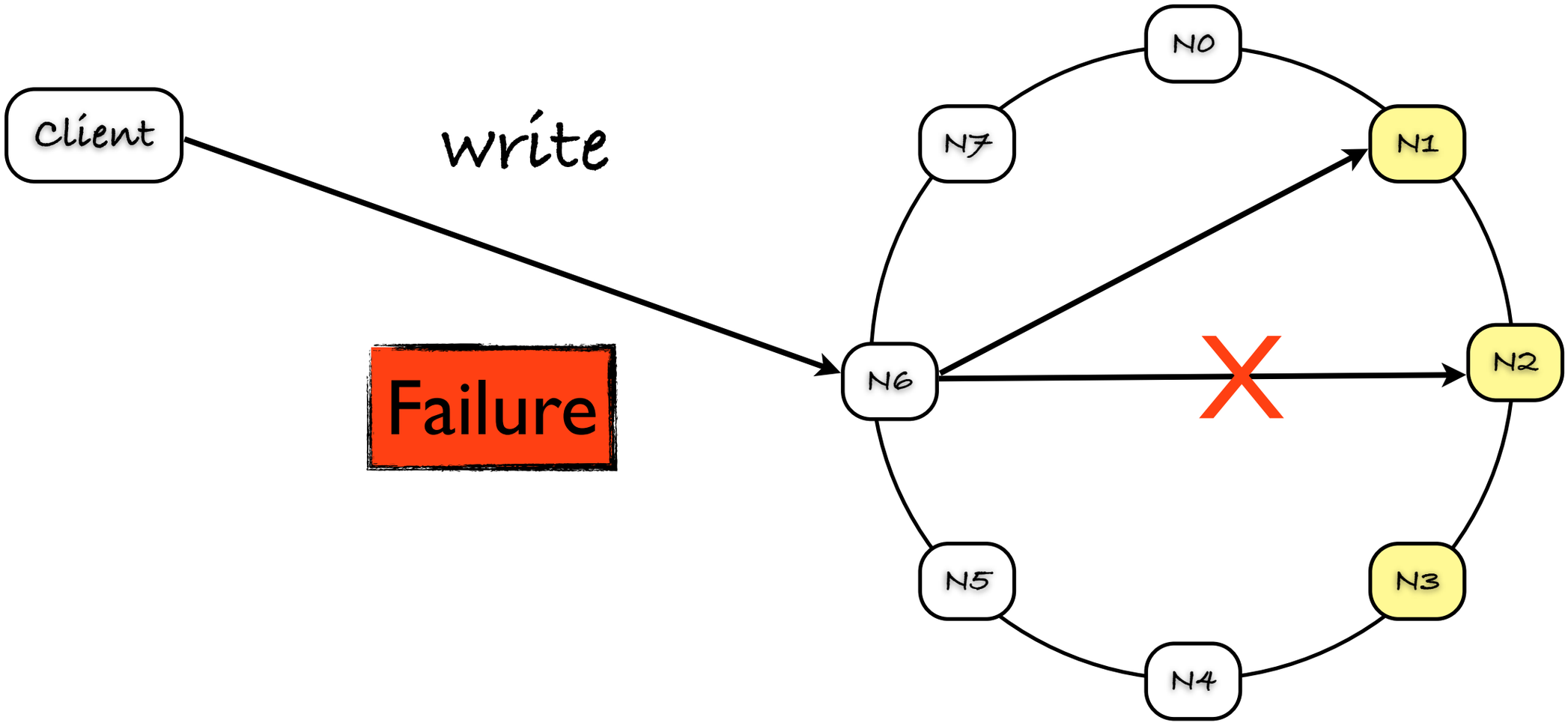
Consistency Level ONE



puis, en background...

RF = 3

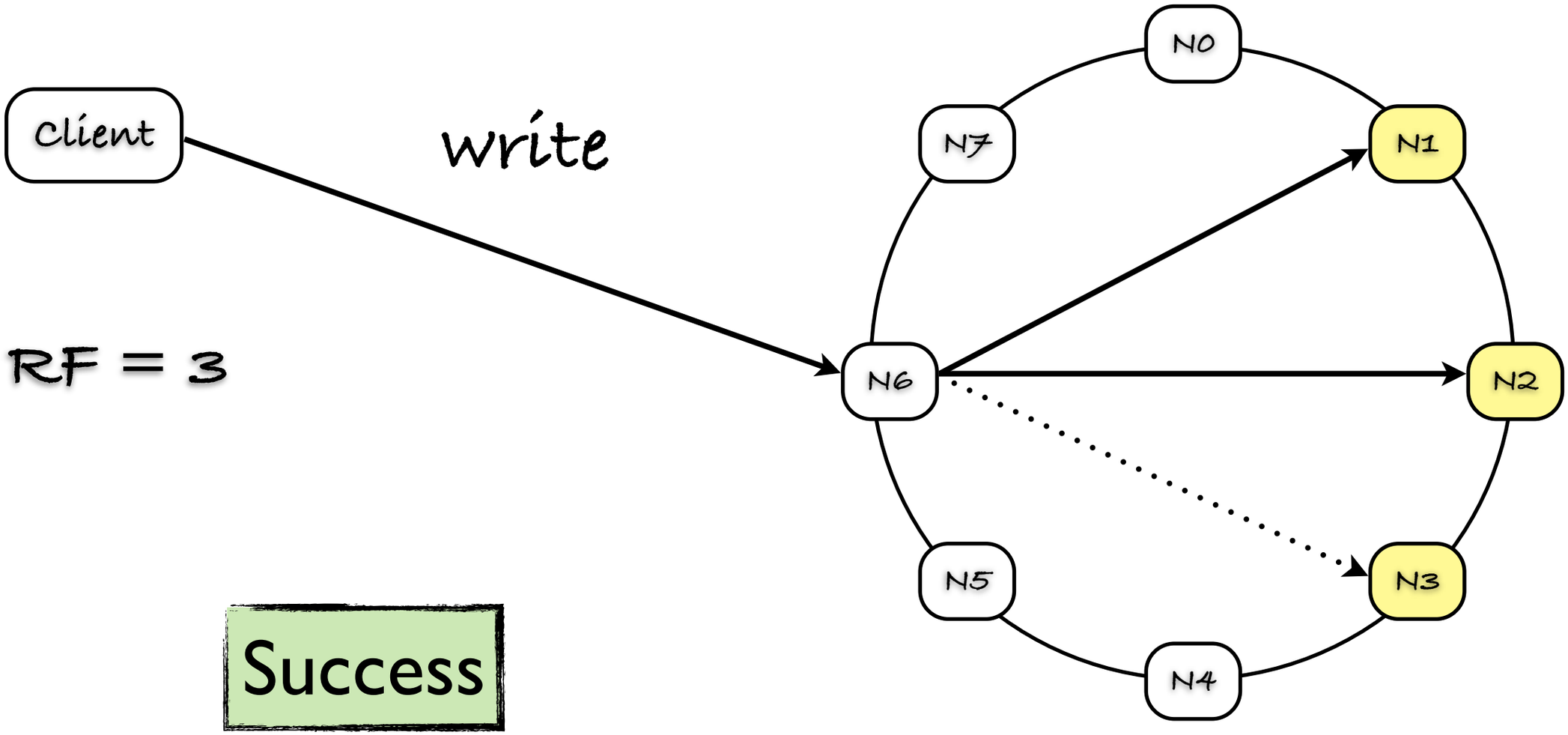
Consistency Level QUORUM



RF = 3

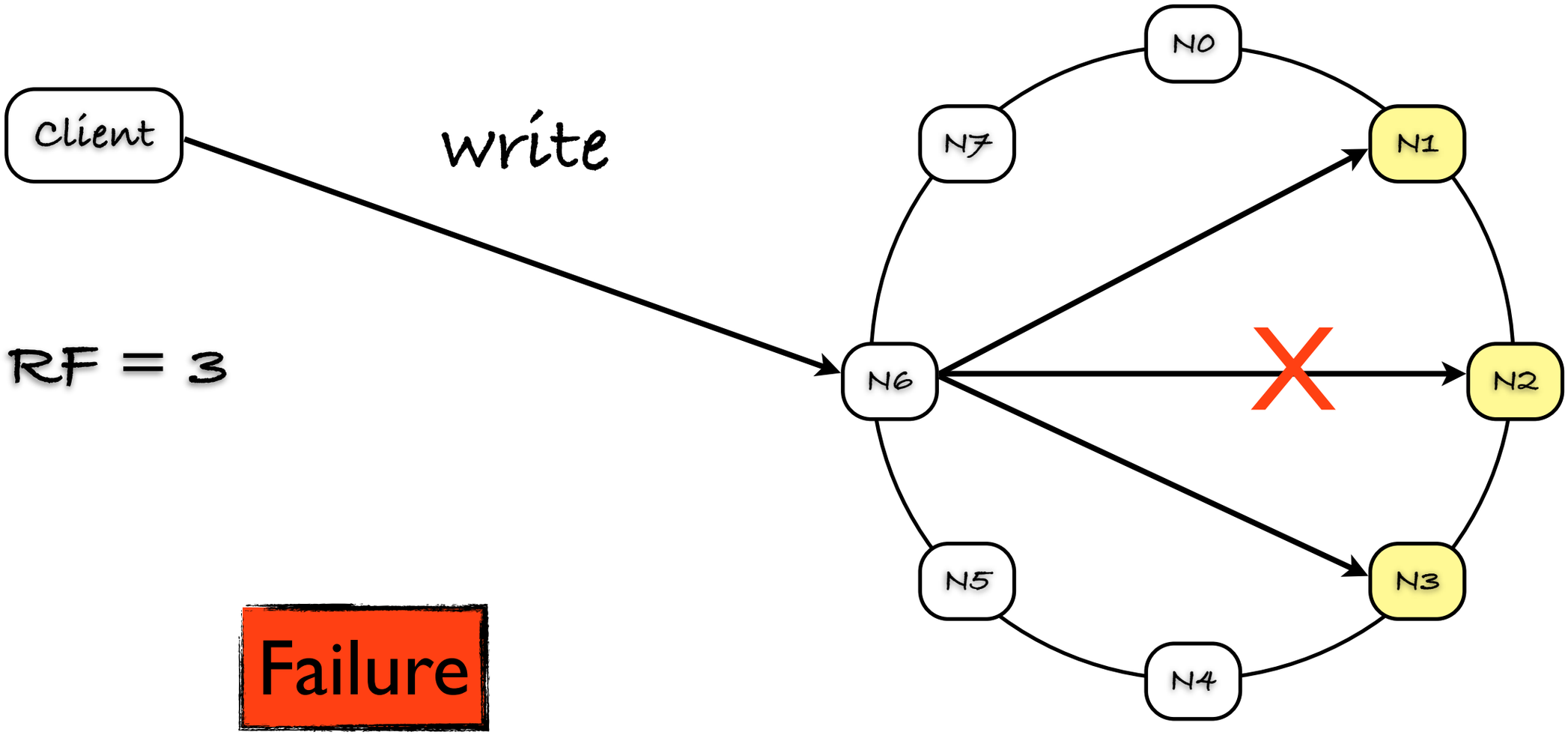
jaxio

Consistency Level QUORUM



Success

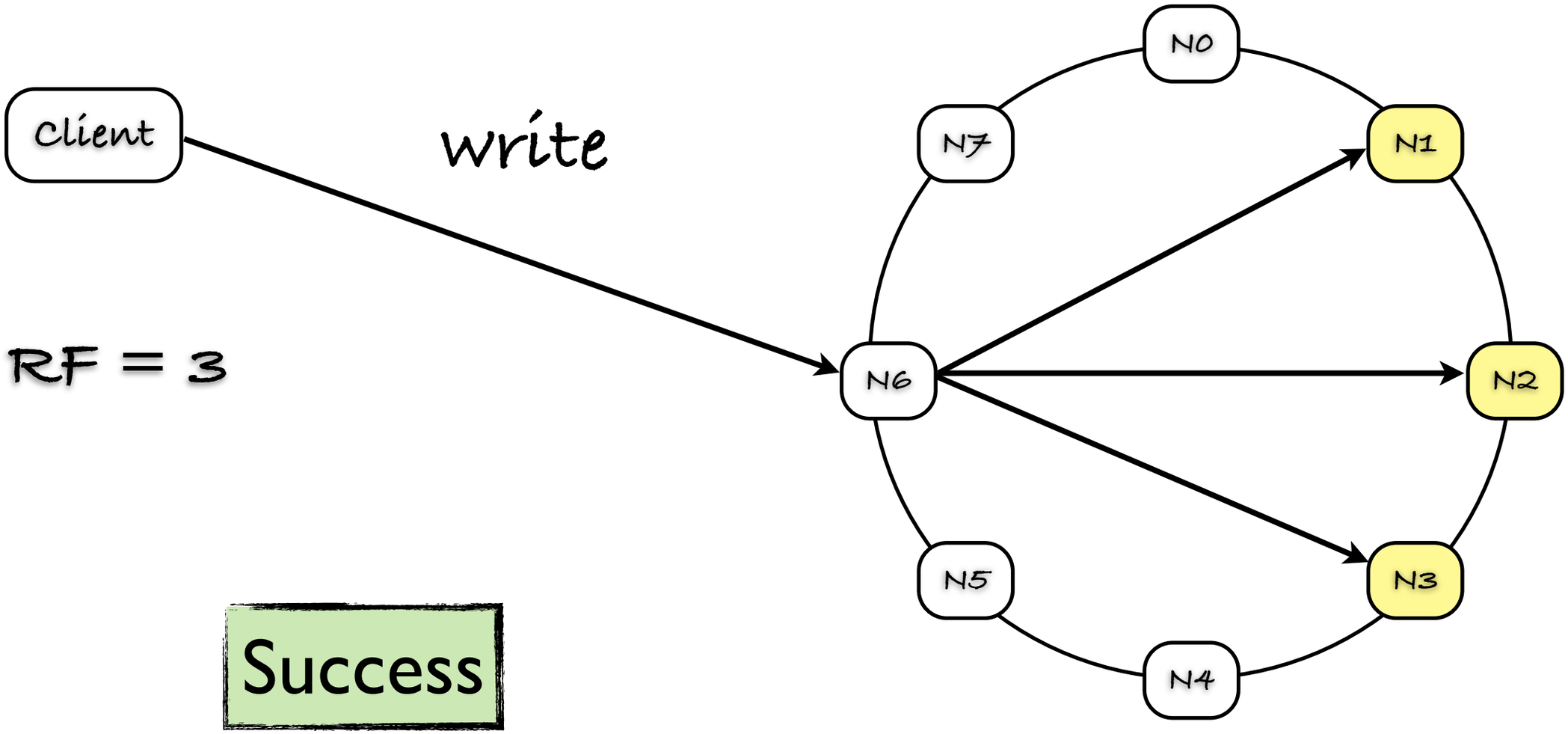
Consistency Level ALL



Failure

jaxio

Consistency Level ALL

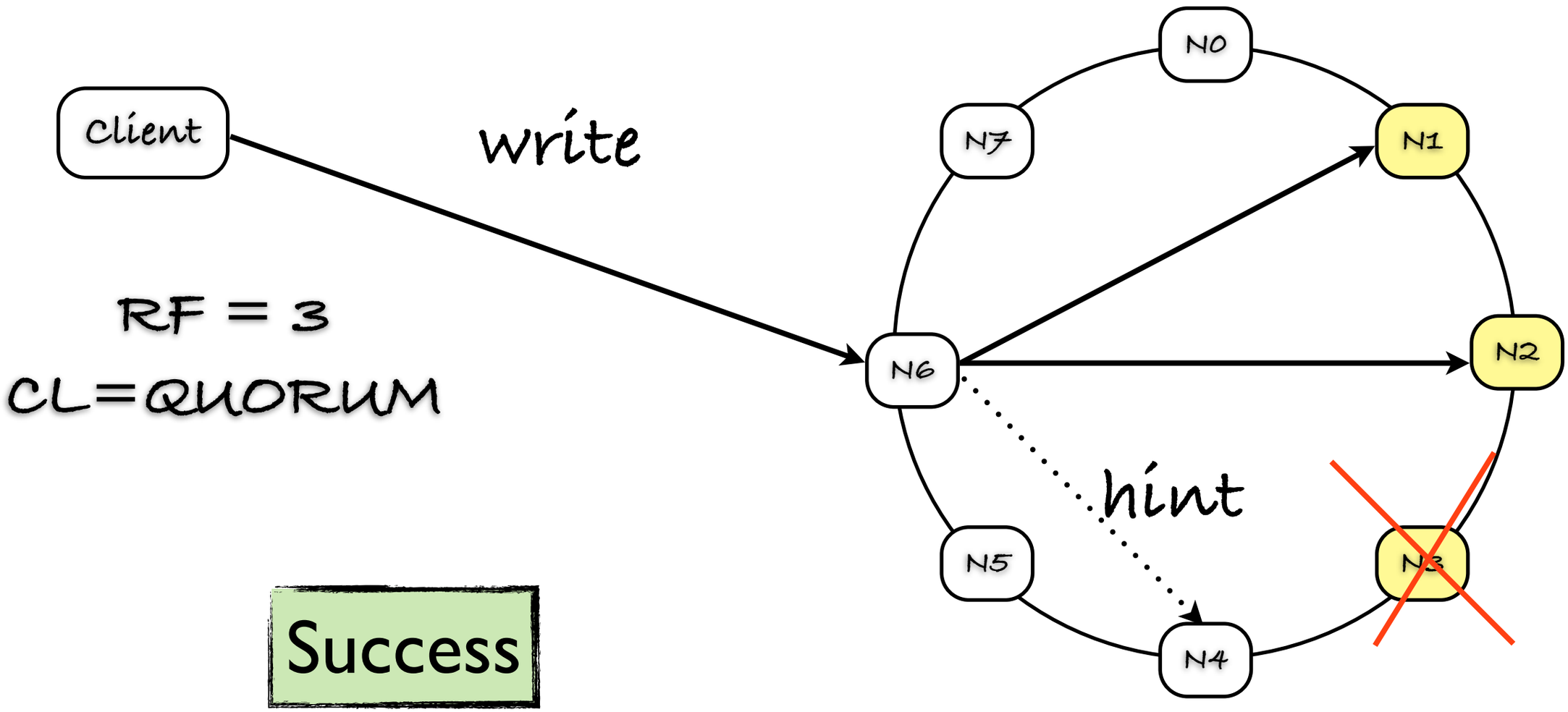


Success

Hinted Handoff

- Quid du noeud sur lequel nous n'avons pas pu écrire?
- Lorsqu'un noeud est indisponible le noeud coordinateur écrit la donnée sur un autre noeud
- Cet autre noeud est en charge de rejouer le write une fois que le noeud indisponible revient dans le ring

Hinted Handoff



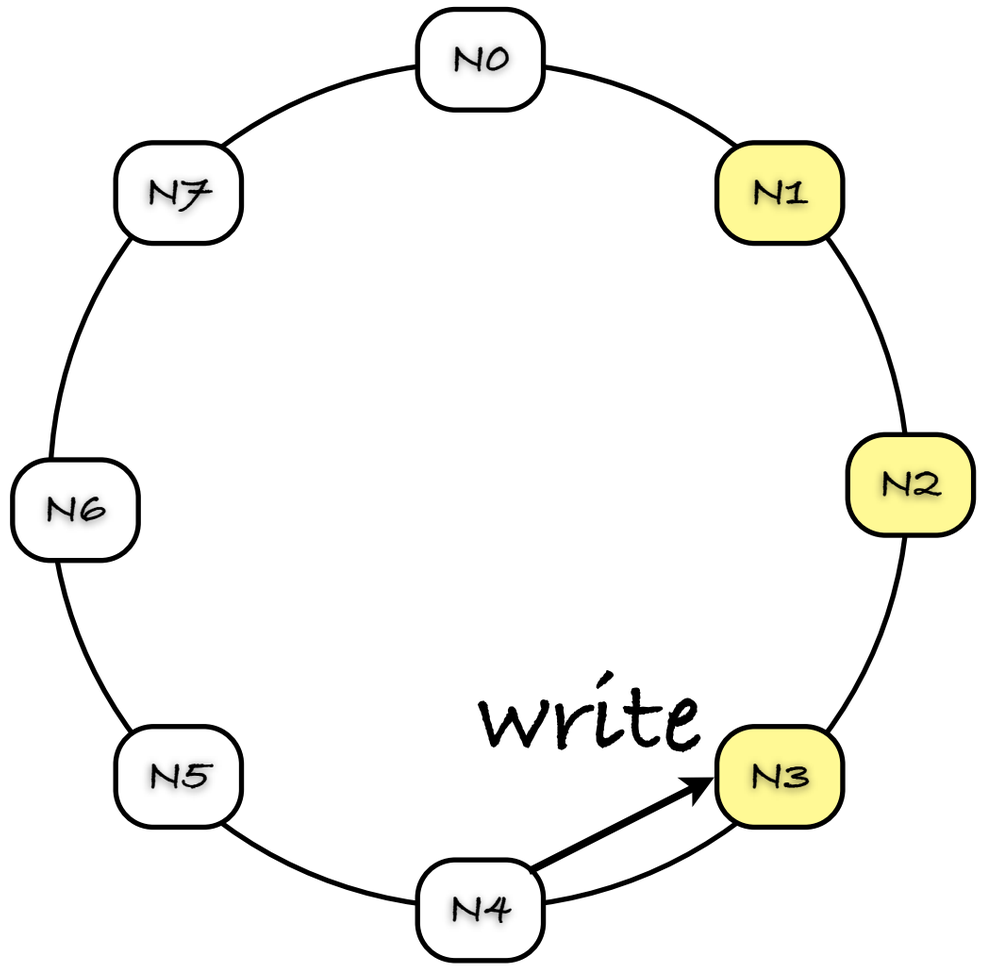
Success

Hinted Handoff

Client

RF = 3

CL=QUORUM



Lorsque N3 revient, N4 lui envoie la donnée

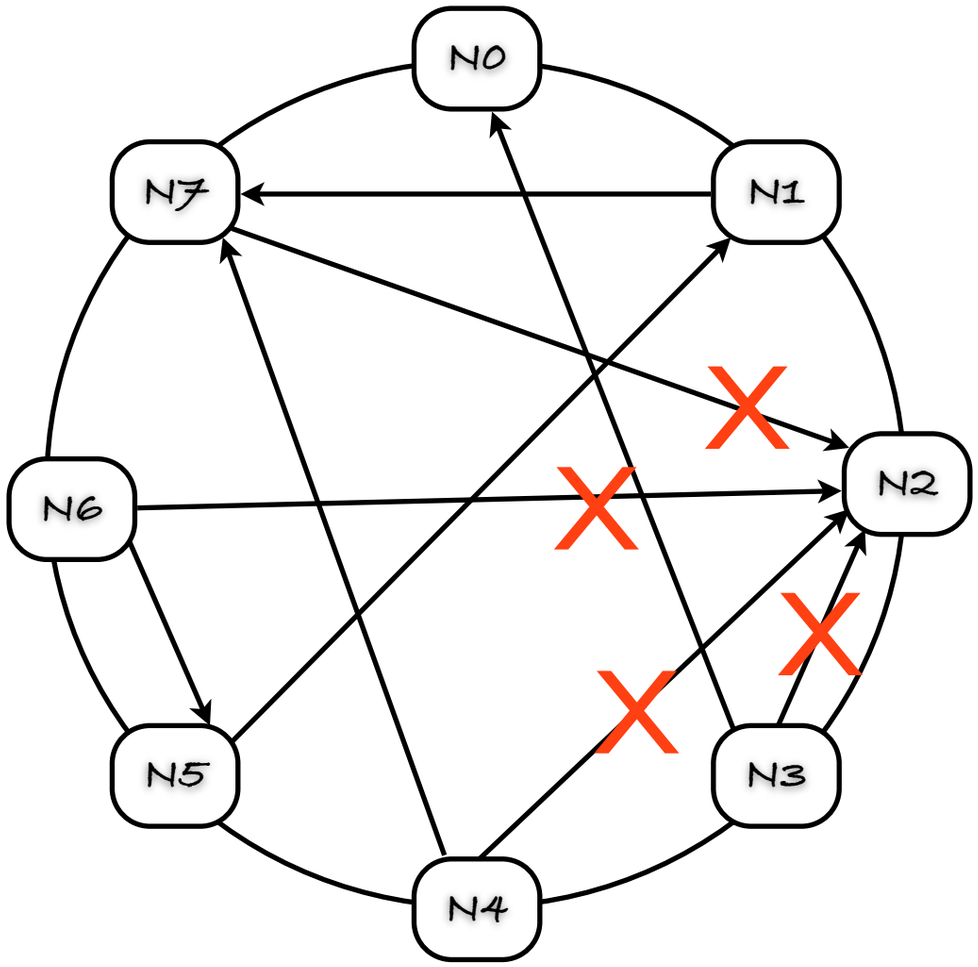
attention au max_hint_window_in_ms (1h par défaut)

jaxio

Gossip

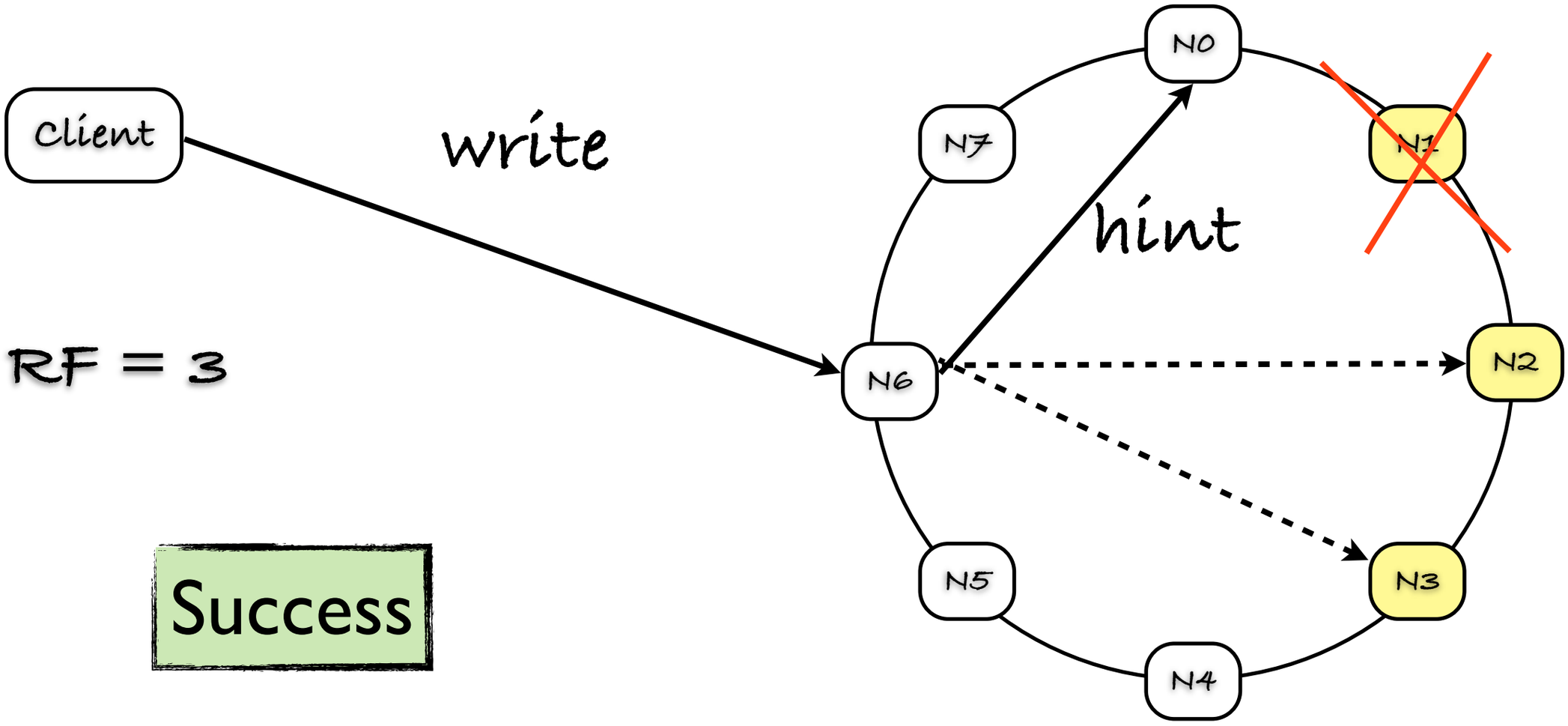
- Comment un noeud sait-il qu'un autre noeud revient dans le ring?
- Les noeuds échangent des meta-données entre eux sur l'état du ring
- Protocole peer to peer

Gossiping



- Chaque seconde, chaque noeud contacte entre 1et 3 noeuds
  - Conçu pour éviter la formation d'ilots indépendants
  - Conçu pour que l'information se propage vite

Consistency Level ANY



Success

Intérêt? Performance

Révision / Questions

- Consistent Hashing
- Token
- Replication Factor
- Replica placement strategy
- Consistency Level
- Hinted Handoff
- Gossip protocol

Retour sur Ecriture

create

password
xyz
10000

update

password
abc
20000

update

password
tralala
30000

3 valeurs pour 1 column

Le timestamp le plus récent fait foi

Mais les nodes ont-ils bien la dernière version?

jaxio

Noeud down revenant dans le ring

- Imaginez un noeud KO pendant une durée > durée de vie d'un hint
  - Il doit récupérer les données
  - Node repair (Anti Entropy Repair)

Lecture

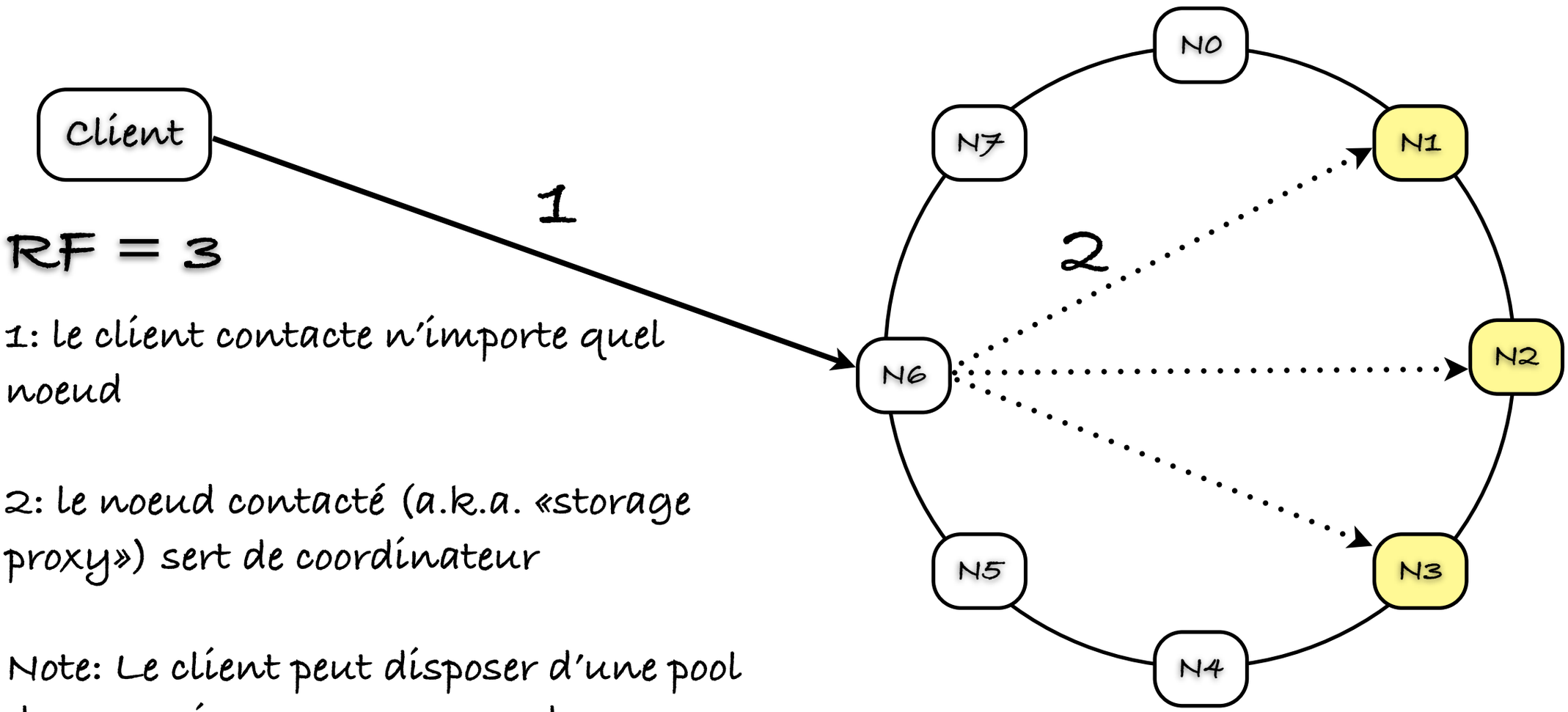
- Comment le client s'assure-t-il que la donnée lue est bien **la plus récente**?
  - Attendre une réponse de tous les serveurs impliqués et comparer?
  - Se fier à la première réponse?
- En pratique, cela dépend du use-case
- Solution: Tuneable Consistency

jaxio

Tuneable Consistency

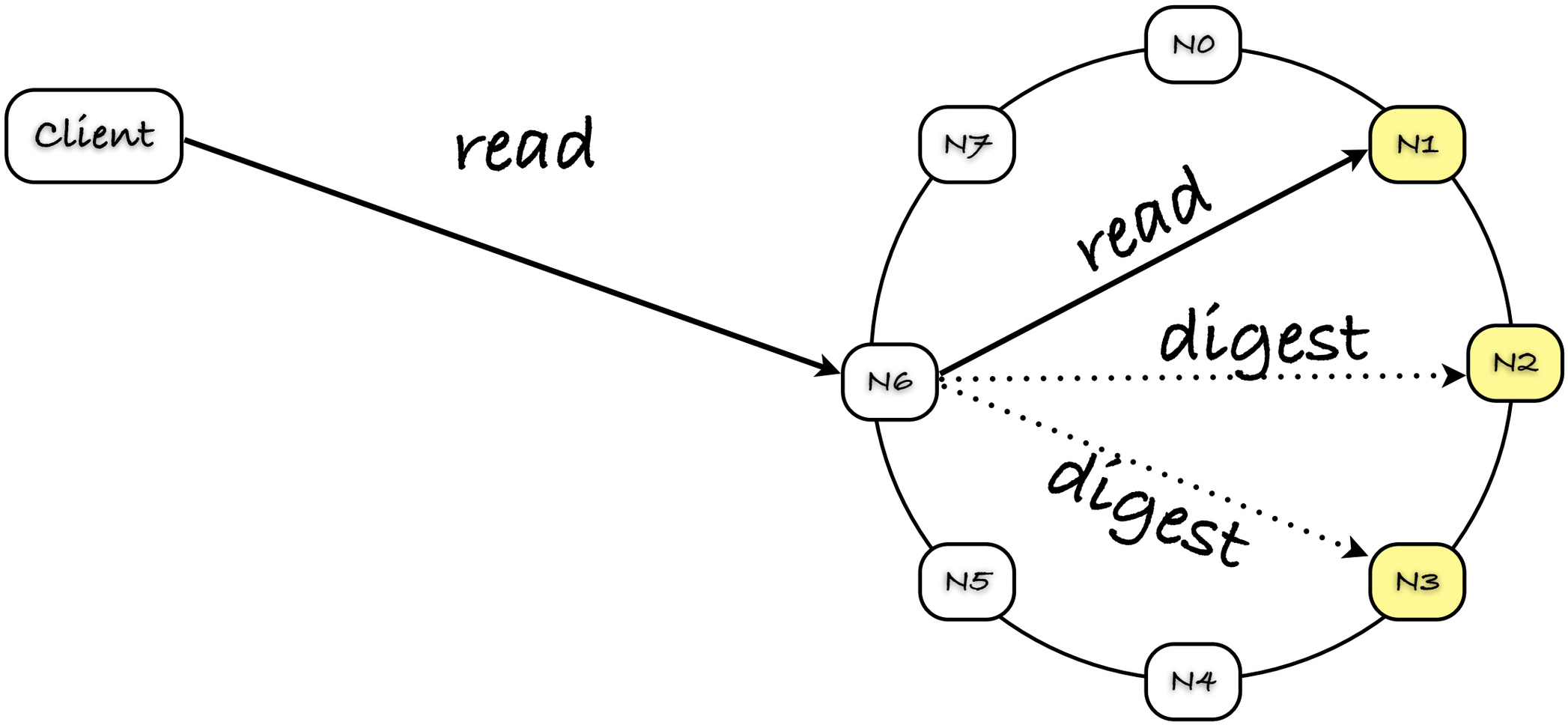
- En lecture, le client indique le Consistency Level désiré
  - **ONE**: je veux la donnée d'une des replicas, je m'en fiche si ce n'est pas la plus récente
  - **QUORUM**: je veux la donnée la plus récente parmi une majorité de replicas, je me dis que la majorité aura peut être raison
  - **ALL**: je veux la donnée la plus récente parmi toutes les replicas, je serai certain d'avoir la plus récente

Lecture



Le client ne sait pas où la donnée sera lue

Consistency Level ONE

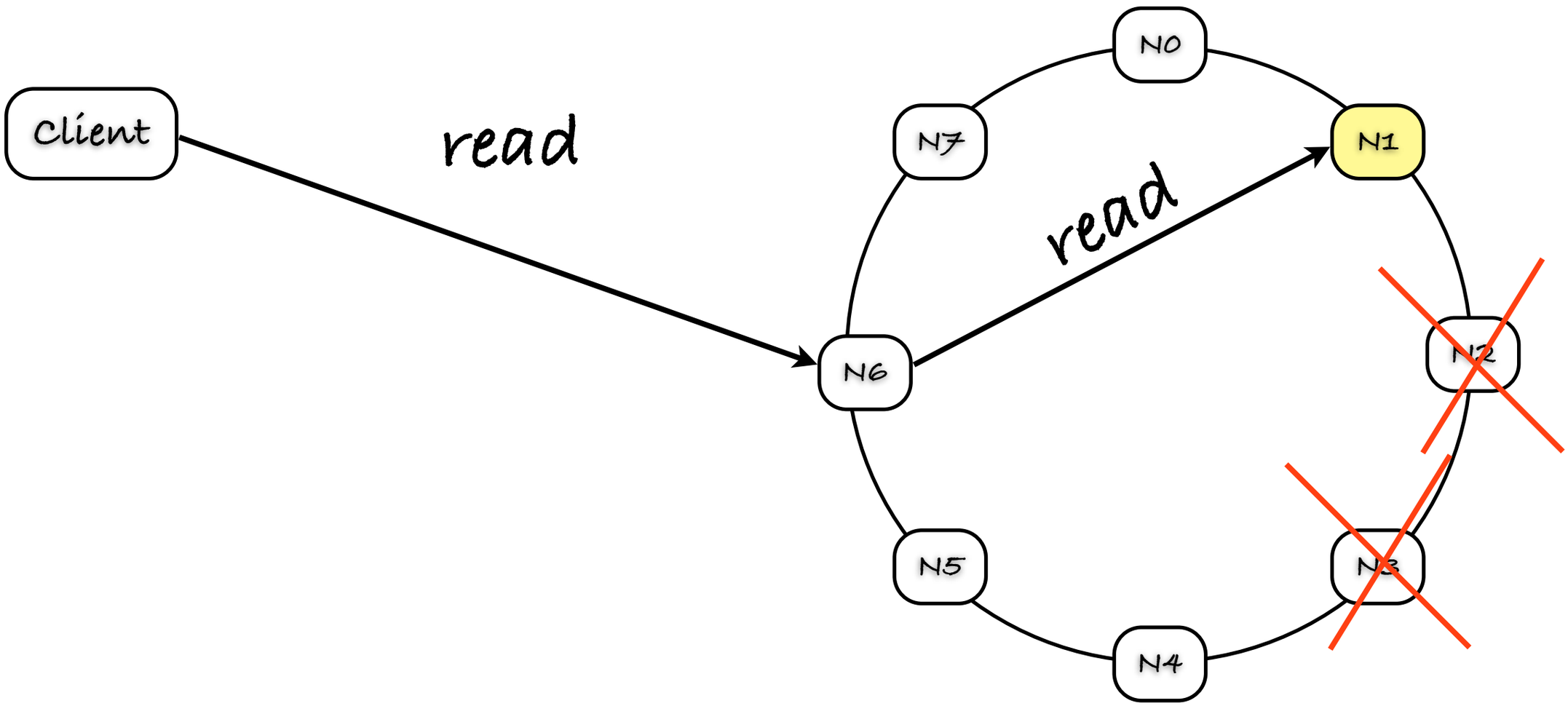


OK, mais N1 avait-il la version la plus récente?

.....➤ read en asynchrone (md5), puis **read repair** si besoin et si configuré

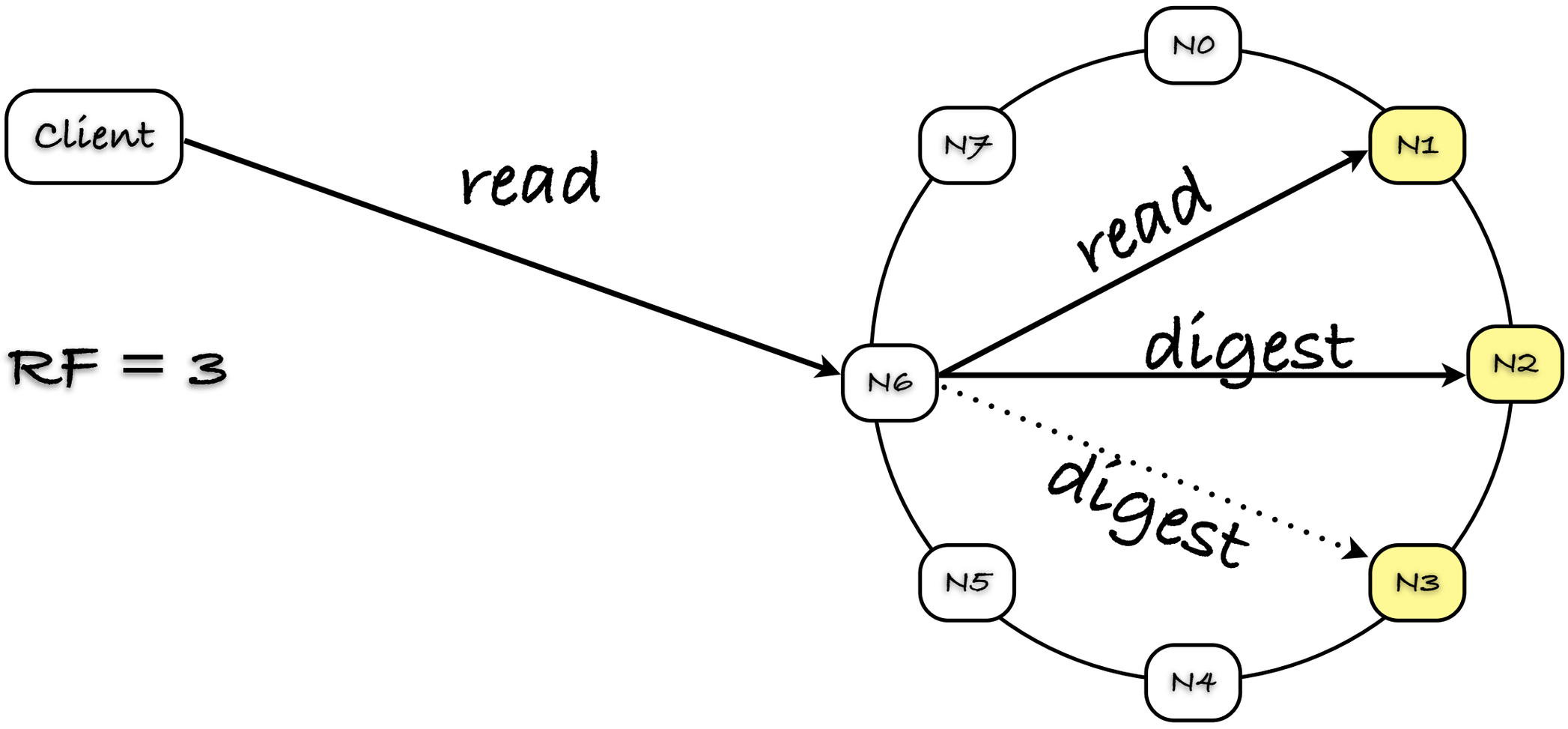
jaxio

Consistency Level ONE



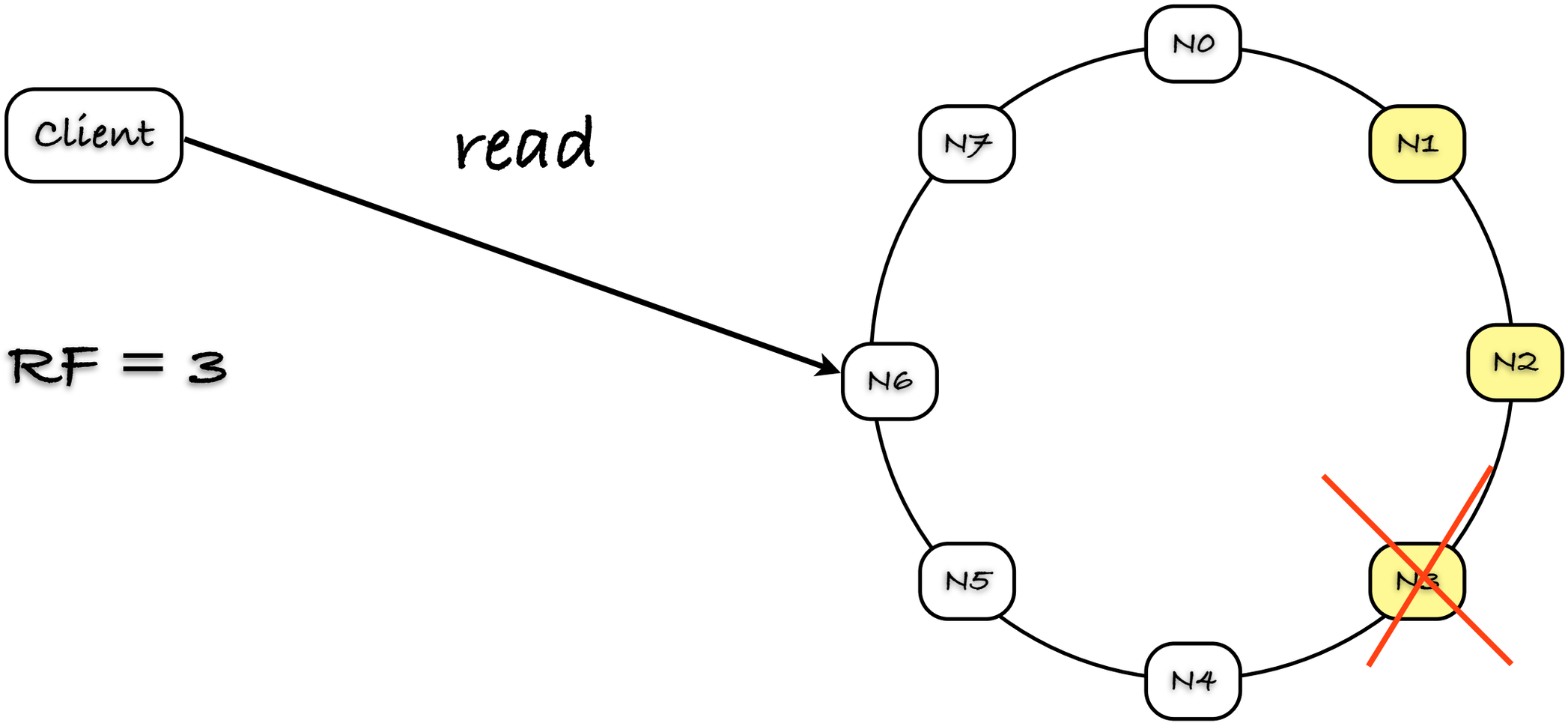
OK, mais N1 avait-il la version la plus récente?

Consistency Level QUORUM



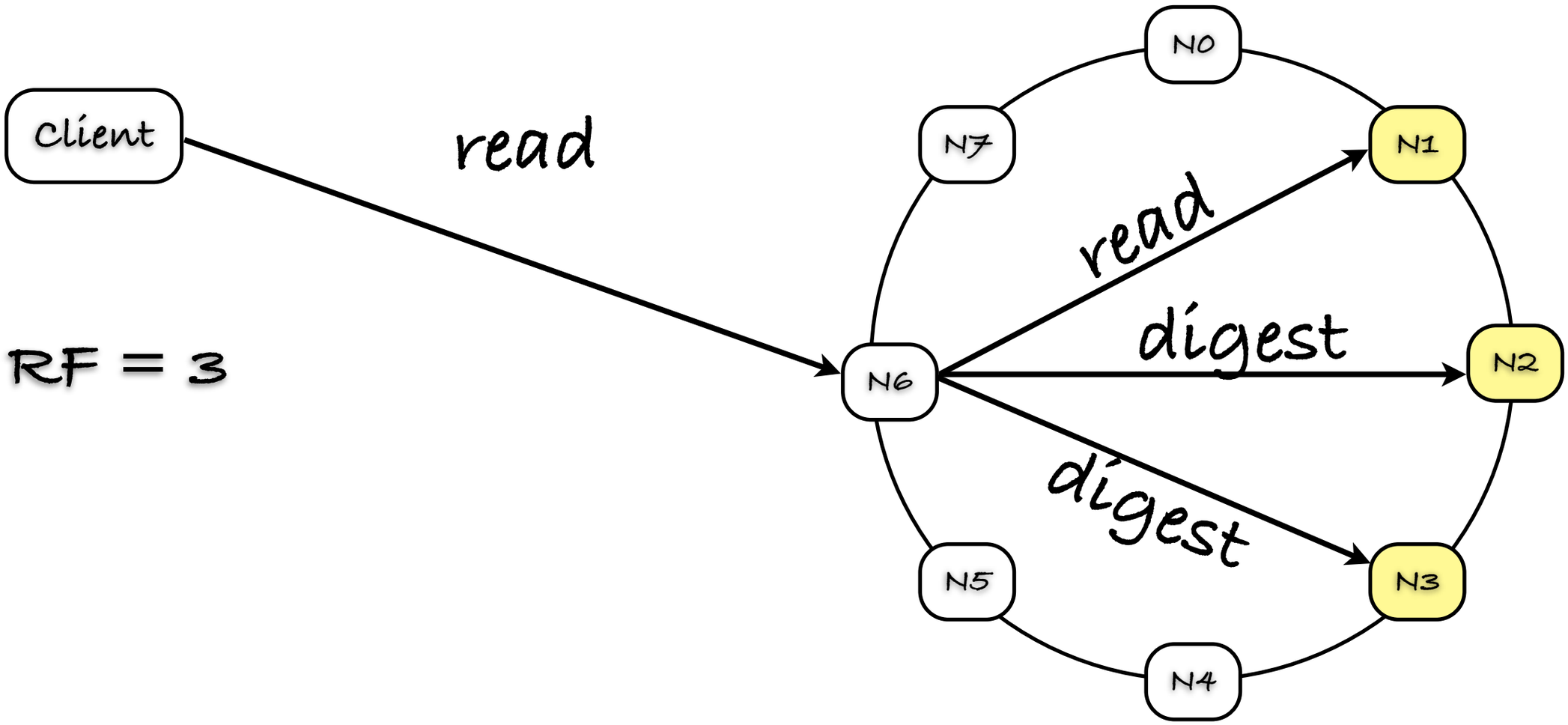
Retourne version la plus récente entre N1 & N2
Mais N3 avait-il une version encore plus récente?

Consistency Level ALL



KO, même pas besoin d'essayer de lire N1 et N2

Consistency Level ALL



OK, retourne version la plus récente

Consistance forte

- Comment être certain que la donnée lue est la plus récente?
  - formule ***W + R > RF***

    W: nb de replica contactées en écriture

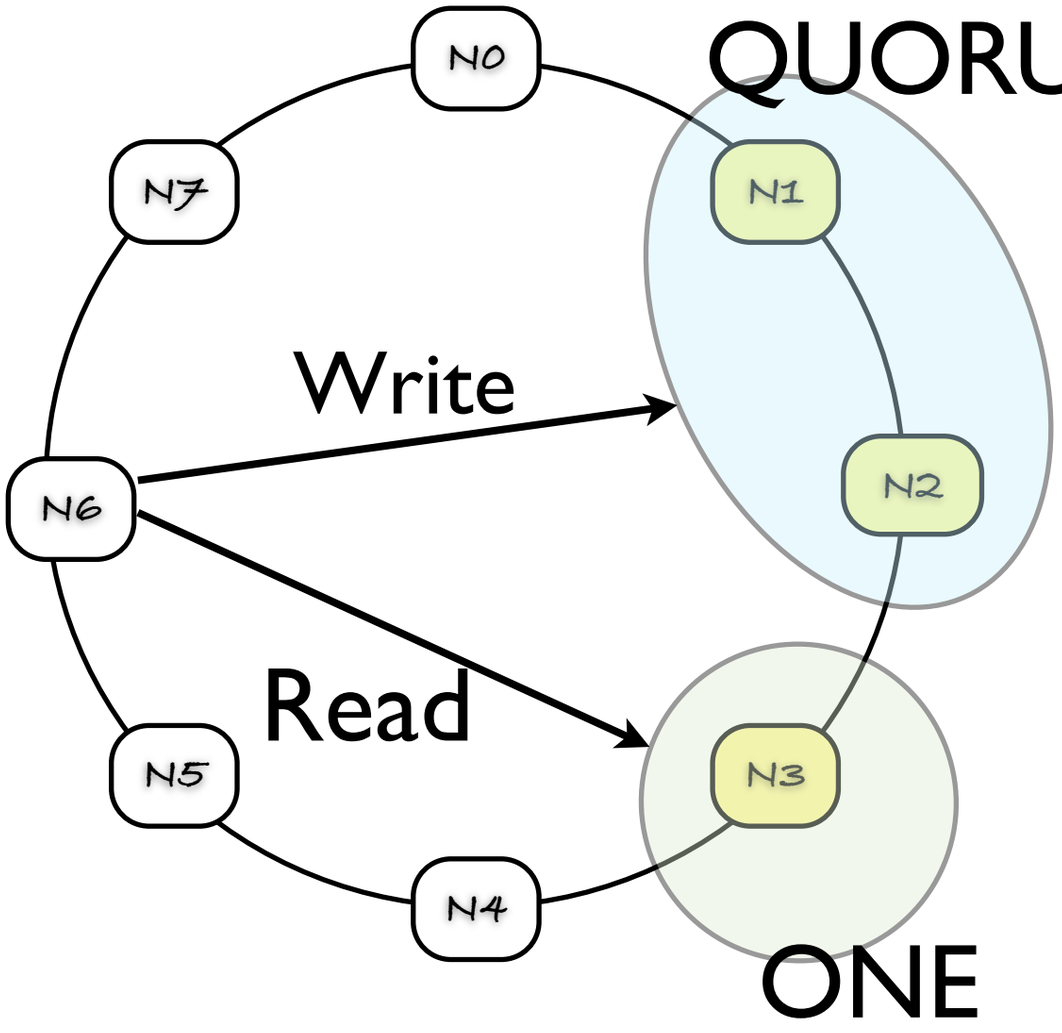
    R: nb de replica contactées en lecture

    RF: replication factor

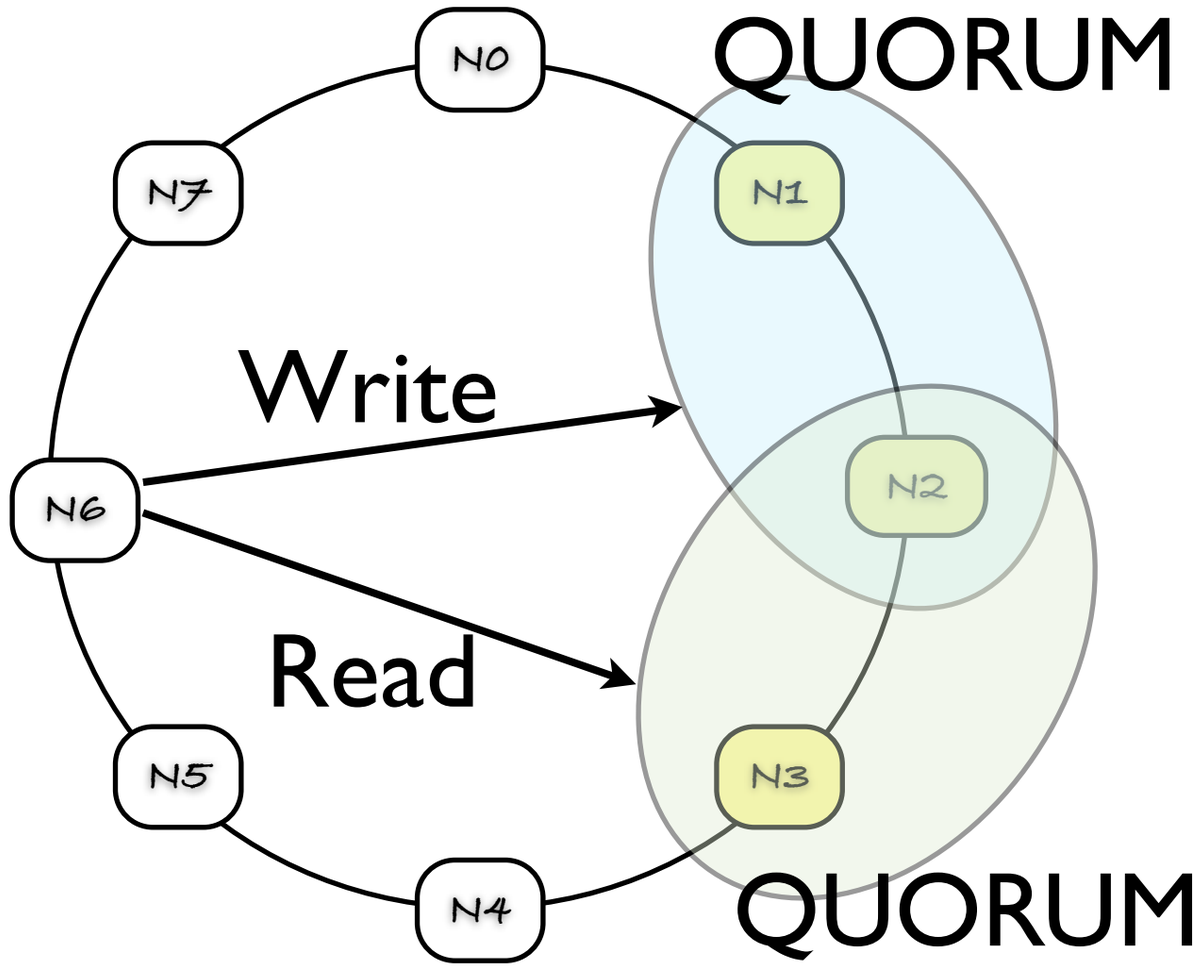
- Cas trivial
  - écriture en ALL et lecture en ONE

Consistency

RF = 3



«Weak»



Strong

jaxio

En pratique

- La réplication fonctionne très bien (< qq ms)

jaxio

Noeud down revenant dans le ring

- Imaginons qu'un noeud ait loupé un delete pendant son down time
  - La donnée non supprimée ne doit pas être prise par le autres noeuds (qui l'ont supprimée) pour une nouvelle donnée!
  - Grace Period des Tombstone (10j par défaut)
  - Anti-Entropy Repair (nodetool repair)

jaxio

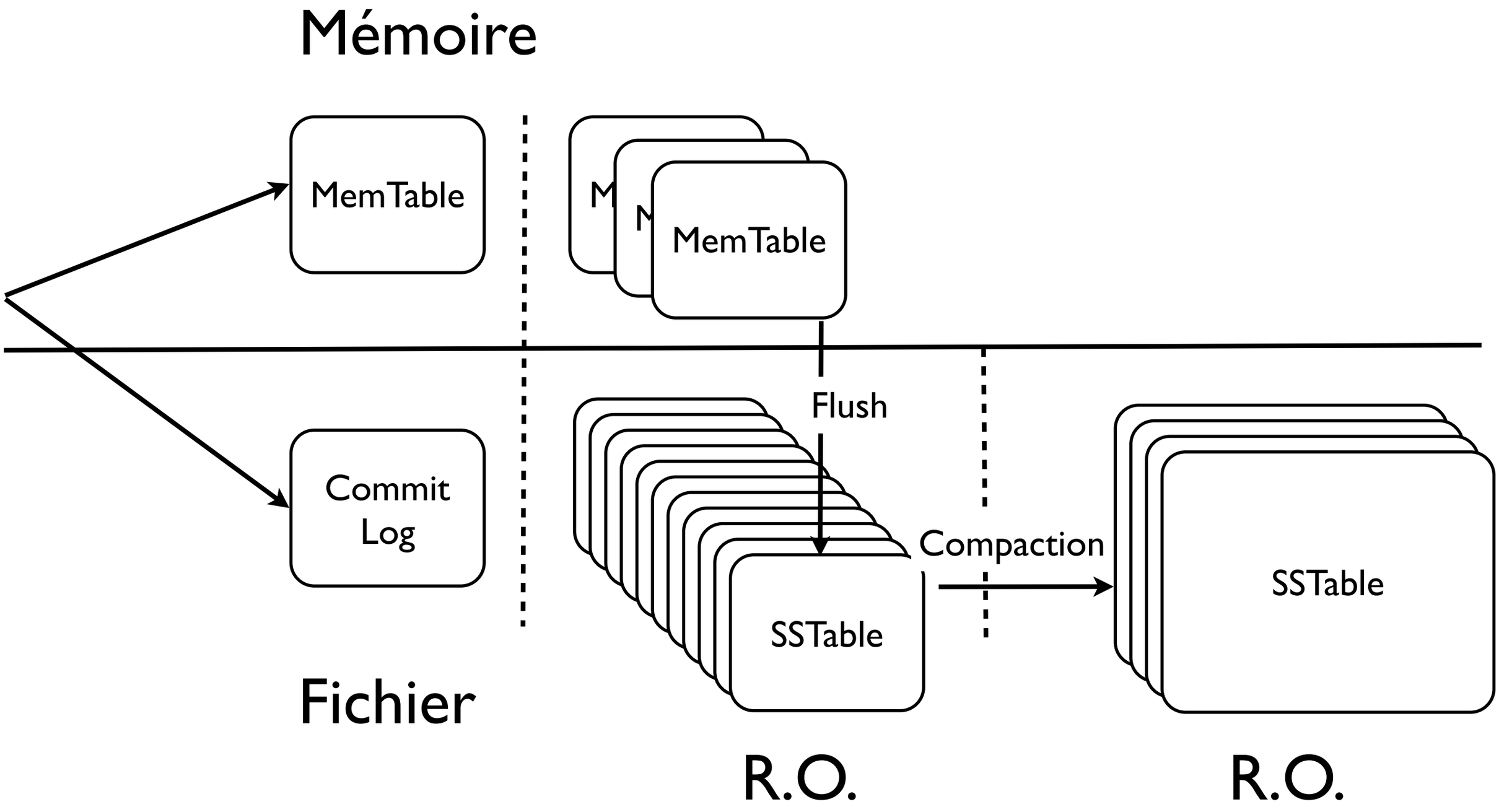
Révision / Questions

- Storage proxy
- Read repair
- Strong consistency: R+W > RF
- Update
- Tombstone
- Anti-Entropy

Performance en écriture

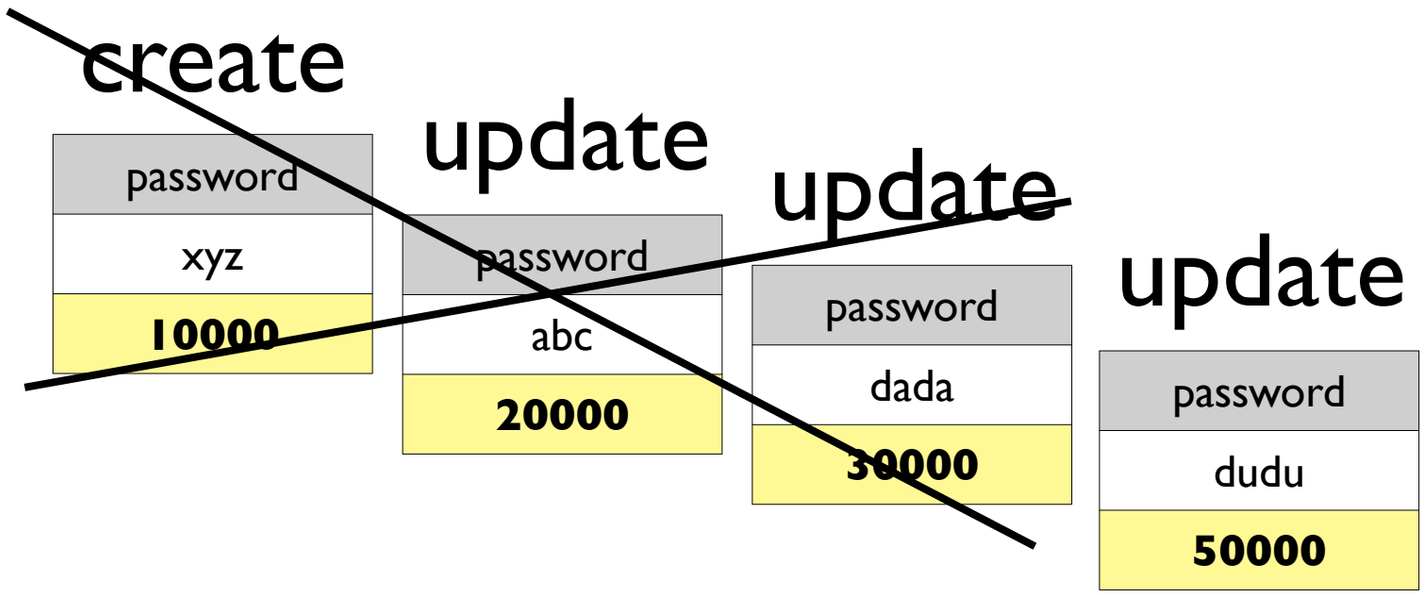
- En écriture, Cassandra est extrêmement rapide, pourquoi?
  - Ecrit dans le fichier Commit log (append only)
  - Ecrit en mémoire (MemTable)
  - Pas d'update...

Performance en écriture



jaxio

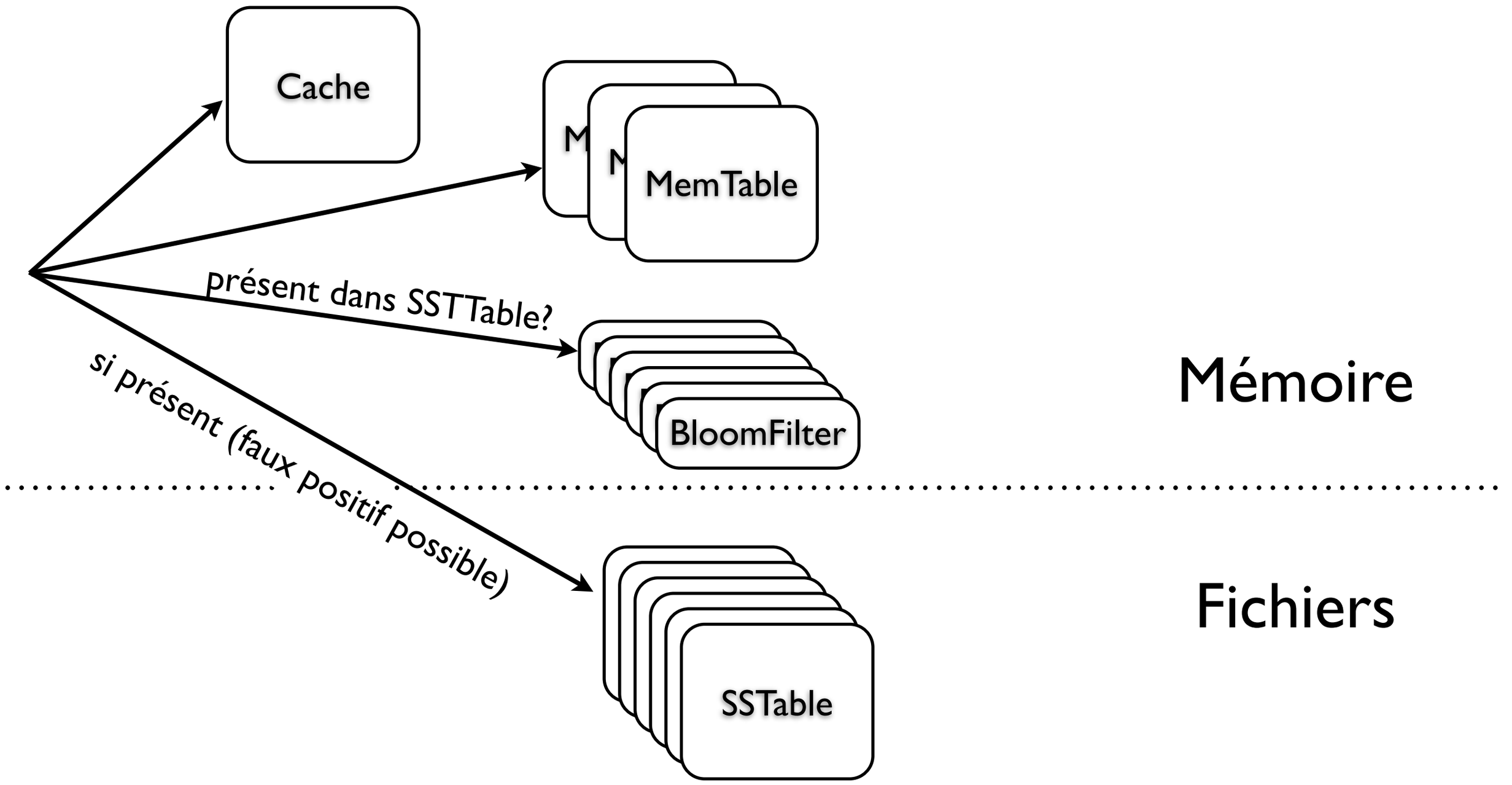
Compaction


create
update
update
update
password
xyz
10000
password
abc
20000
password
dada
30000
password
dudu
50000

Performance en lecture

- En lecture, Cassandra est quasiment aussi performant
  - Contact en priorité le noeud le plus «proche» de la donnée
  - Index pour row key et col. name + Cache
  - Bloom Filter

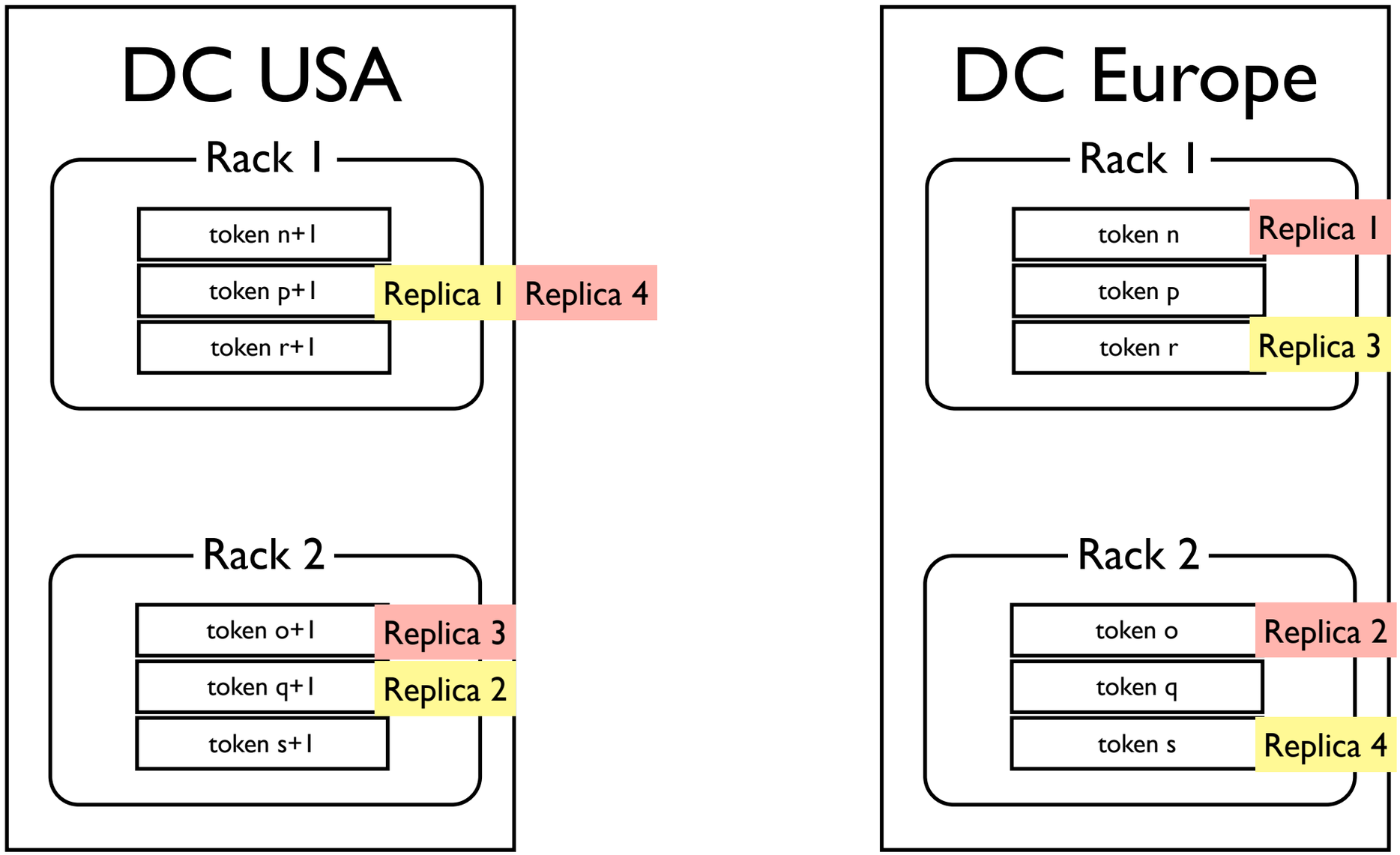
Performance en lecture



Snitch

- Une adresse IP => Un endroit (DataCenter, Rack)
- Utile pour répartir les replicas entre les racks, entre les DataCenters
- Le ‘dynamic snitch’ prend aussi en compte le temps de réponse des noeuds

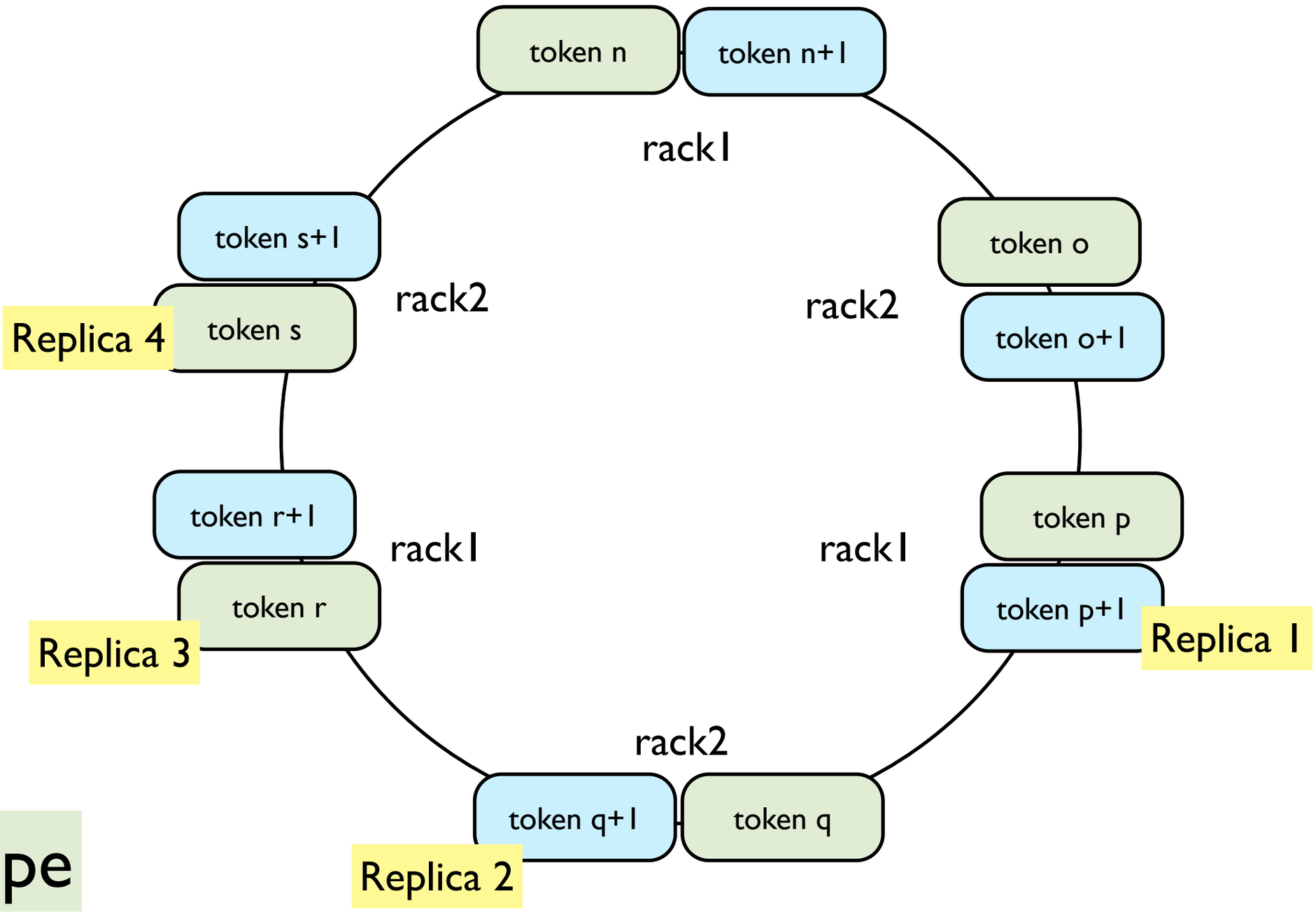
Data Centers / Racks



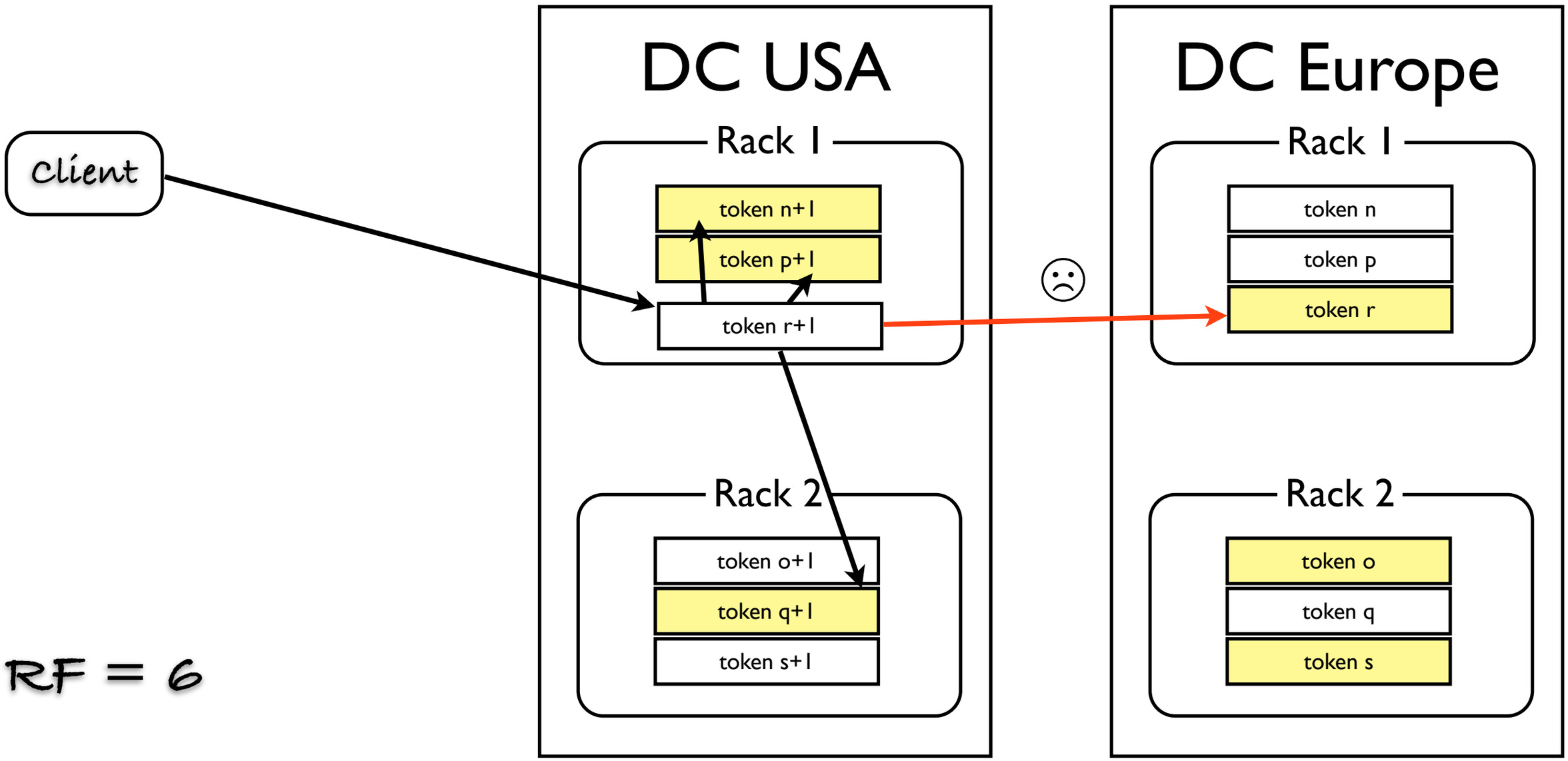
RF = 4

jaxio

Tokens forment toujours un ring

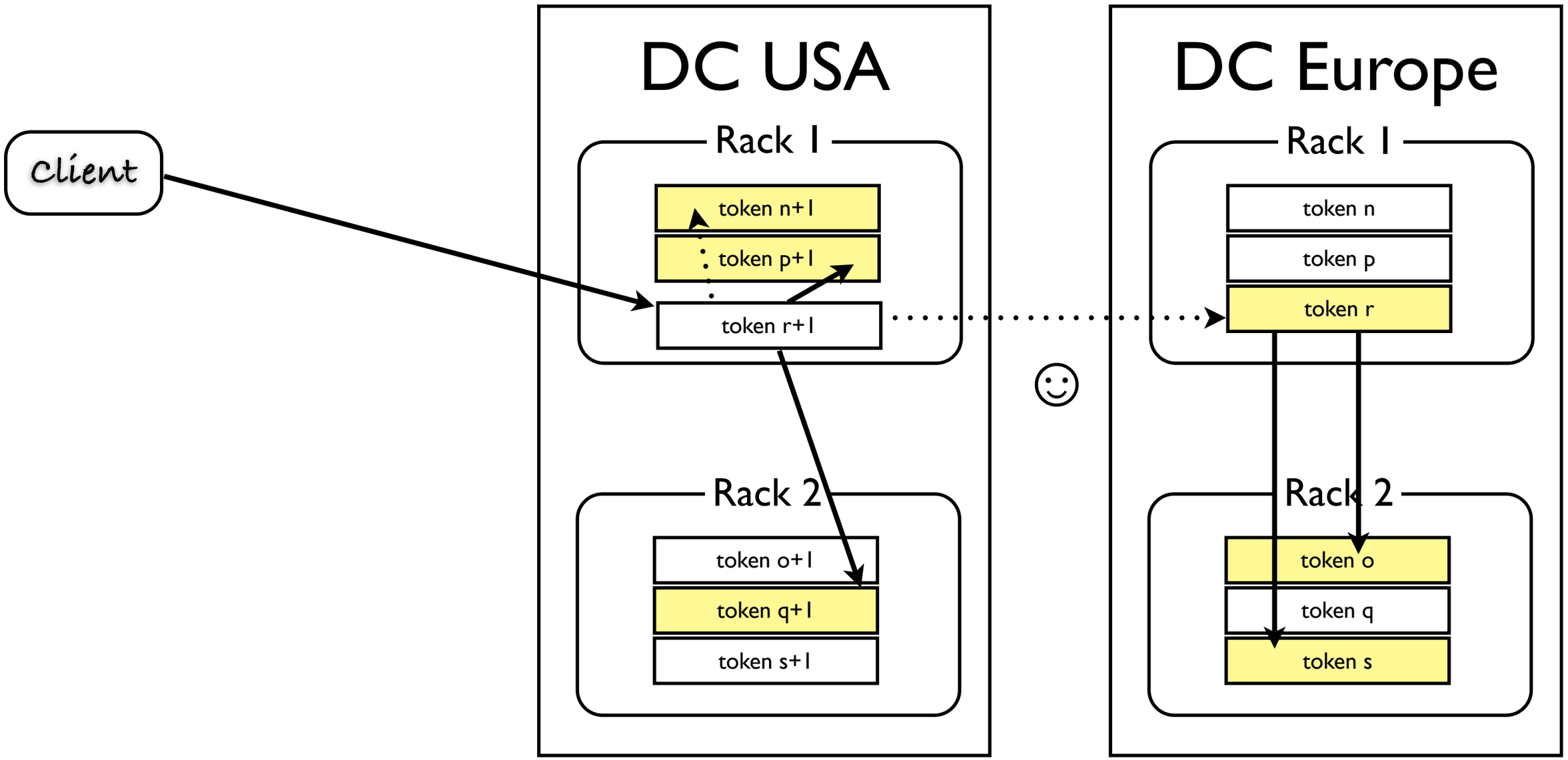


QUORUM pas toujours adapté



RF = 6

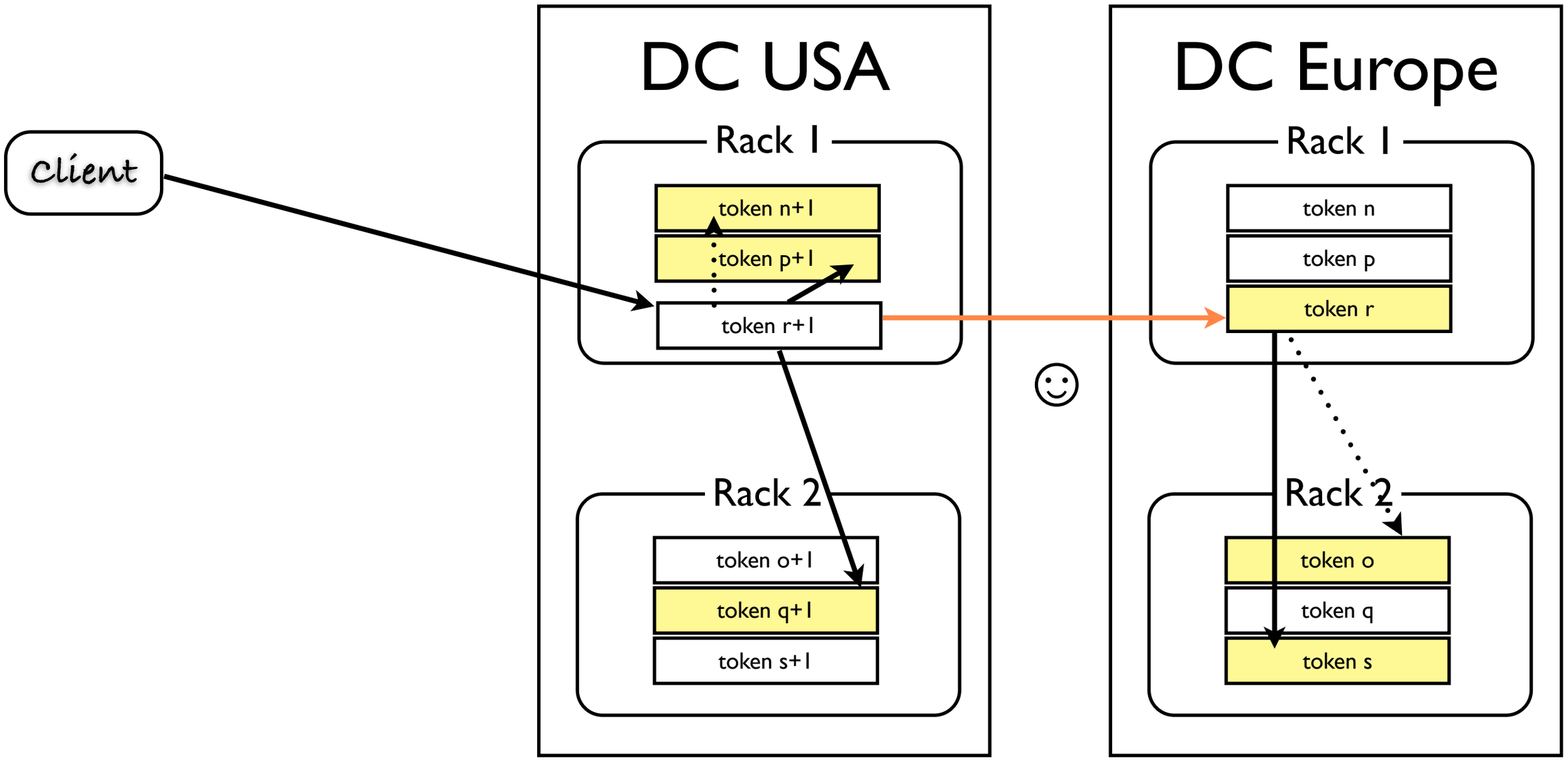
LOCAL_QUORUM



QUORUM sur 1 seul Data Center

jaxio

EACH_QUORUM



QUORUM sur chacun des Data Center

jaxio

Sujets non abordés...

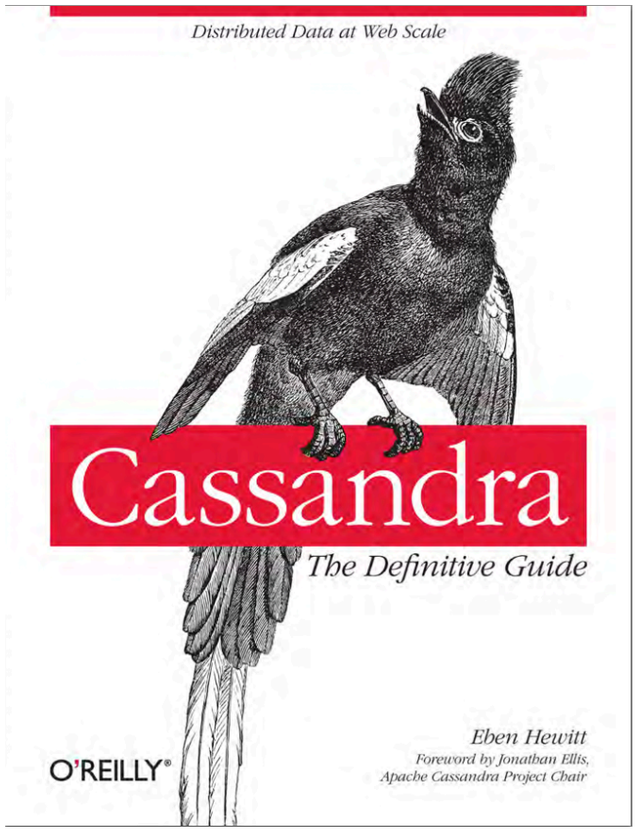
- Ajouter/Supprimer/Remplacer un noeud
- OrderPreserving Partitionner
- Major Compaction
- Failure detector intégré au Gossip
- Tunning / Monitoring
- Tools
- Client API (Hector, etc.), CQL
- etc..

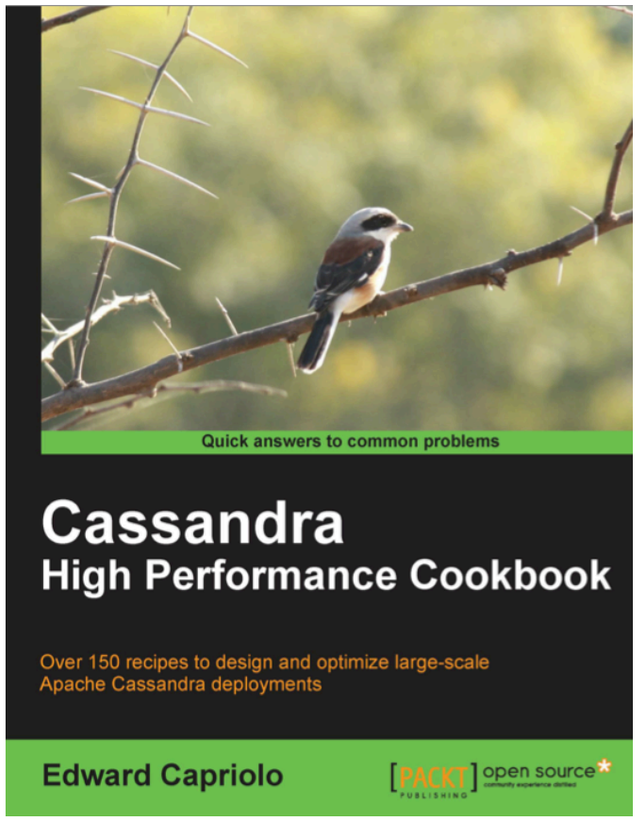
Questions?

- Et peut-être des réponses :)

jaxio

Pour approfondir...





http://www.datastax.com/docs/1.0/index

http://wiki.apache.org/cassandra/

http://ria101.wordpress.com/2010/02/24/hbase-vs-cassandra-why-we-moved/

jaxio