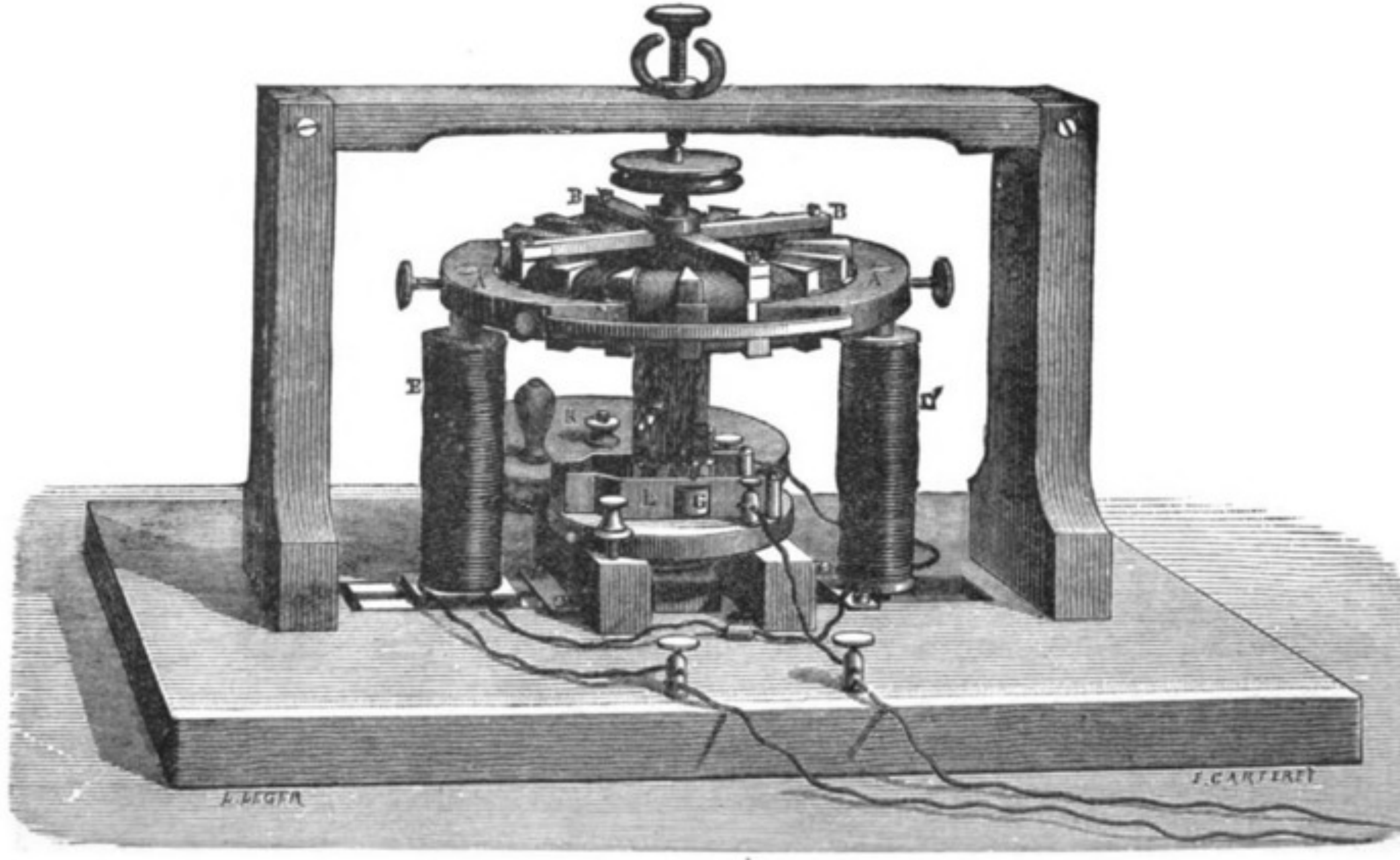


Dynamo concepts in depth.

Pavlo Baron, codecentric AG

entwickler.press
Pragmatische IT-Architektur
Schlanke Technologien und Konzepte für dynamische Unternehmen
Pavlo Baron

pavlo BARON
BIG DATA
FÜR IT-ENTSCHEIDER
RIESIGE DATENMENGEN UND MODERNE TECHNOLOGIEN GEWINNBRINGEND NUTZEN
EXTRA: Mit kostenlosem E-Book
HANSER

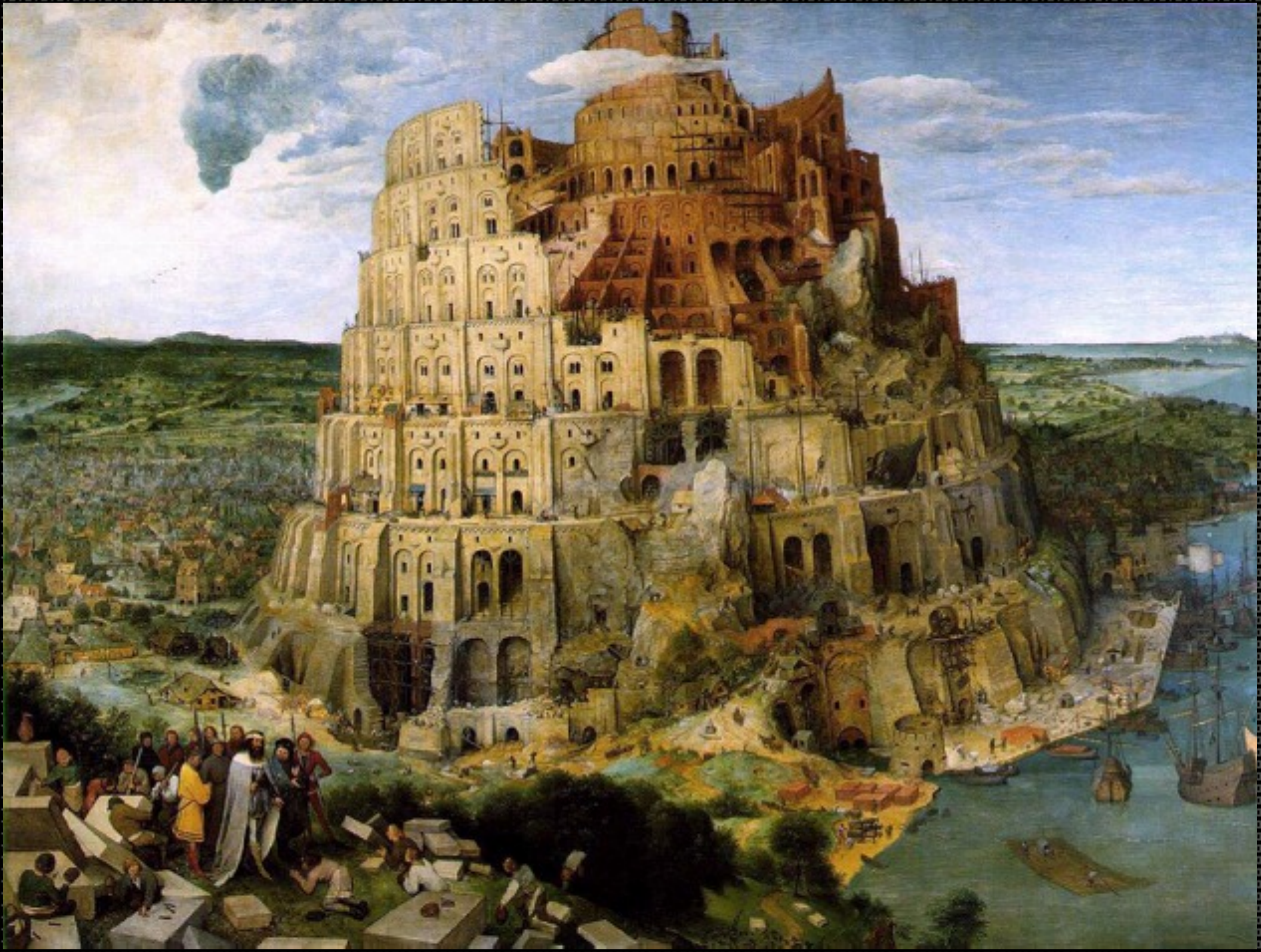
Pavlo Baron

pavlo.baron@codecentric.de
@pavlobaron

The shopping cart case

The 2 AM alarm call case

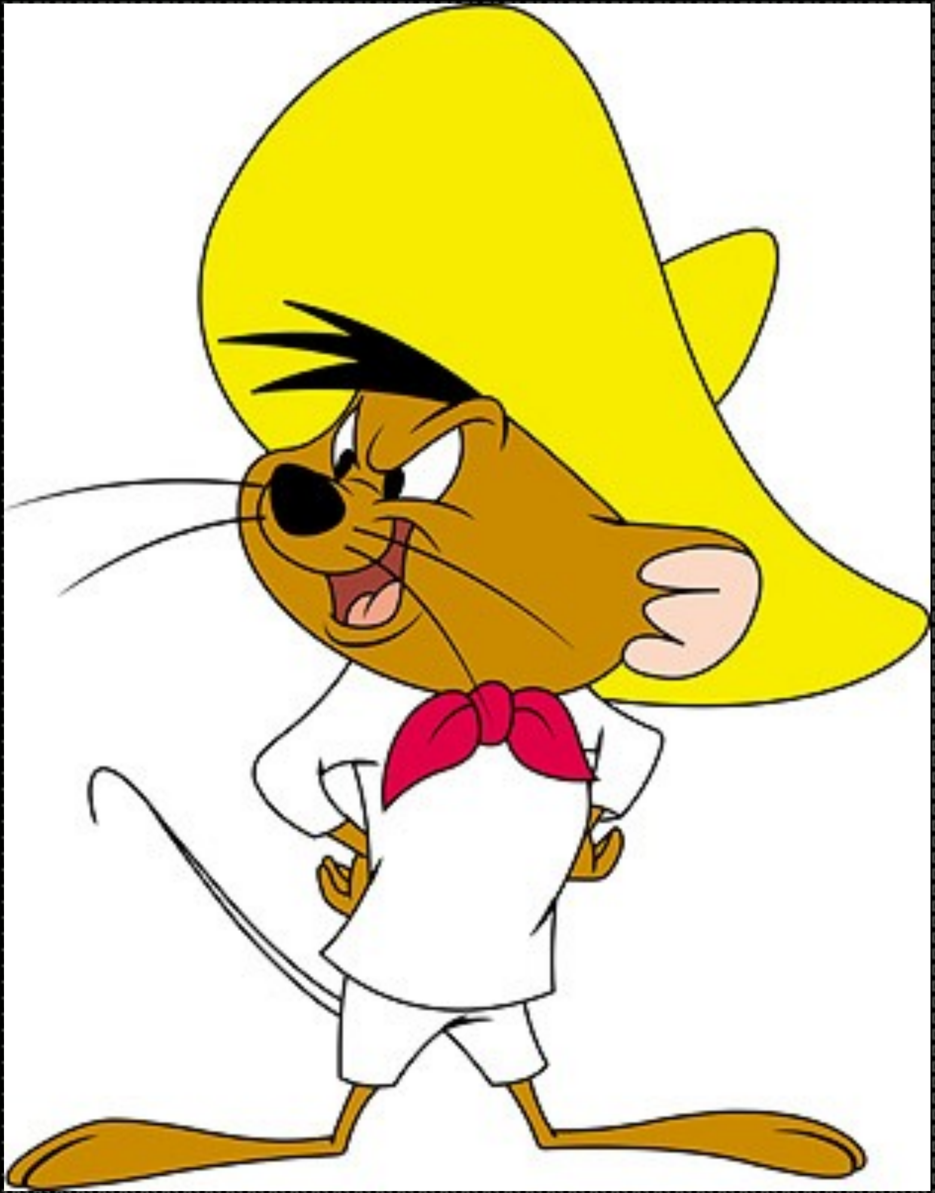
The Tower of Babel case



The Neo vs. Smiths case

The Pavlo case

So Dynamo isn't about speed.



It's about immediate, reliable writes.

It's about operation relaxation.

It's about distribution and fault tolerance.

It's about almost linear scalability.

Time and timestamps

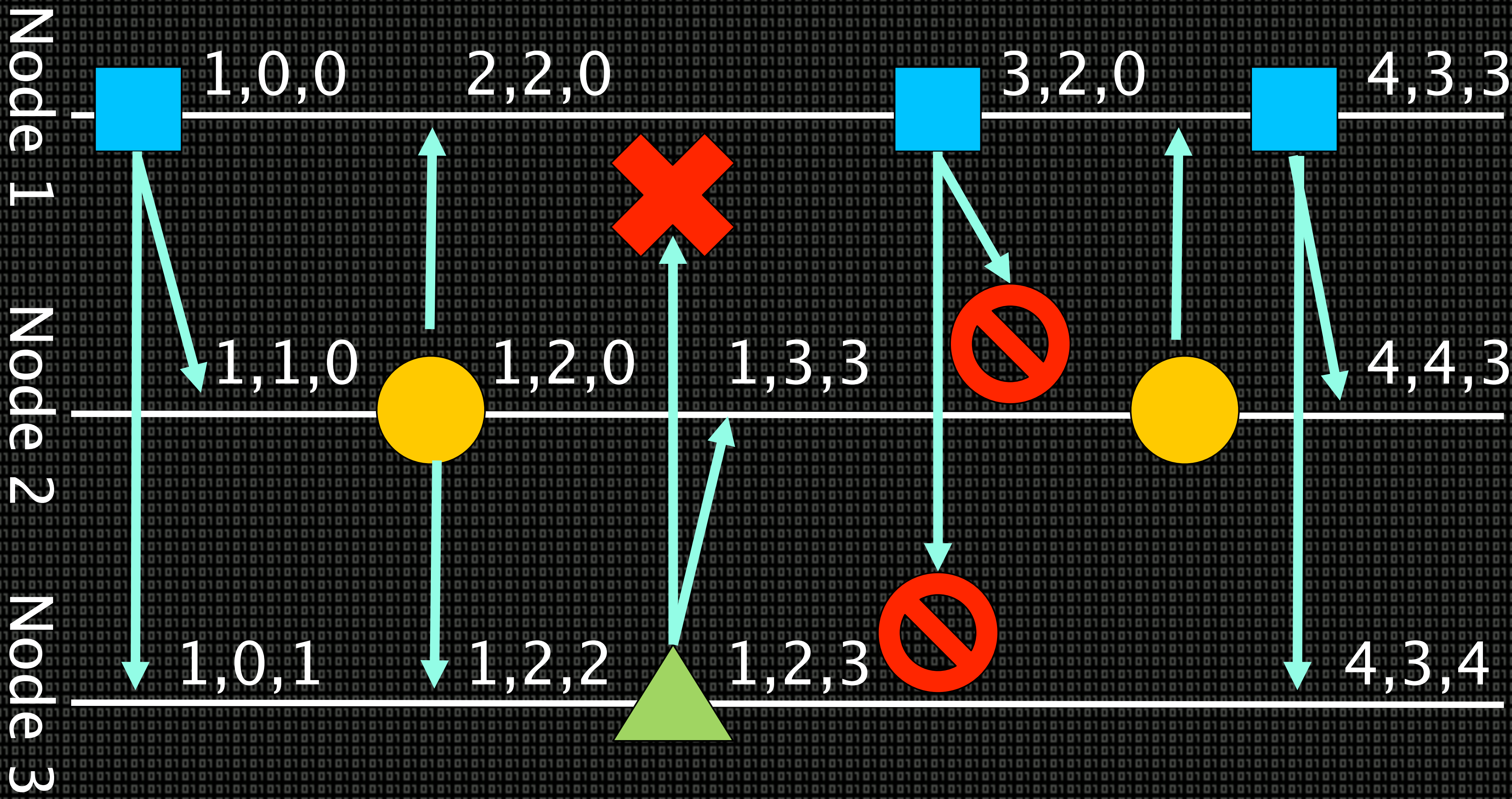
Clocks

V(i), V(j): competing

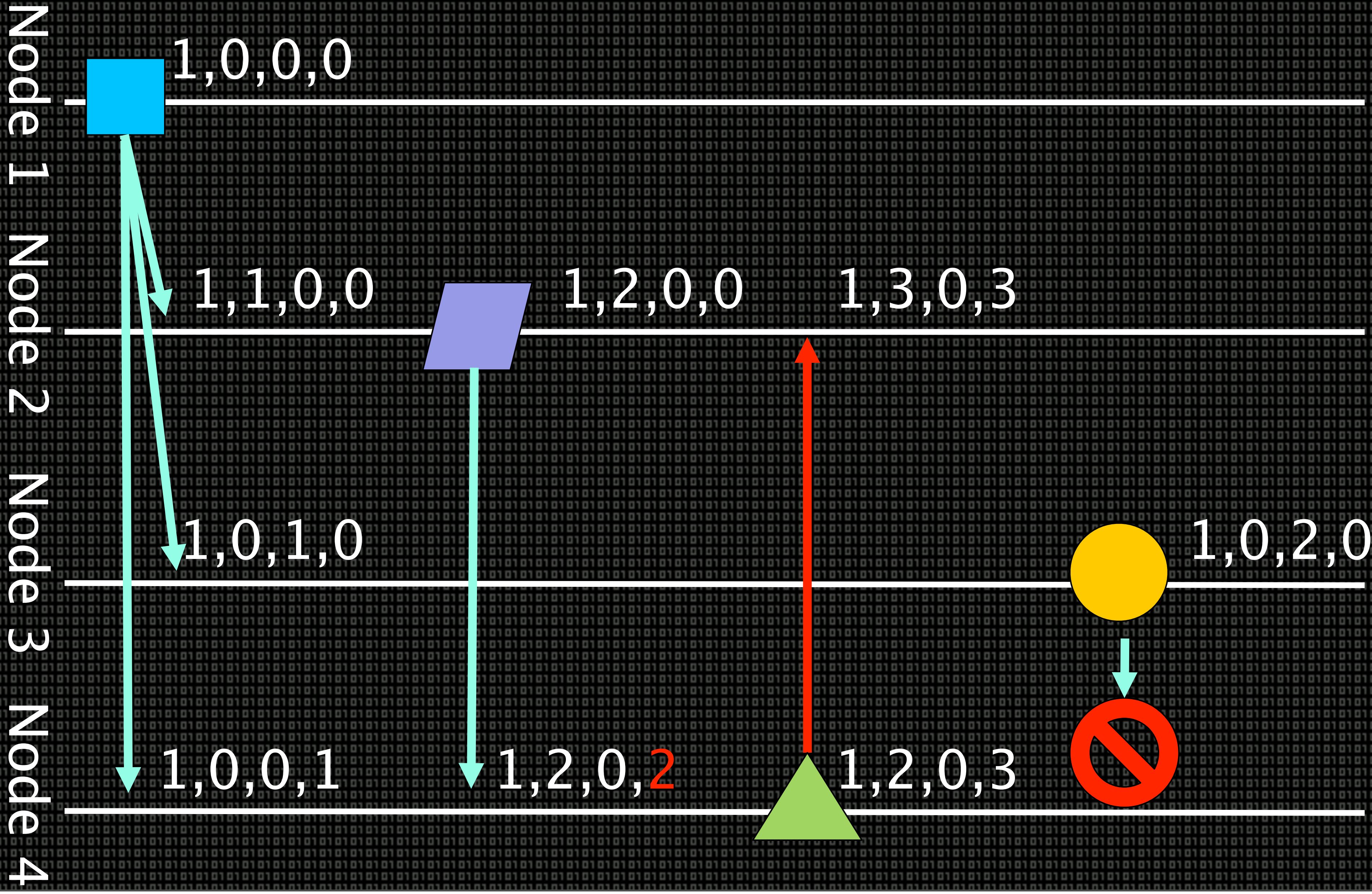
Conflict resolution:

1: siblings, client
2: merge, system
3: voting, system

Vector clocks



Vector clocks



O(1) for data lookups / delta tracking

#

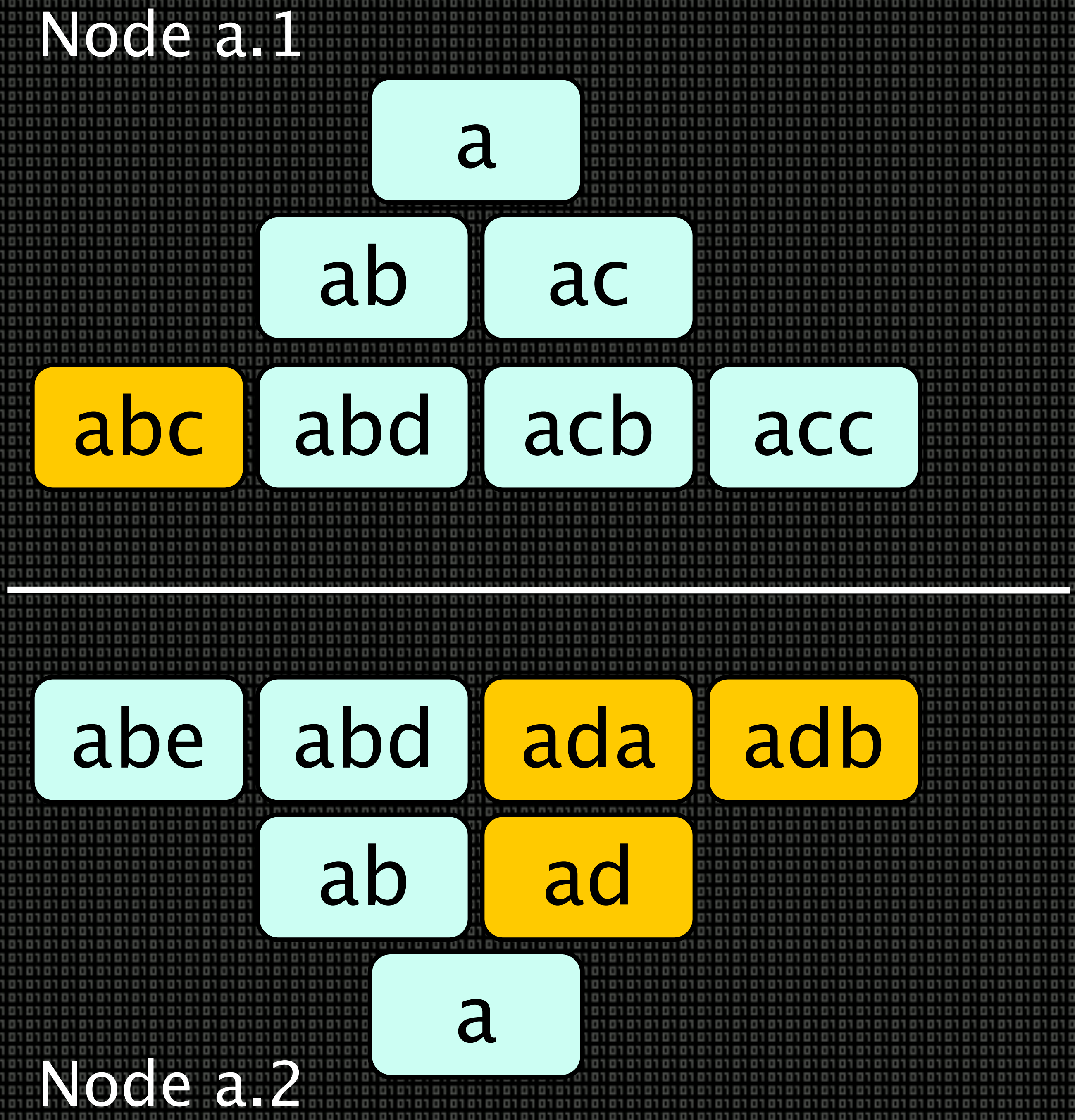
Merkle Trees

N, M: nodes
HT(N), HT(M): hash trees

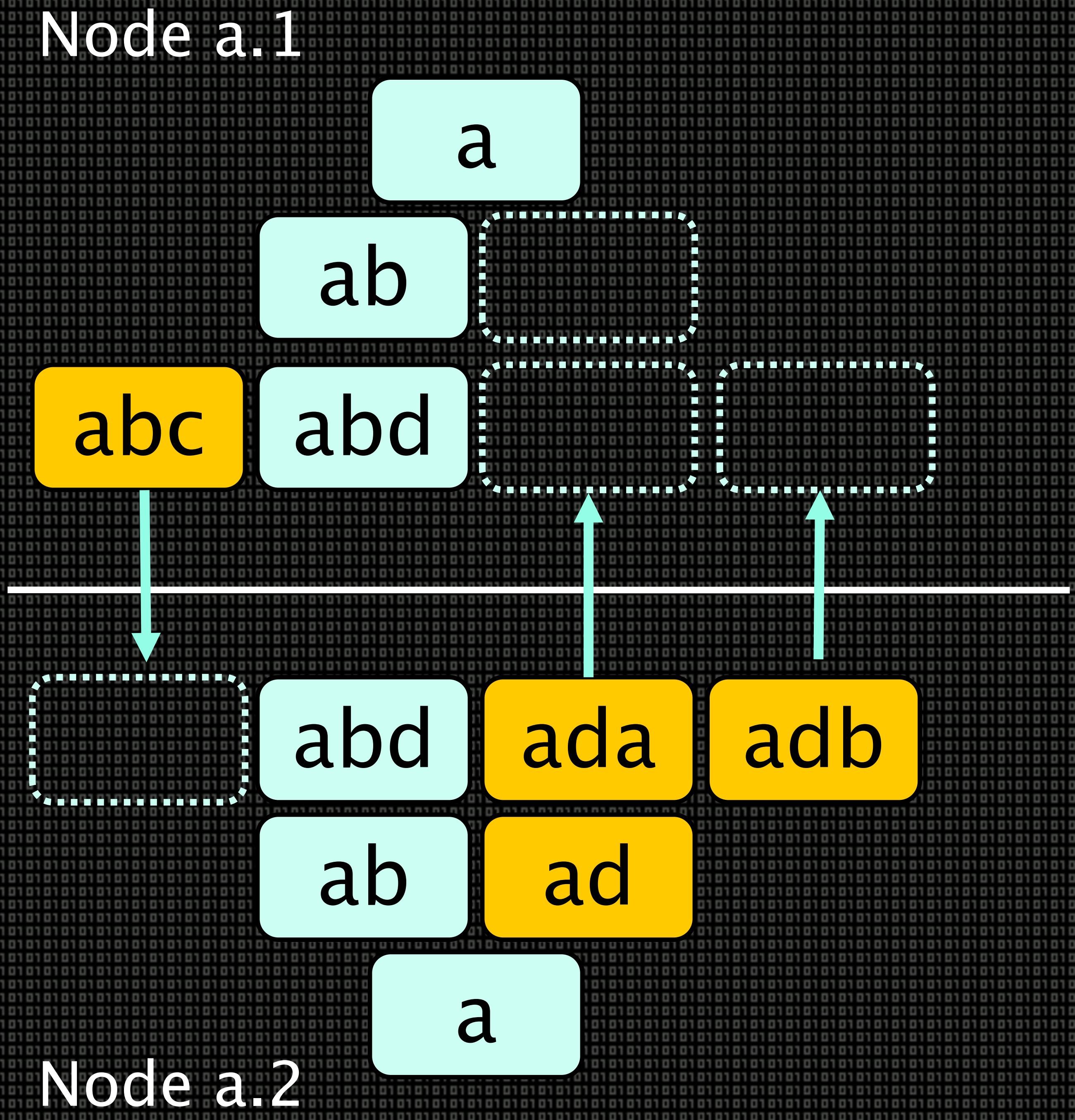
M needs update:

```
    obtain HT(N)
    calc delta(HT(M), HT(N))
    pull keys(delta)
```

Merkle Trees



Merkle Trees



‘Equal” nodes based decentralized distribution

Consensus, agreement, voting, quorum

Consistent hashing – the ring

X bit integer space

$0 <= N <= 2 ^ X$

or: 2 x Pi

$0 <= A <= 2 x Pi$

$x(N) = cos(A)$

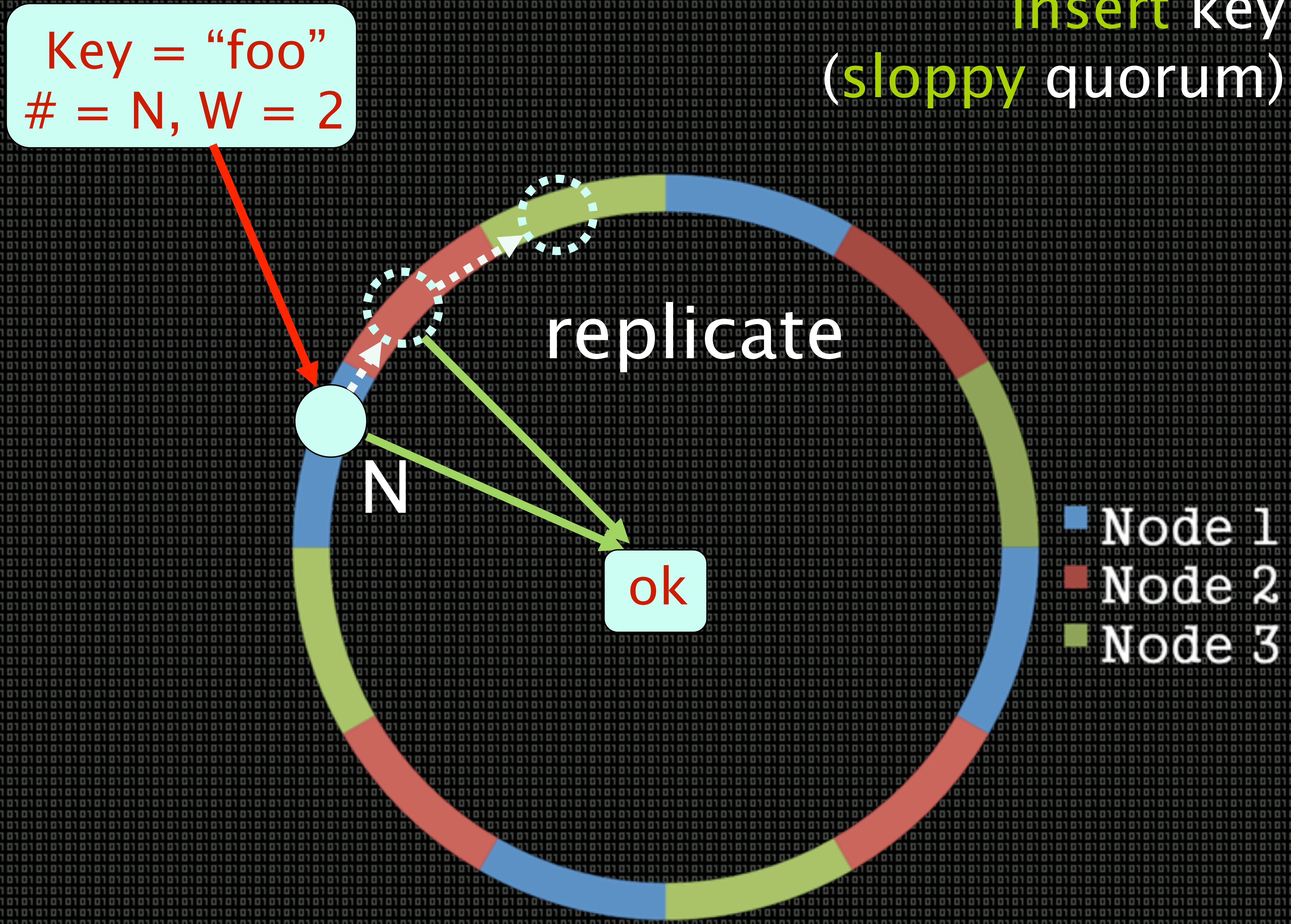
$y(N) = sin(A)$

Quorum

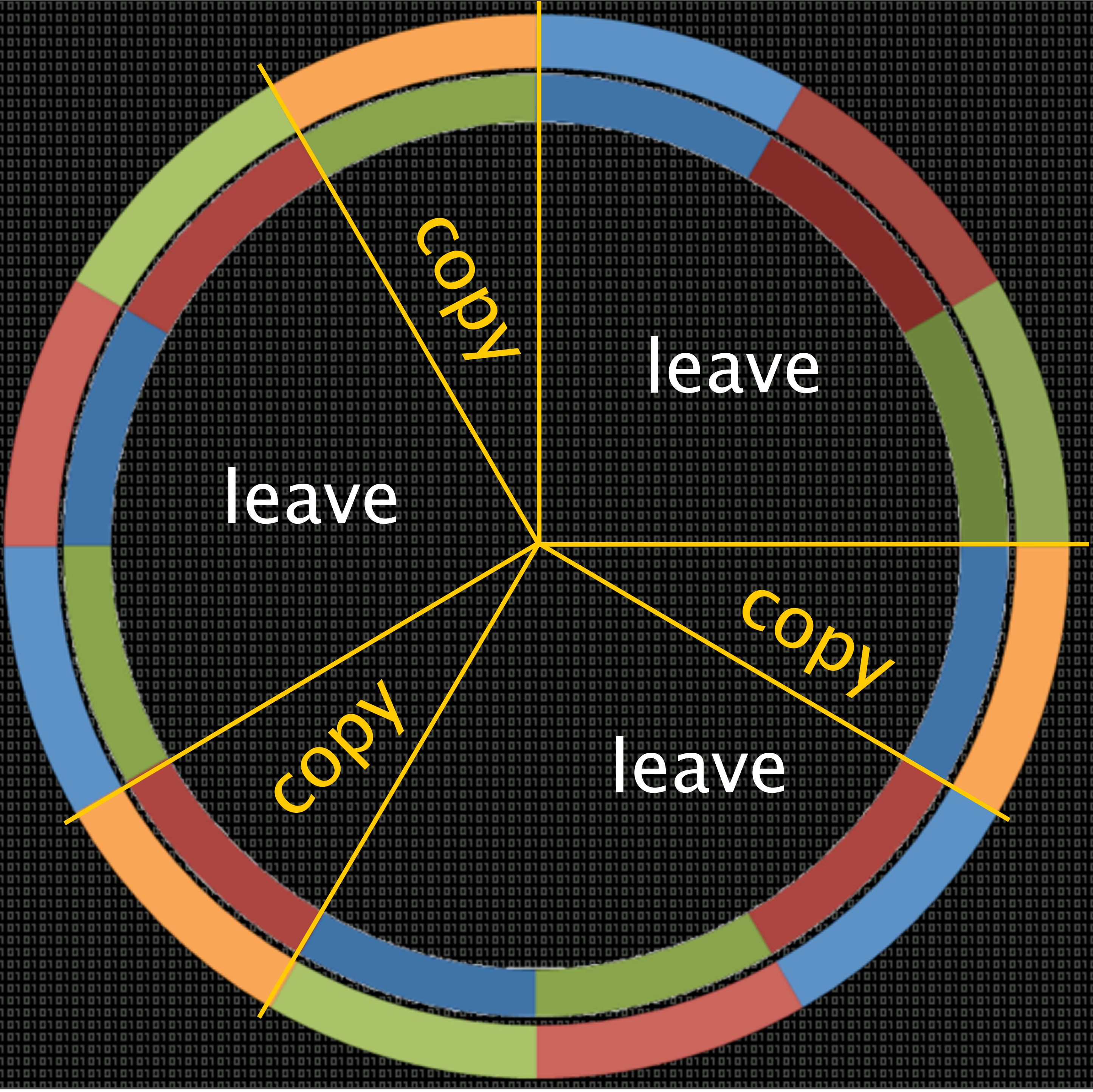
V: vnodes holding a key
W: write quorum
R: read quorum
DW: durable write quorum

$$W > 0.5 * V$$
$$R + W > V$$

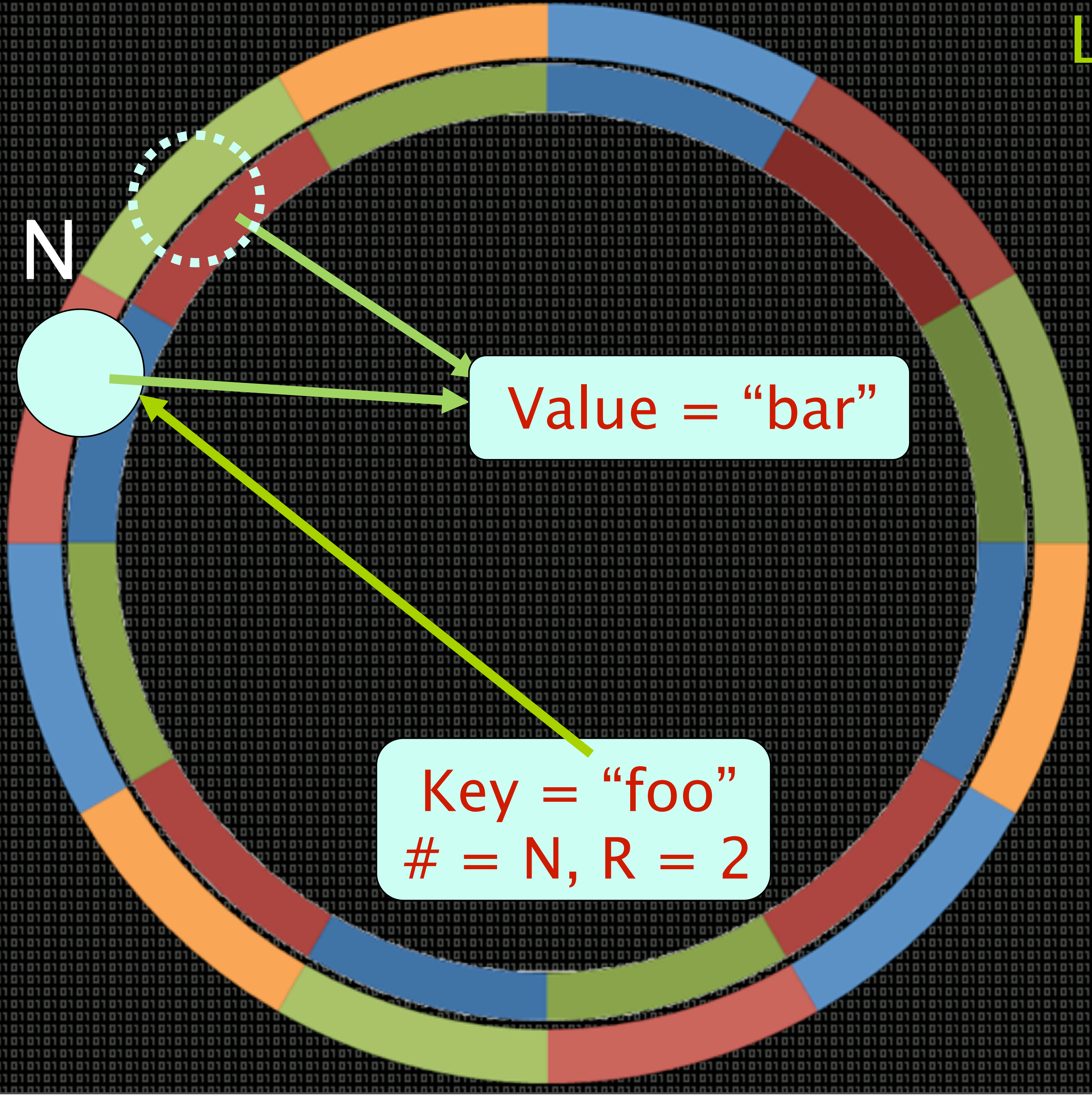
Insert key (sloppy quorum)



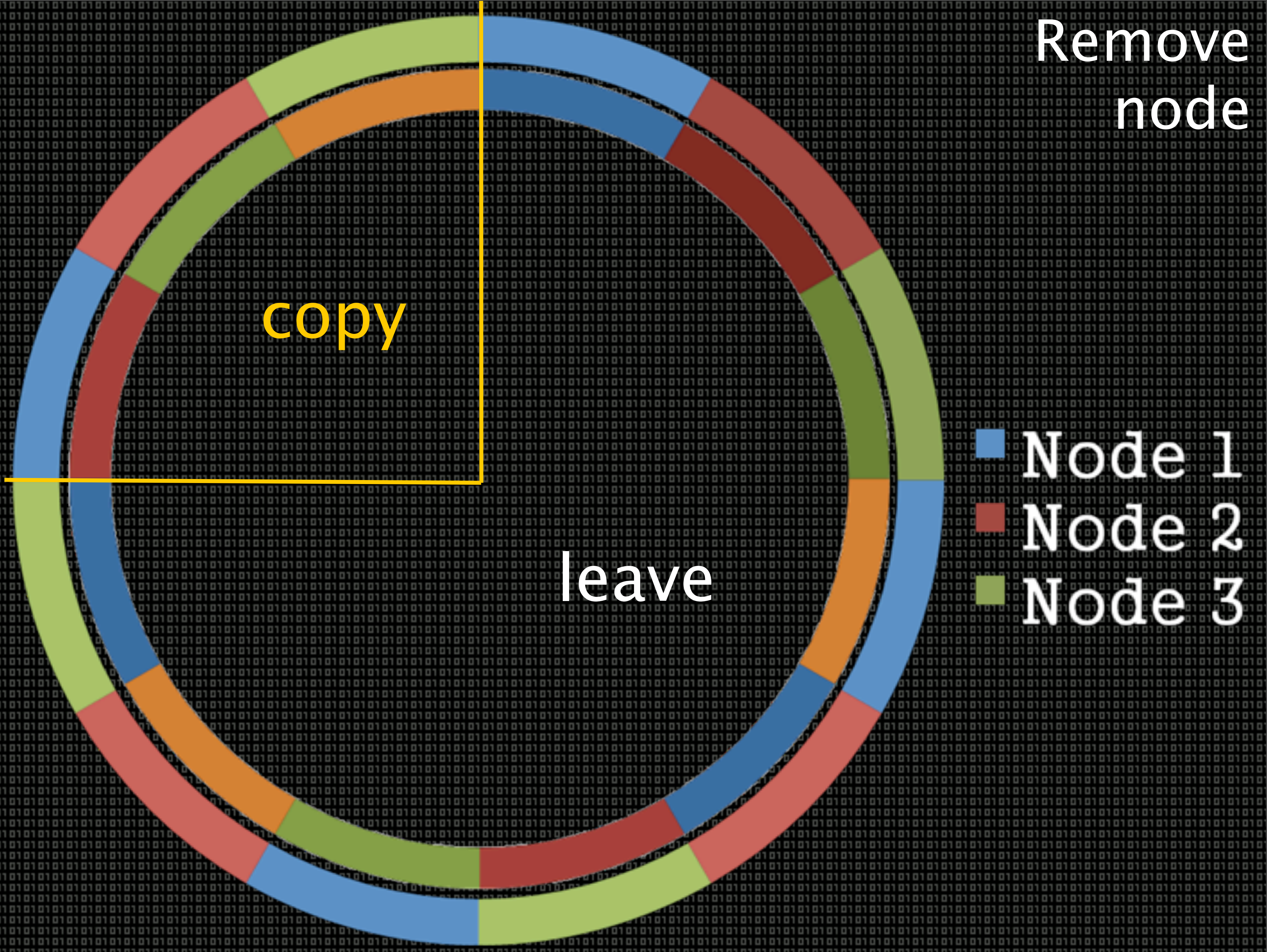
Add node



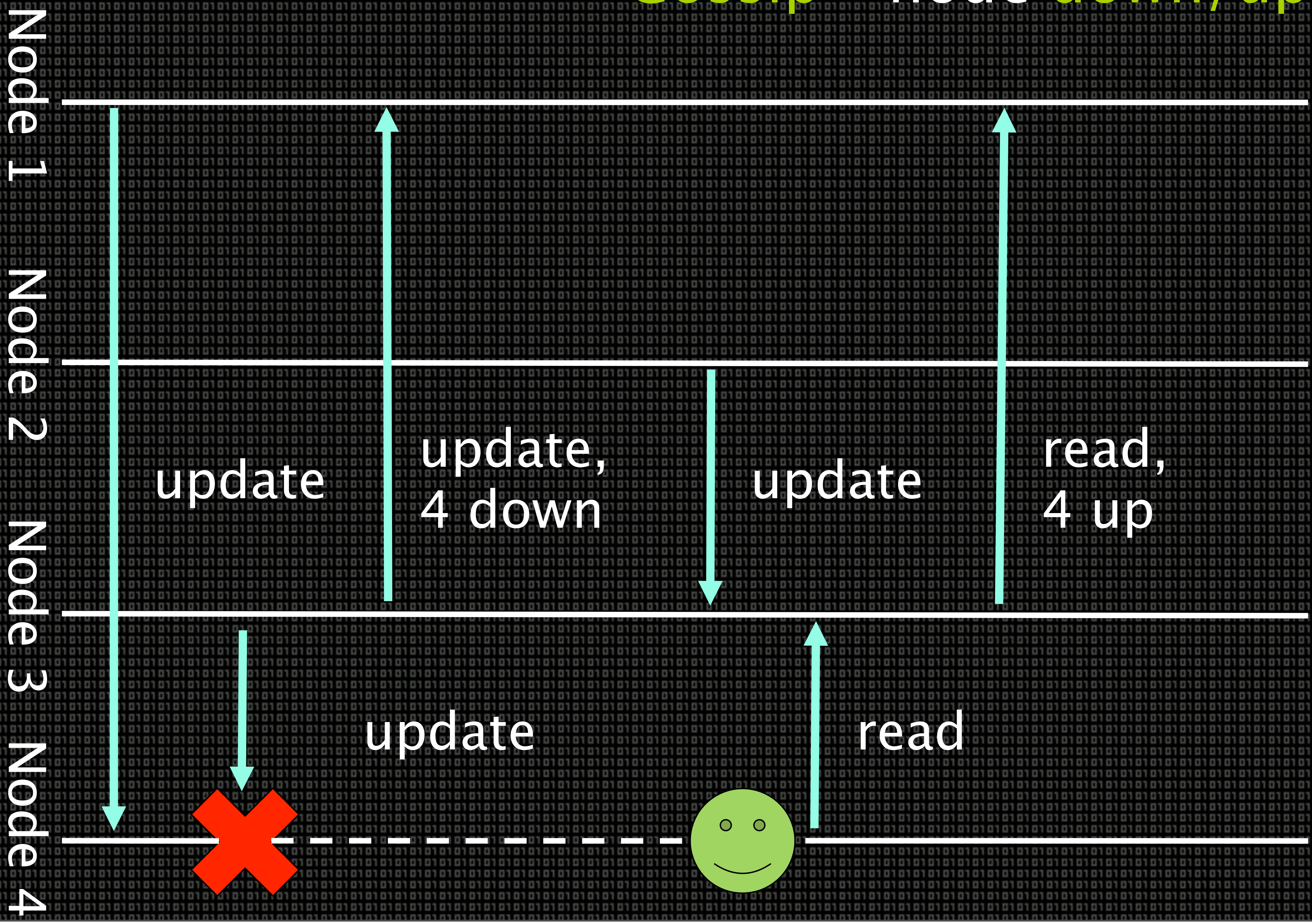
Lookup key (sloppy quorum)



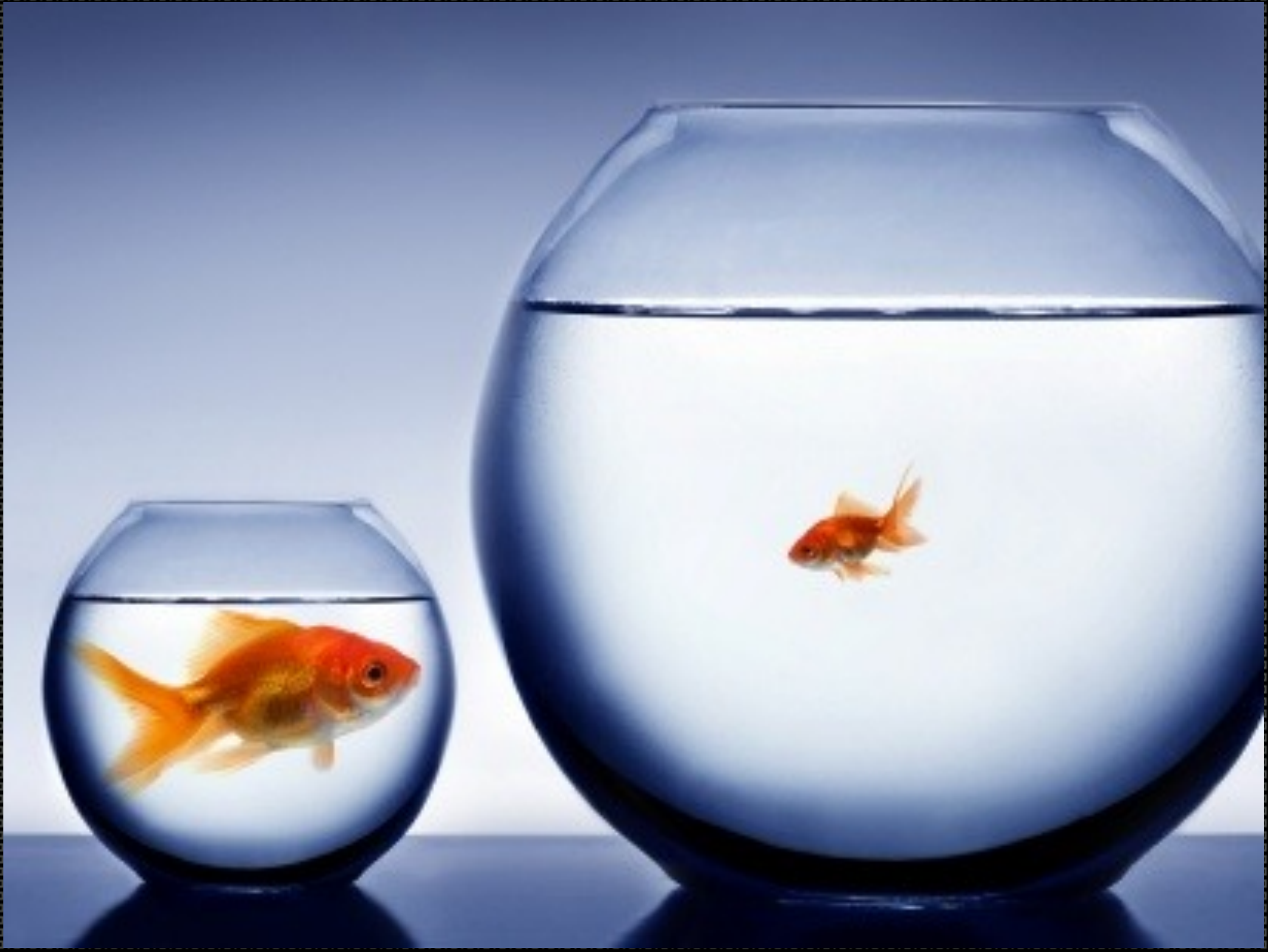
🟦 Node 1
🟥 Node 2
🟩 Node 3
🟧 Node 4


Remove node
copy
leave
Node 1
Node 2
Node 3

Gossip - node down/up



Eventual consistency

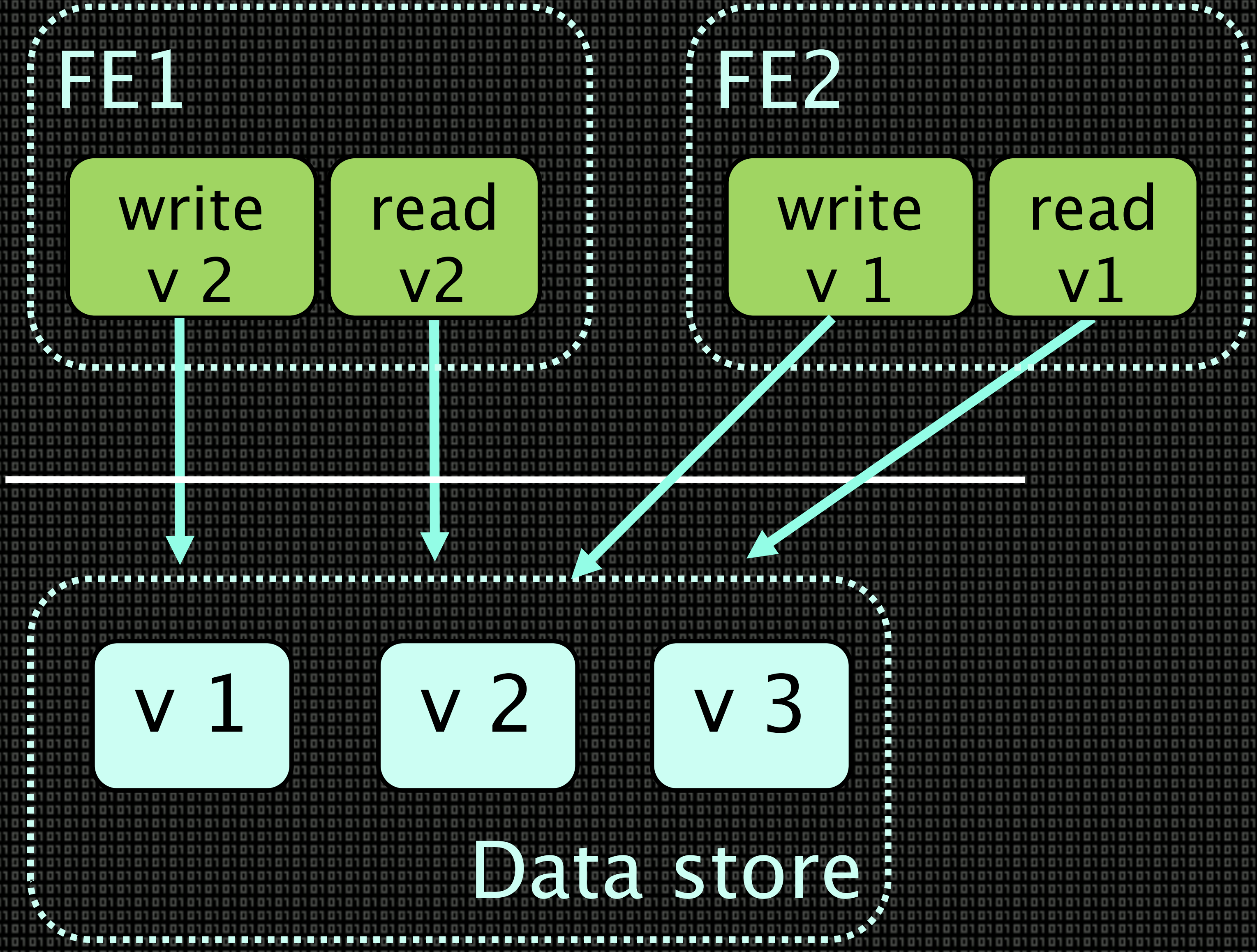


BASE

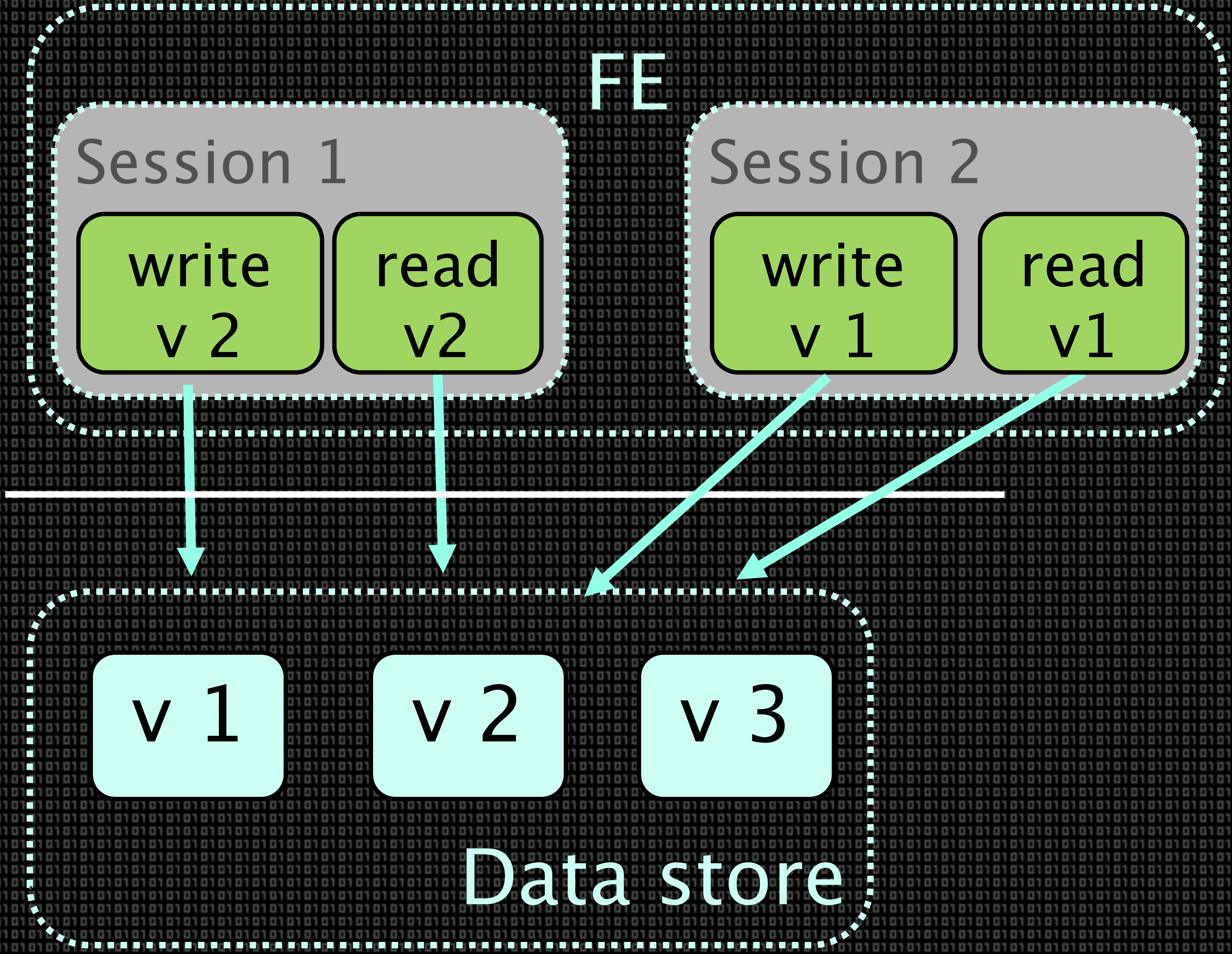
Basically Available,
Soft-state,
Eventually consistent

Opposite to ACID

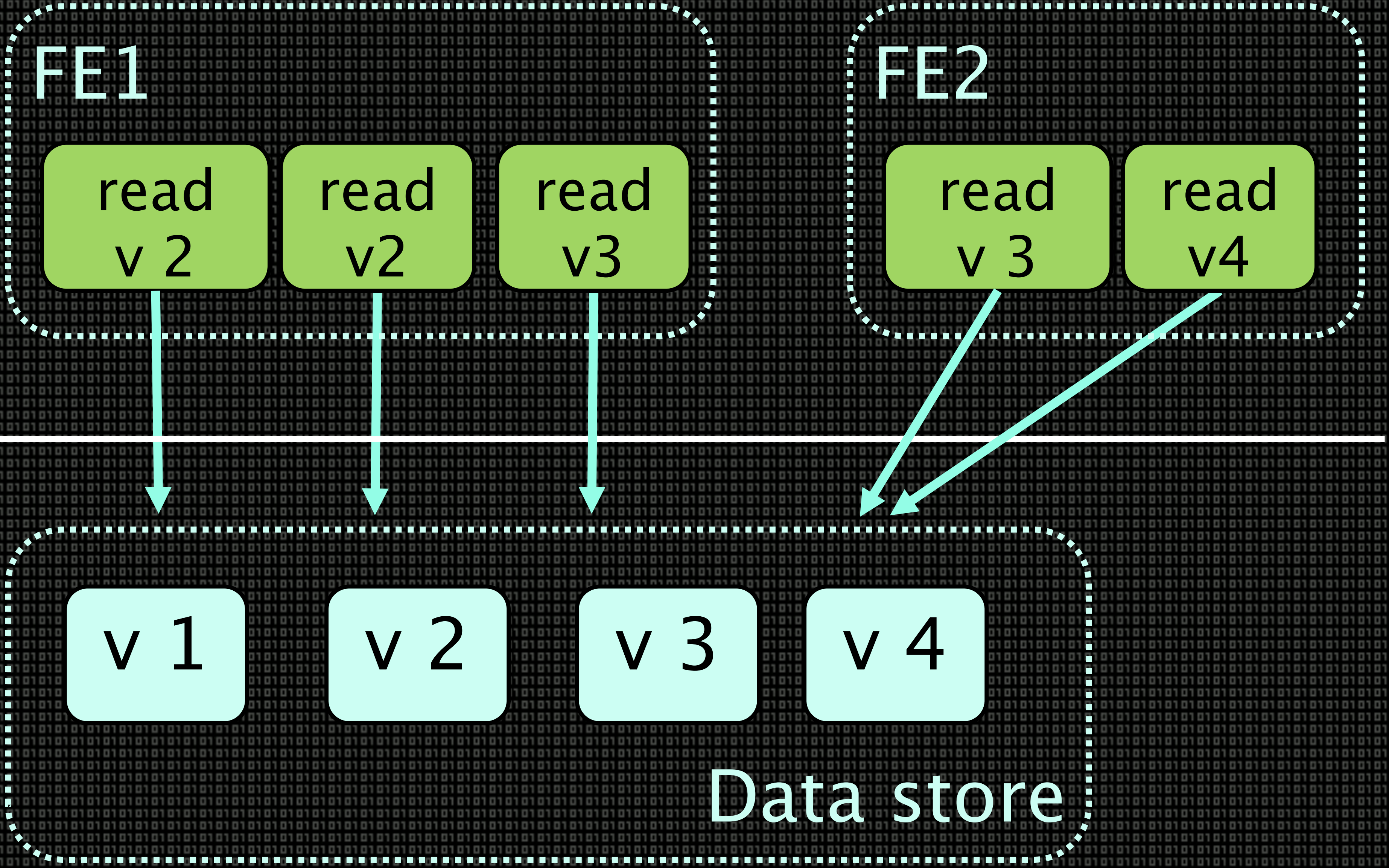
Read your write consistency



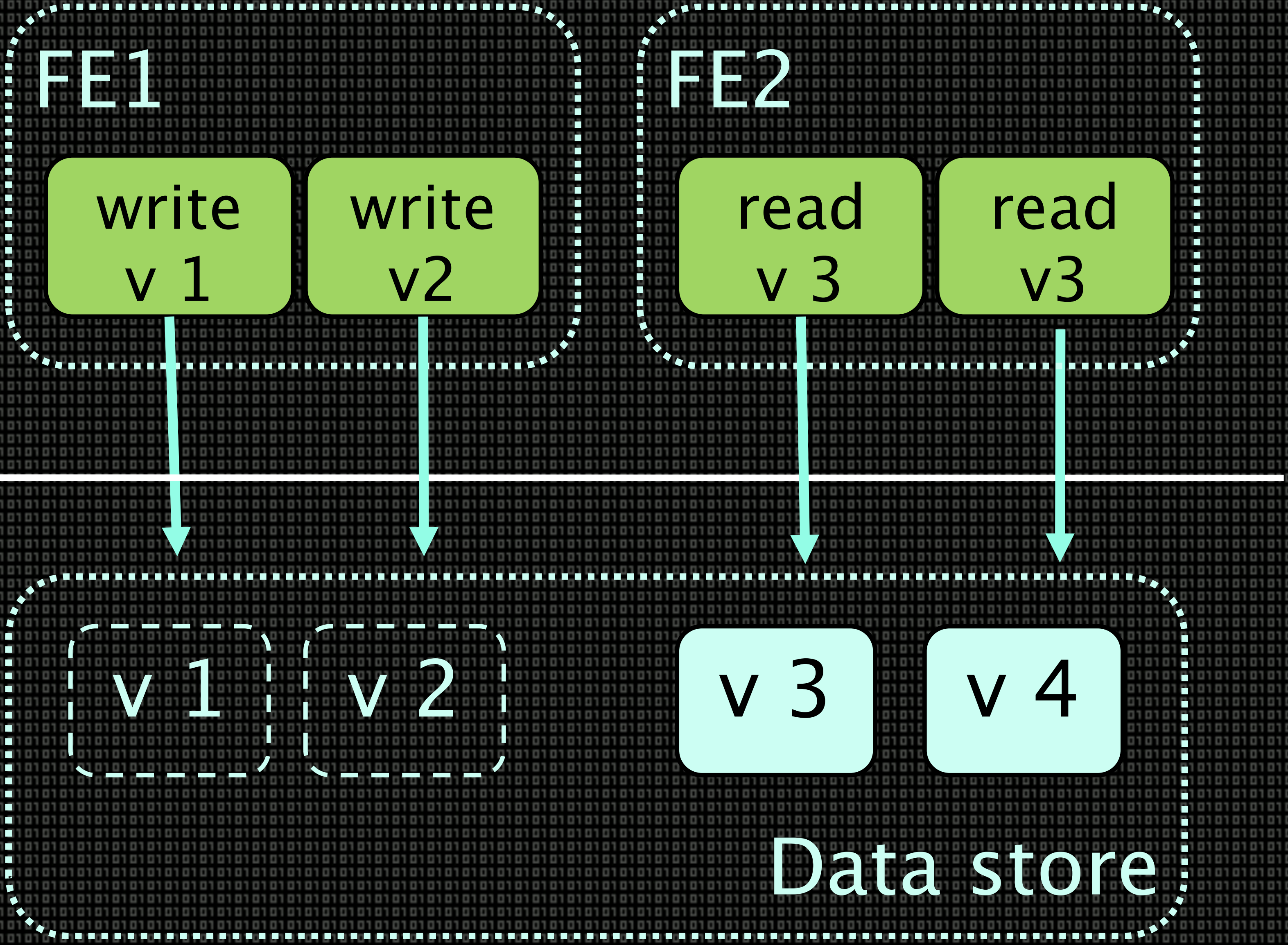
Session consistency



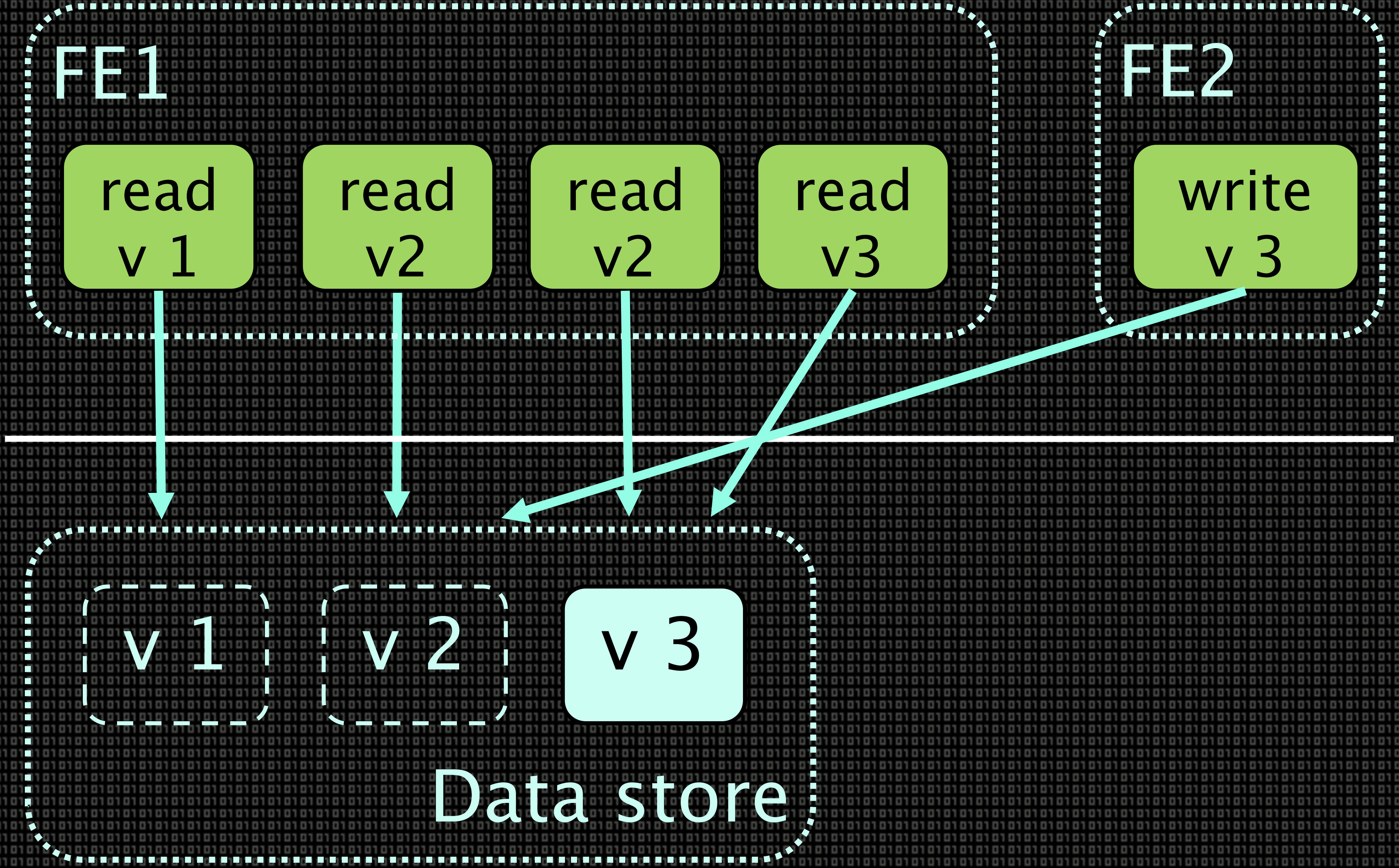
Monotonic read consistency



Monotonic write consistency



Eventual consistency

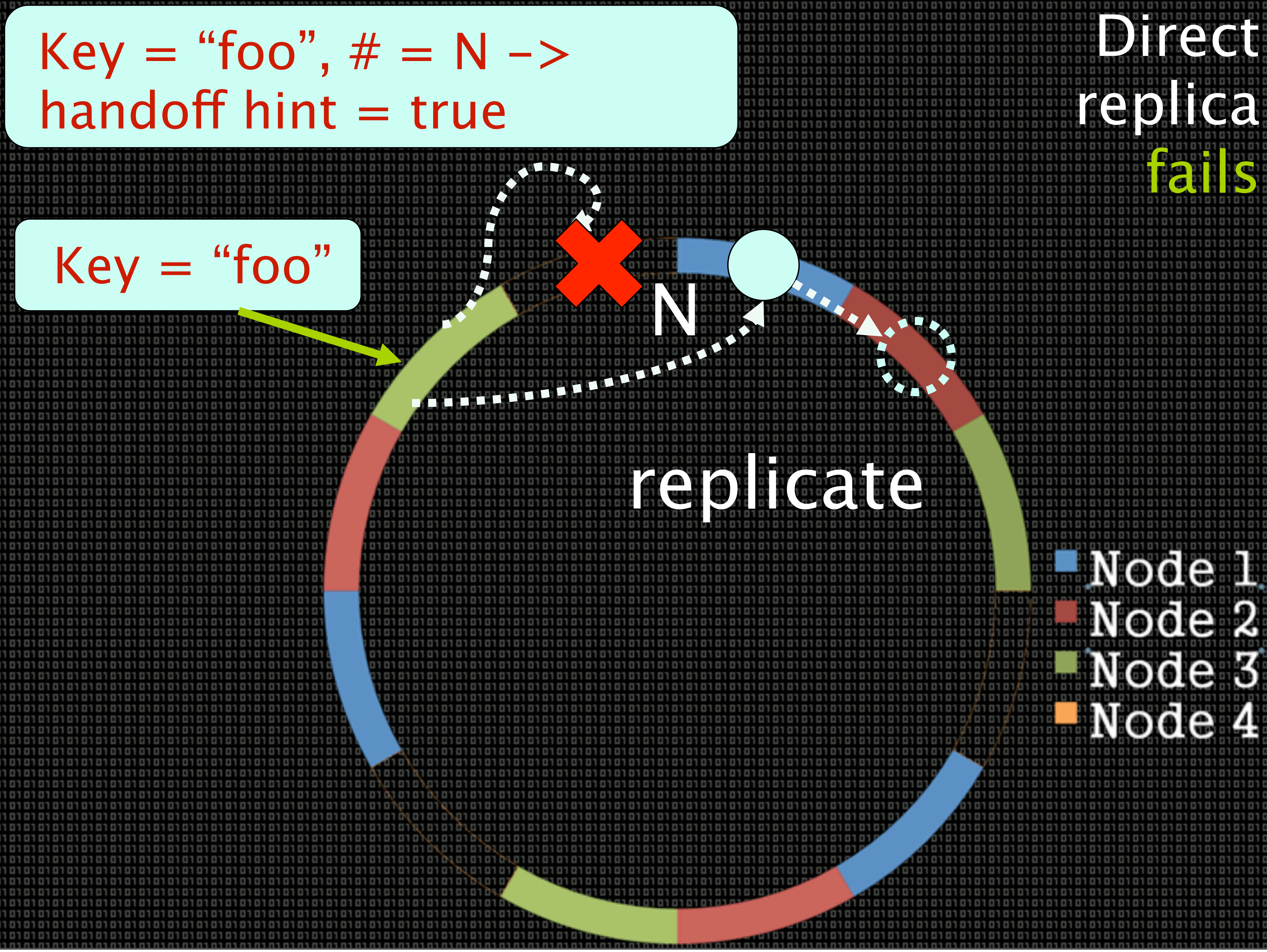


Hinted handoff

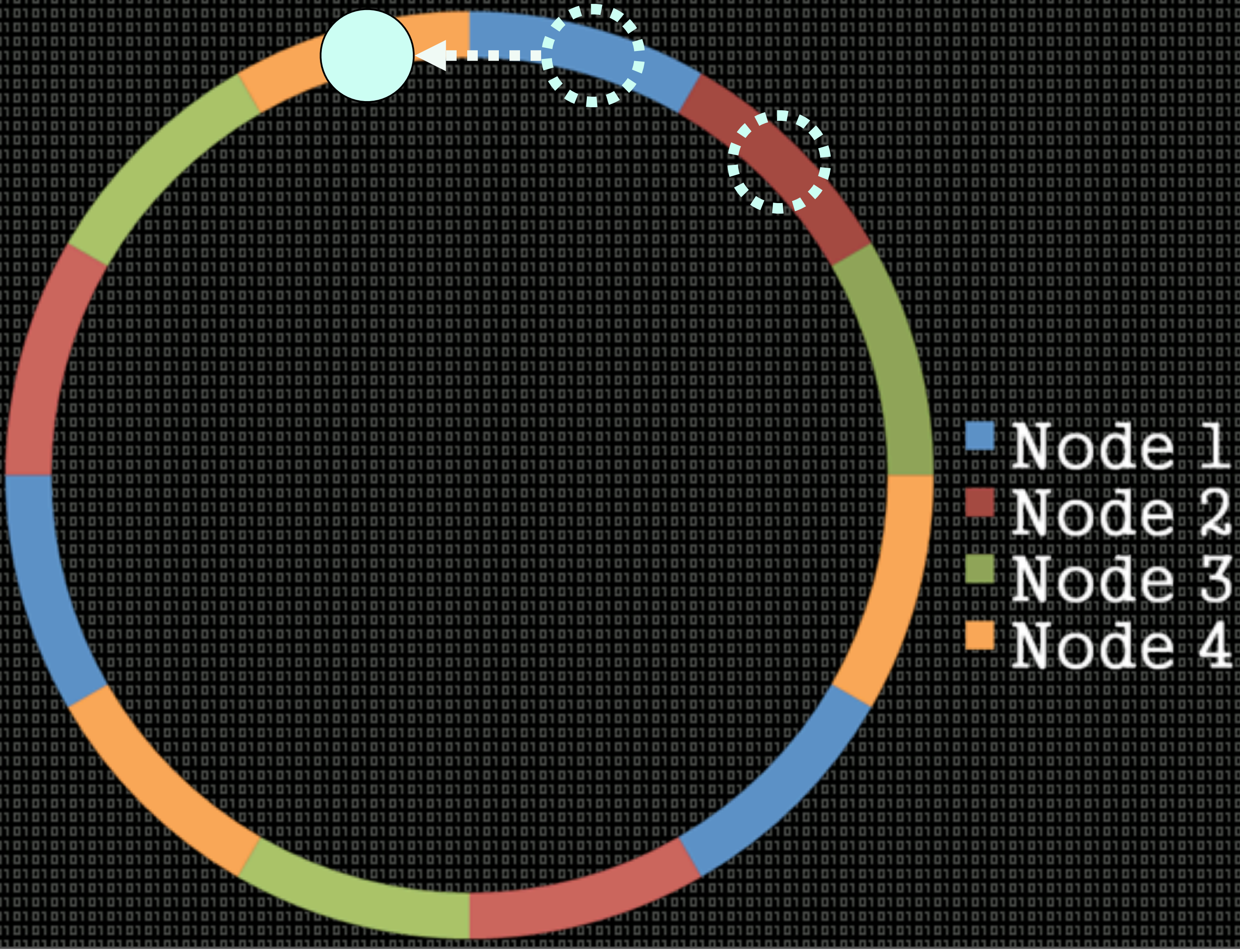
N: node, G: group including N

```
node(N) is unavailable
        replicate to G or
        store data(N) locally
        hint handoff for later
  node(N) is alive
        handoff data to node(N)
```

Direct replica fails

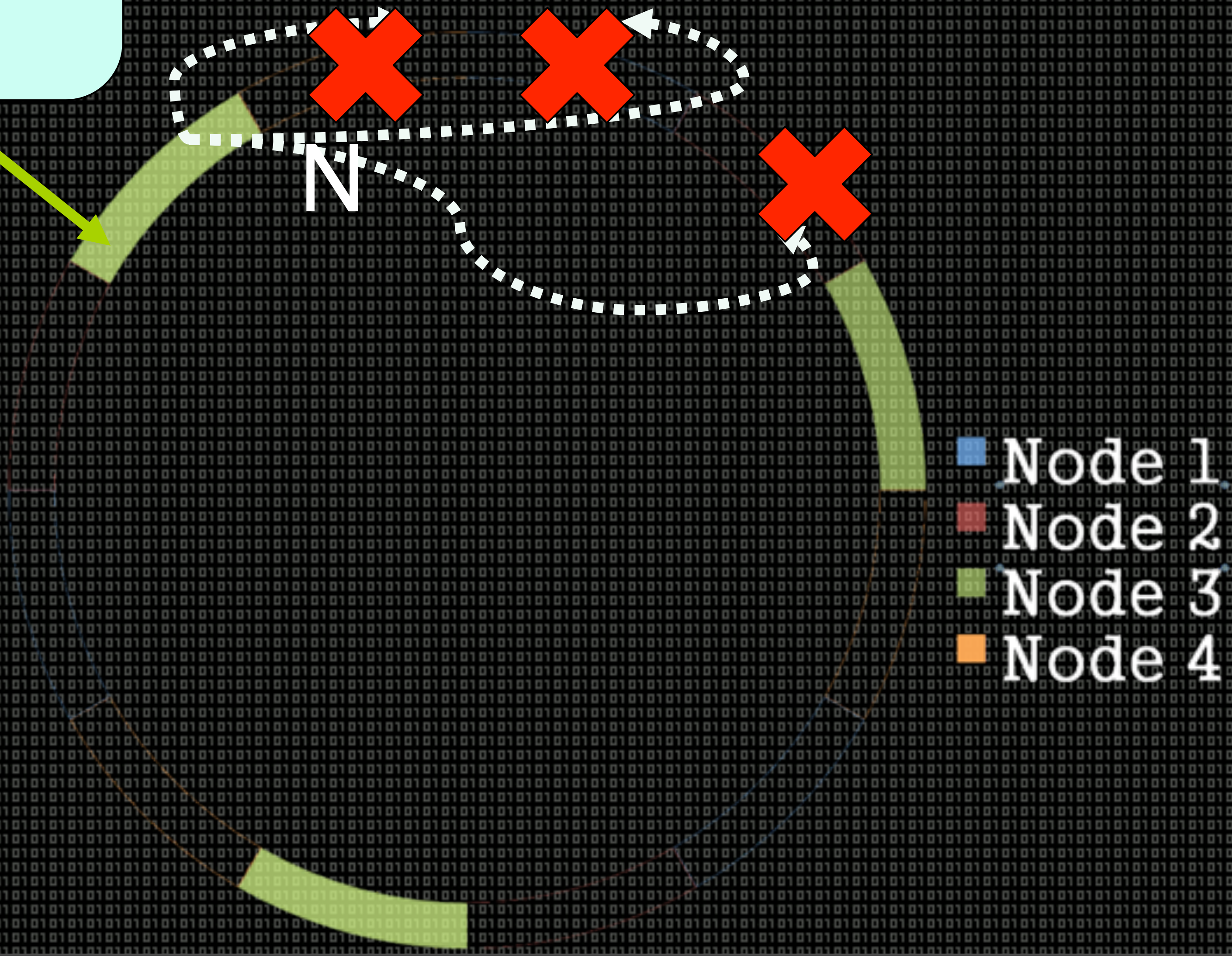


Replica recovers

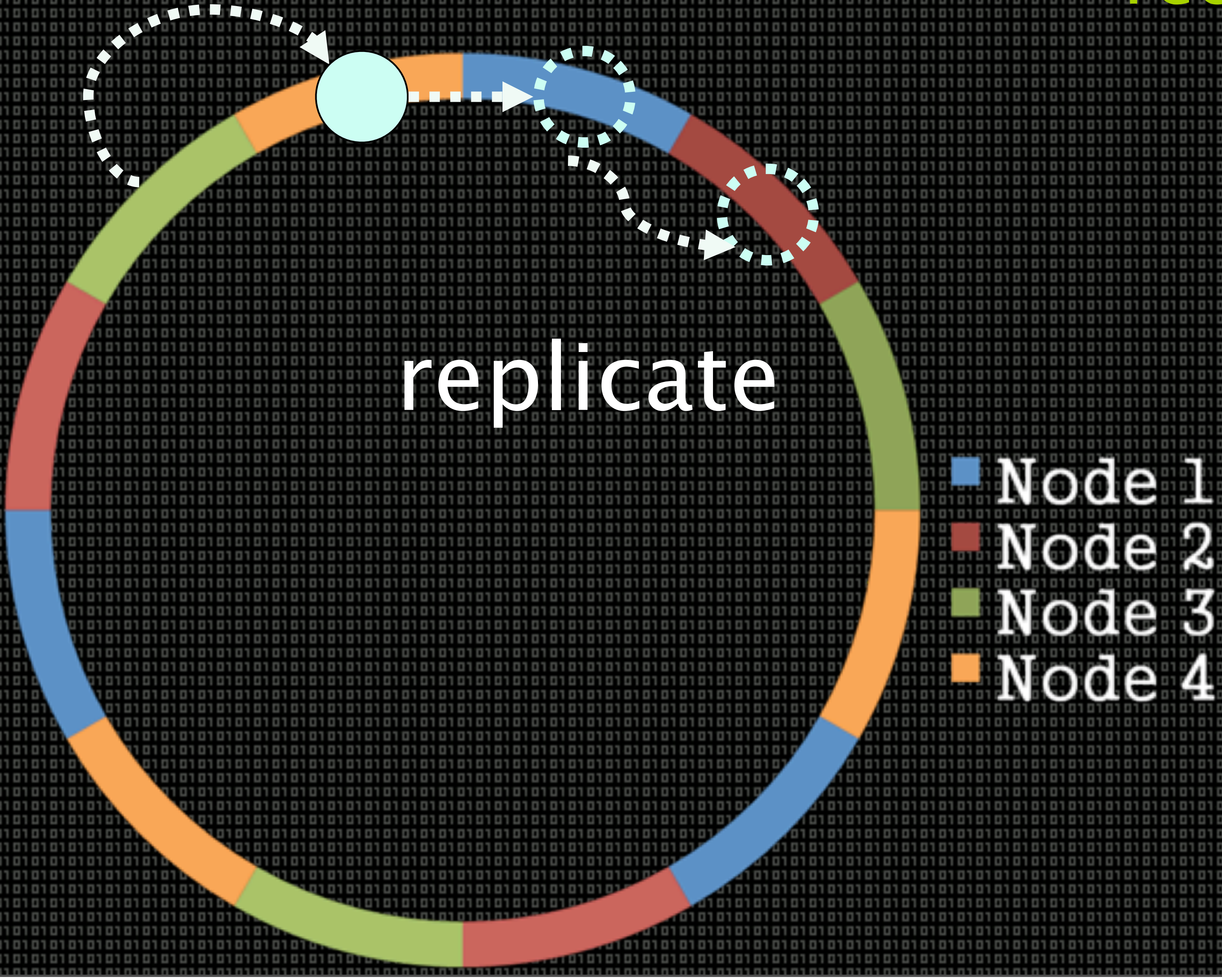


All replicas fail

Key = “foo”, # = N -> handoff hint = true



All replicas recover



Latency is an adjustment screw

Availability is an adjustment screw

CAP – the variations

CA – irrelevant

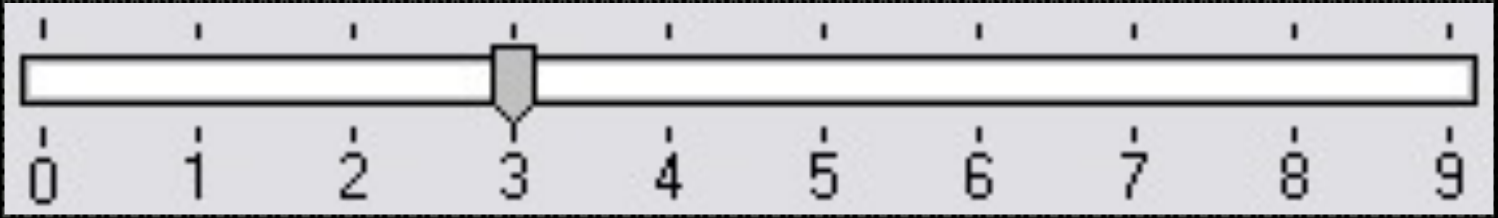
CP – eventually unavailable offering maximum consistency

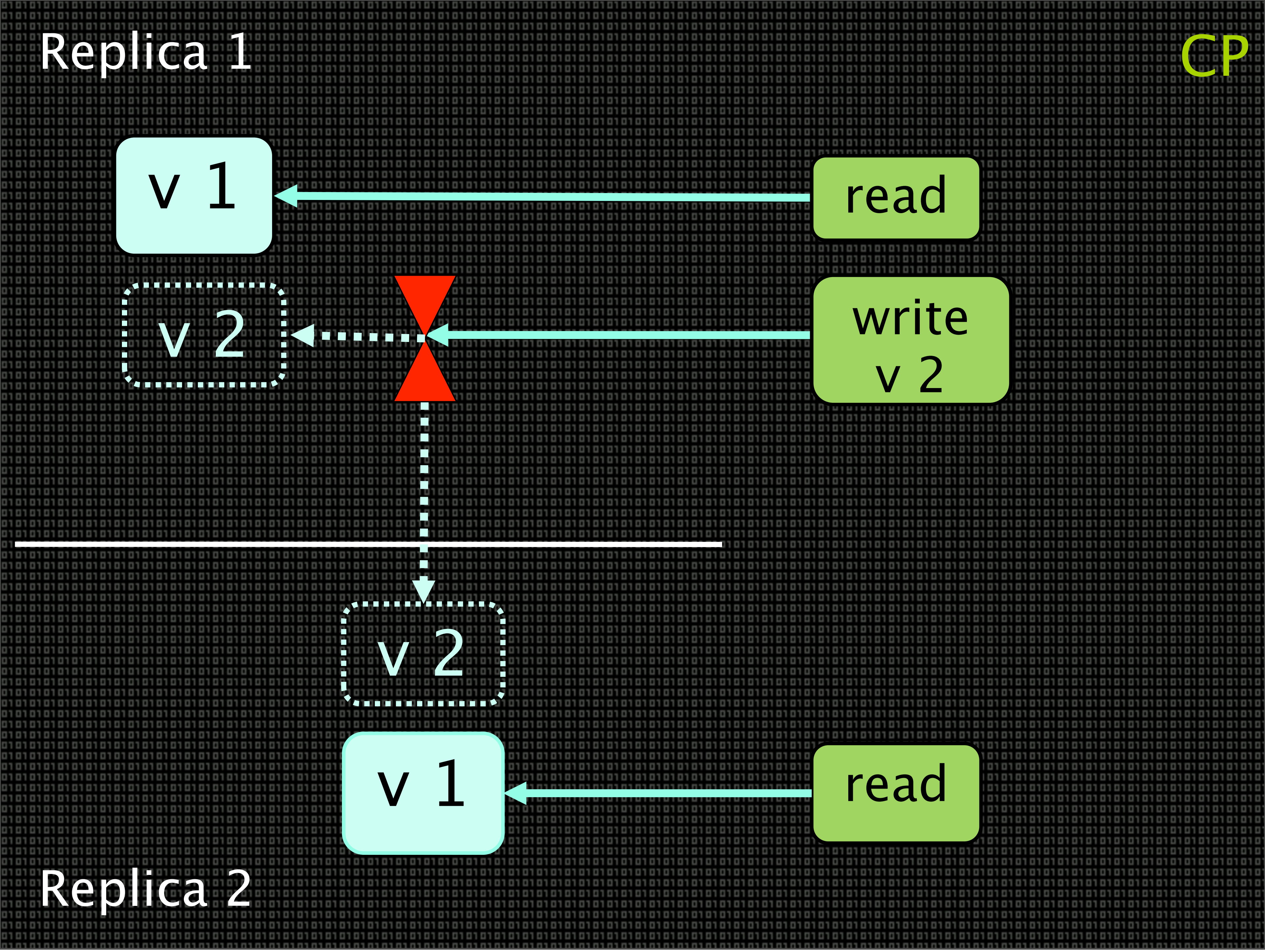
AP – eventually inconsistent offering maximum availability

CAP – the tradeoff

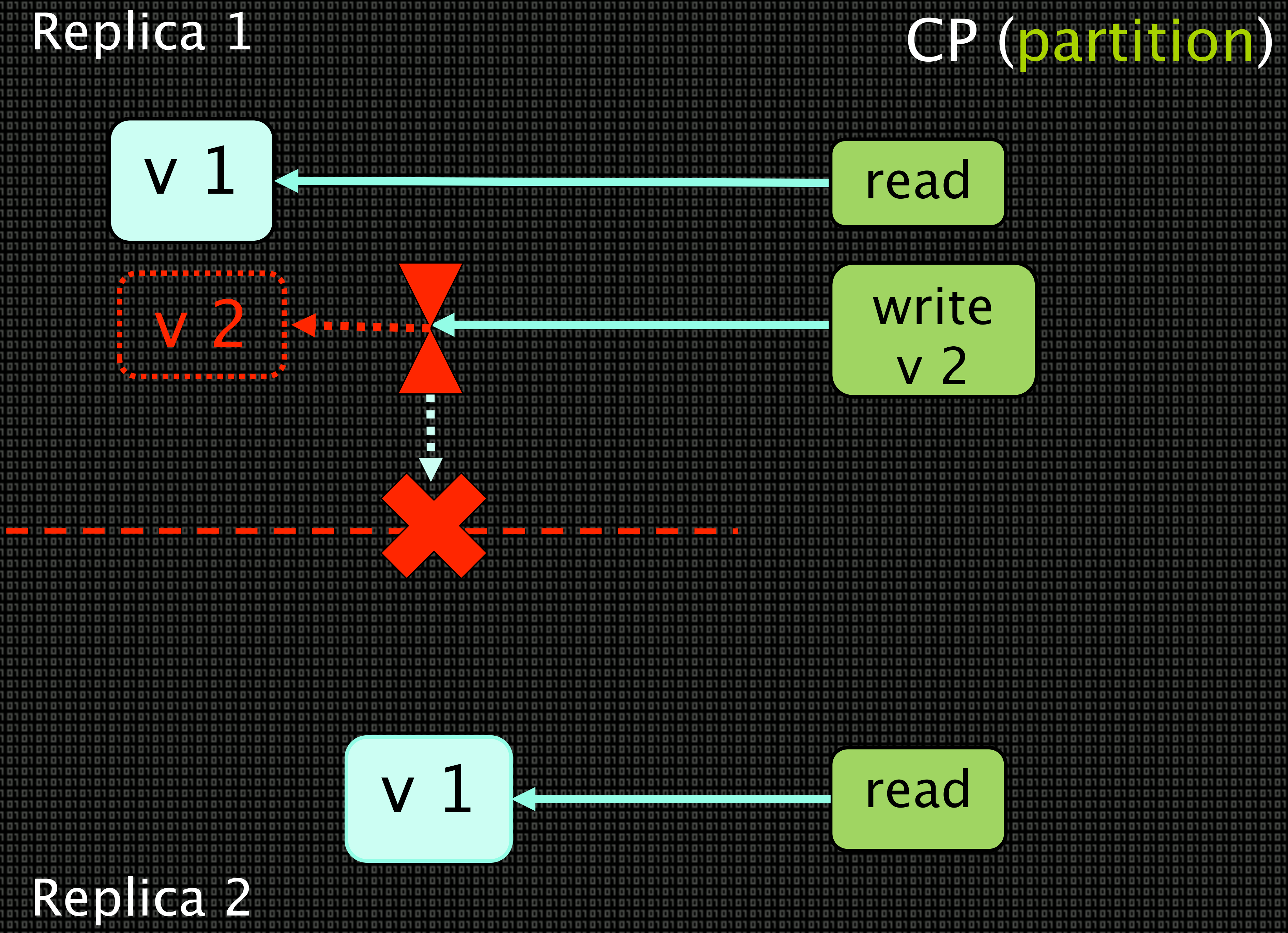
A

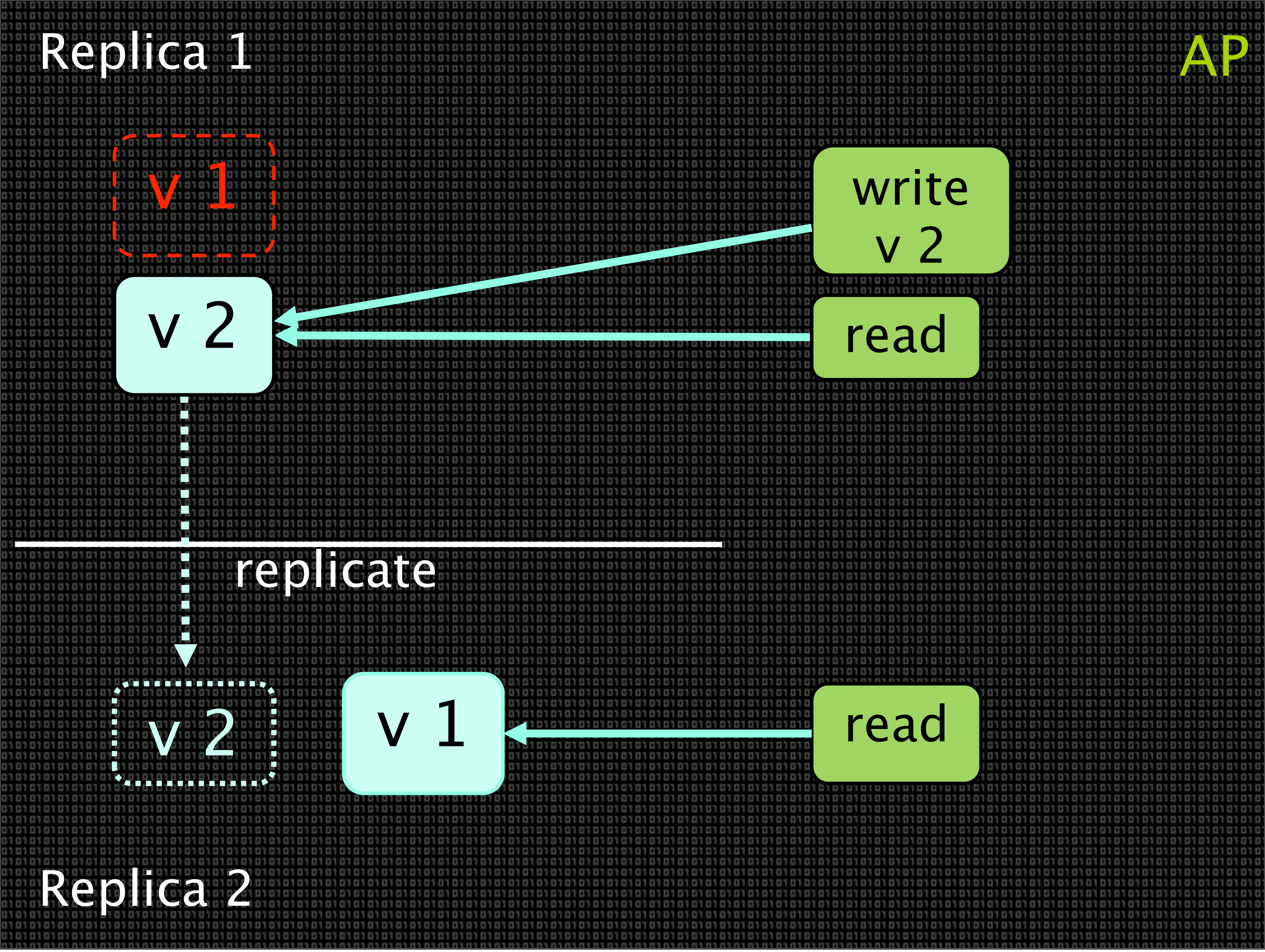
C



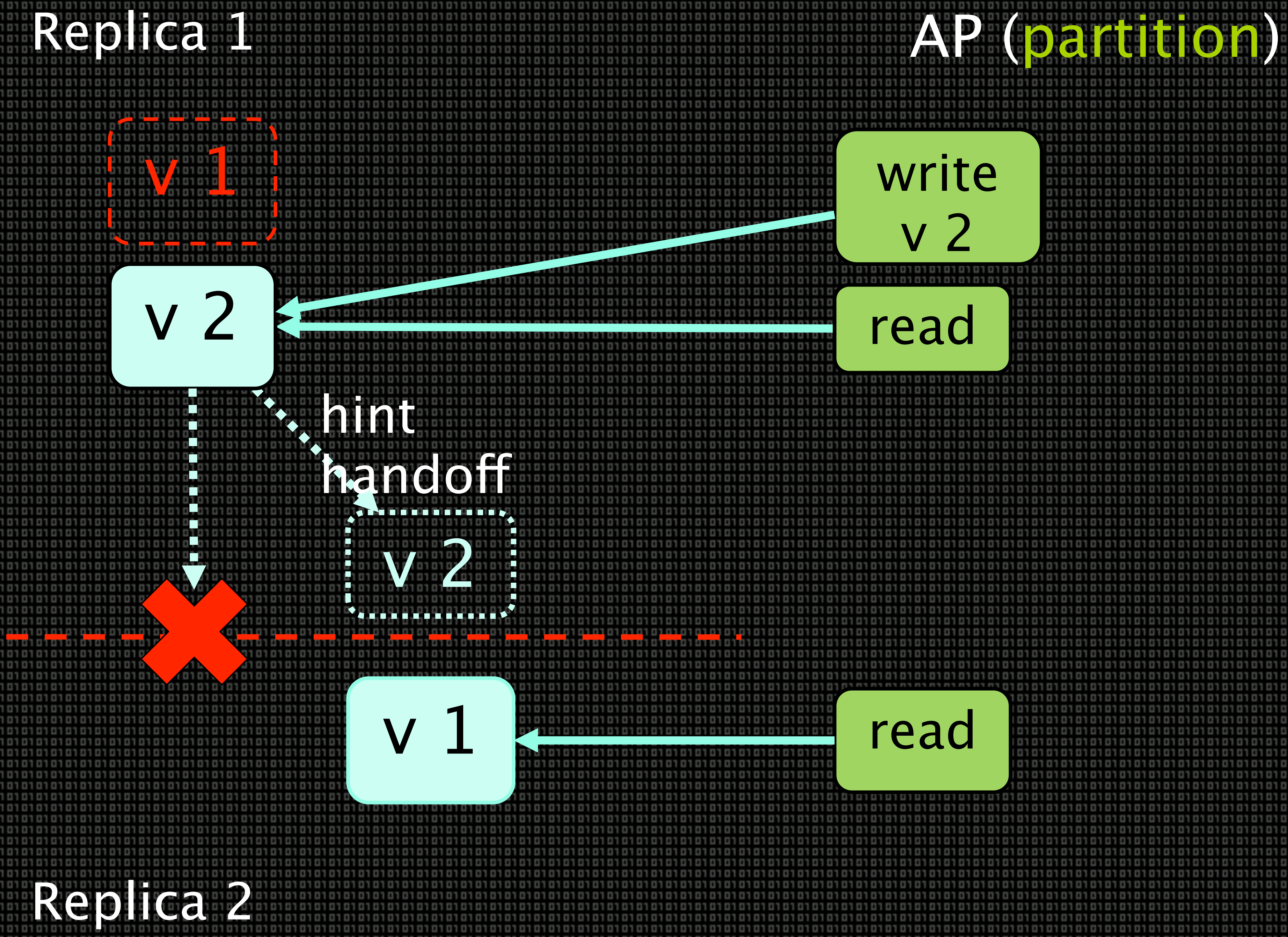

Replica 1
CP
v 1
read
v 2
write v 2
v 2
v 1
read
Replica 2

CP (partition)

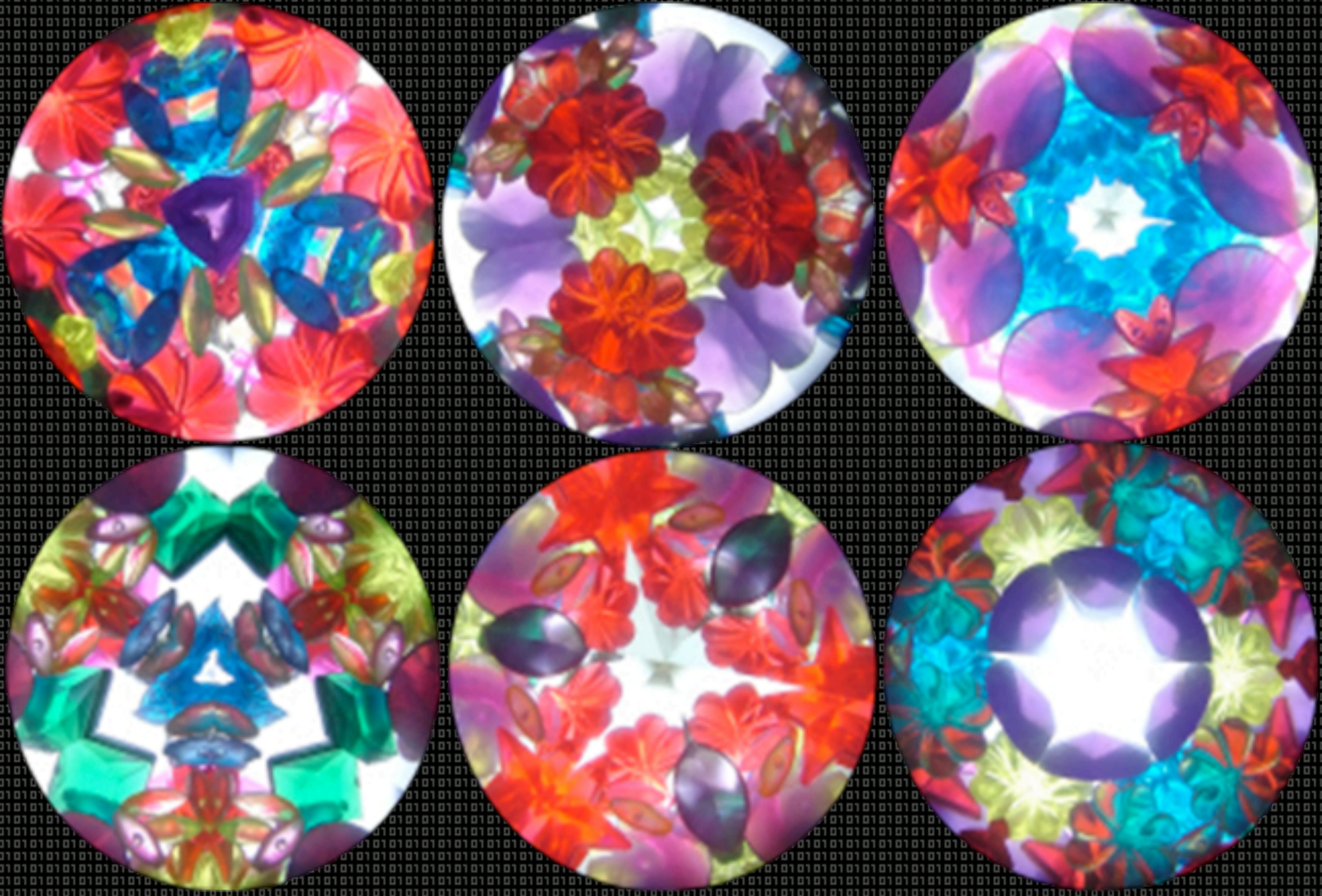



Replica 1
AP
v 1
write v 2
v 2
read
replicate
v 2
v 1
read
Replica 2

AP (partition)



Frequent structure changes



Thank you

Many graphics I’ve
created myself

Some images originate from
istockphoto.com

except few ones taken
from Wikipedia
and product pages