Decentralizing Distributed Consensus with Faster Paxos

[Distributed consensus revised]

Heidi Howard @ Cambridge University
heidi.howard@cl.cam.ac.uk

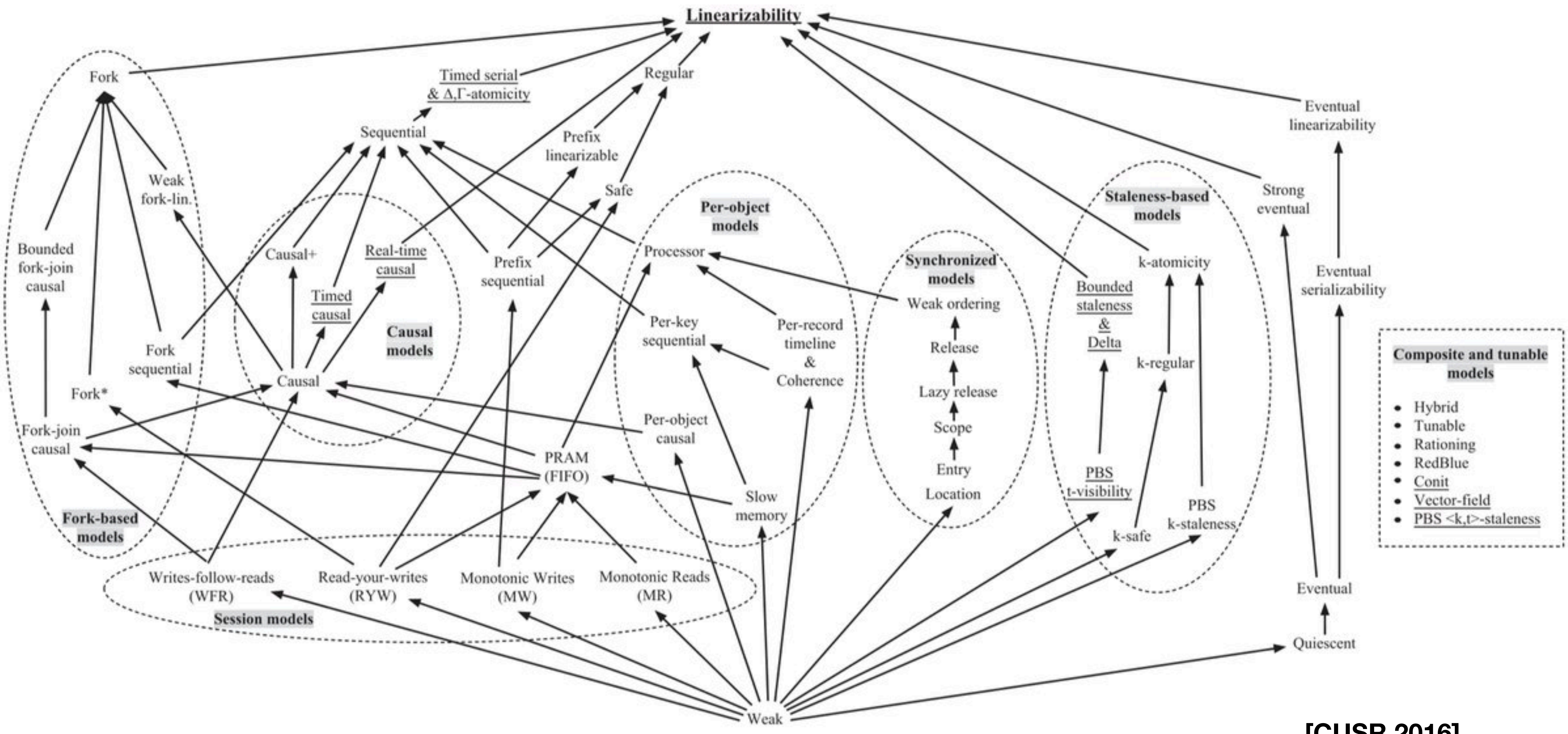
TL;DR

Linearizability is not as expensive as you might think.

Paxos and friends offer a one-size-fits-all solution to distributed consensus which may not perform best for every application.

Faster Paxos demonstrates that decentralised consensus algorithms can perform better than we previously believed.

I ❤ Linearizability

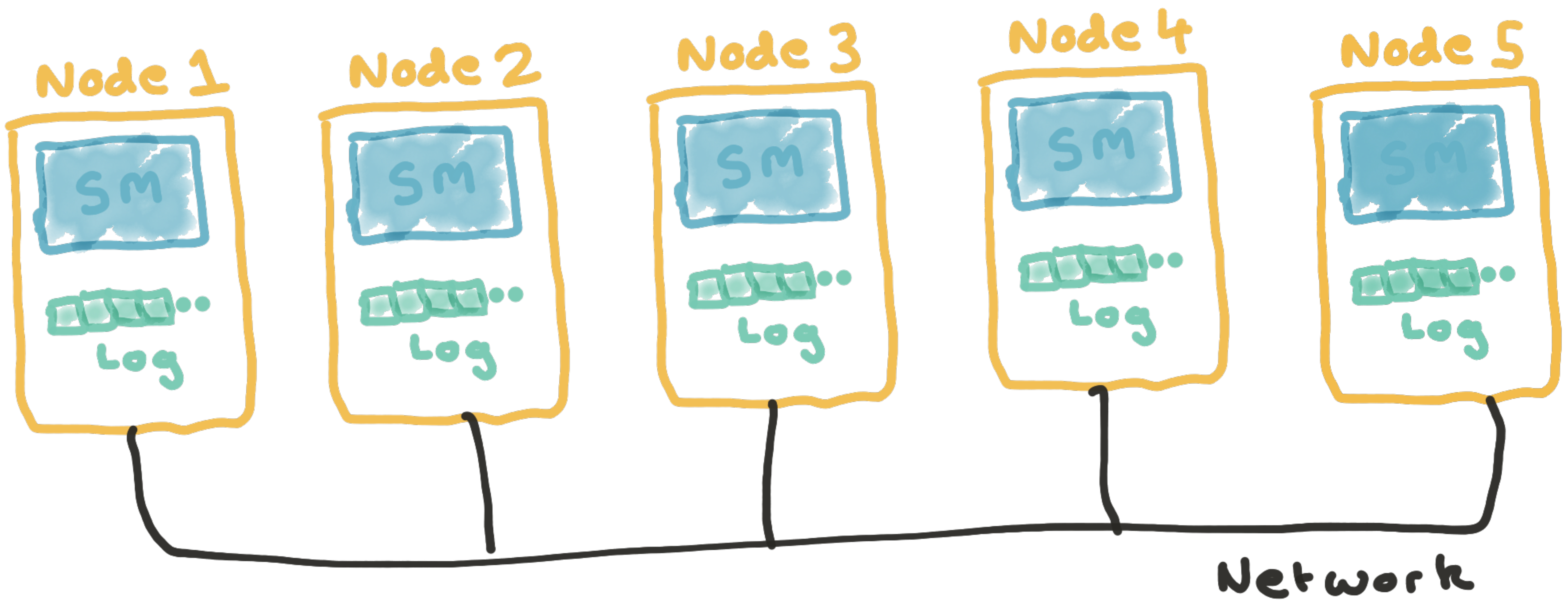

Linearizability
Fork
Timed serial
& Δ,Γ-atomicity
Regular
Sequential
Prefix
linearizable
Weak
fork-lin.
Safe
Per-object
models
Staleness-based
models
Strong
eventual
Eventual
linearizability
Bounded
fork-join
causal
Causal+
Real-time
causal
Prefix
sequential
Processor
Synchronized
models
k-atomicity
Timed
causal
Weak ordering
Bounded
staleness
&
Delta
Eventual
serializability
Per-key
sequential
Per-record
timeline
&
Coherence
Causal
models
Fork
sequential
Release
k-regular
Causal
Fork*
Lazy release
Composite and tunable
models
Hybrid
Tunable
Rationing
RedBlue
Conit
Vector-field
PBS <k,t>-staleness
Scope
Per-object
causal
Fork-join
causal
PRAM
(FIFO)
Entry
PBS
t-visibility
Location
Slow
memory
PBS
k-staleness
Fork-based
models
k-safe
Writes-follow-reads
(WFR)
Read-your-writes
(RYW)
Monotonic Writes
(MW)
Monotonic Reads
(MR)
Eventual
Session models
Quiescent
Weak

[CUSR 2016]

I  Linearizability

But I don’t trust 

State Machine Replication



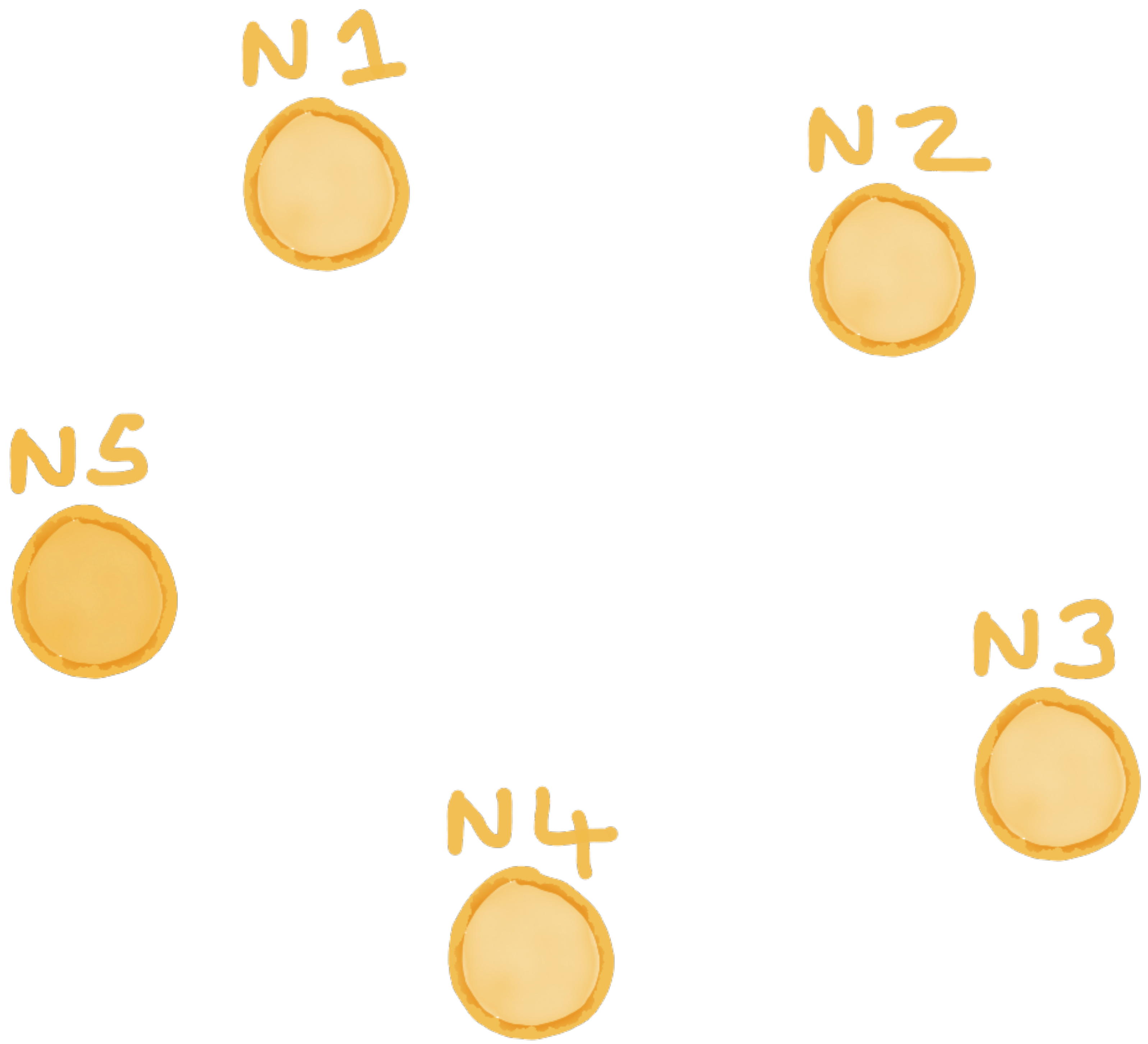
Multi-Paxos

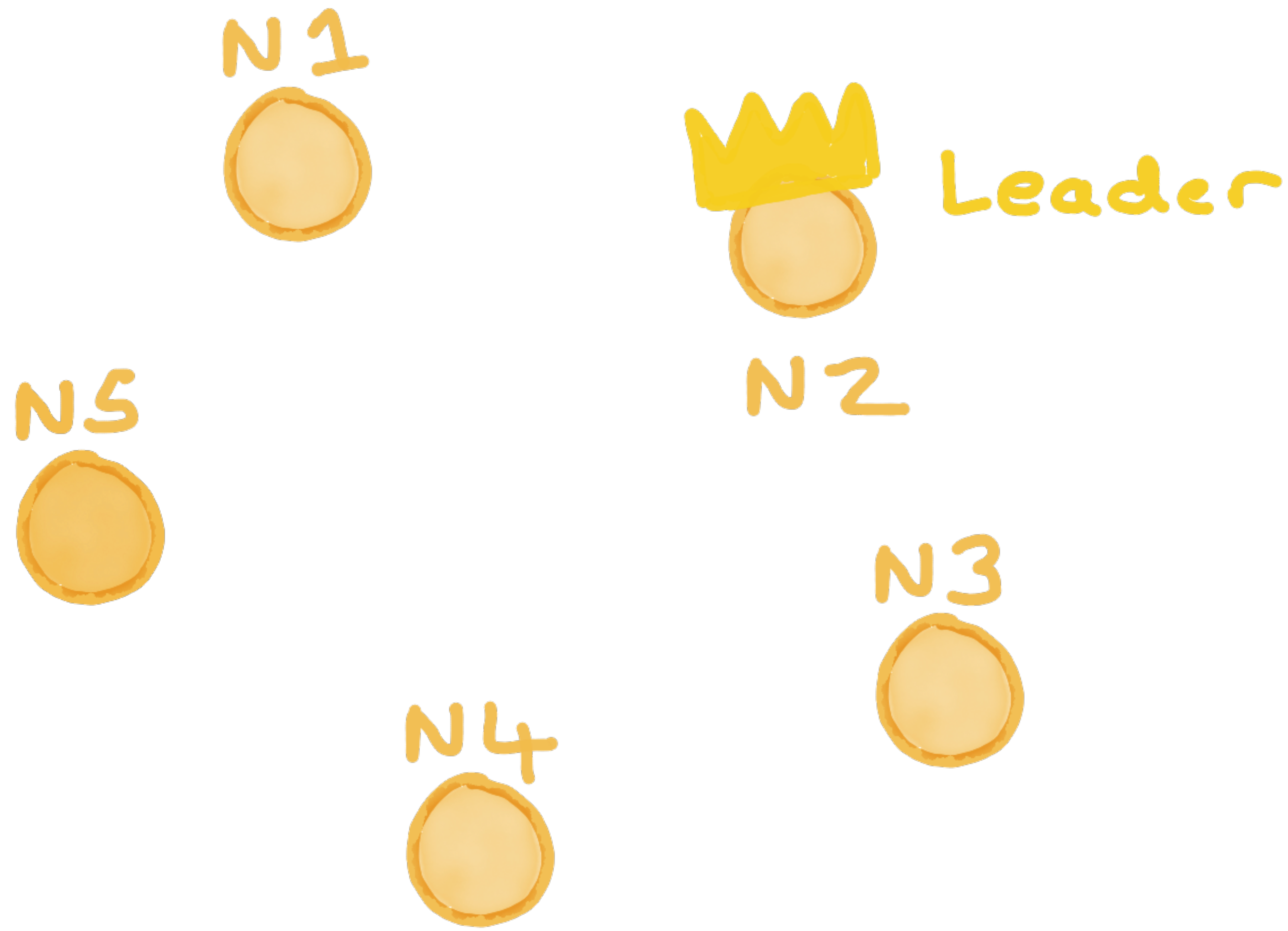
1. Select a node to be leader.

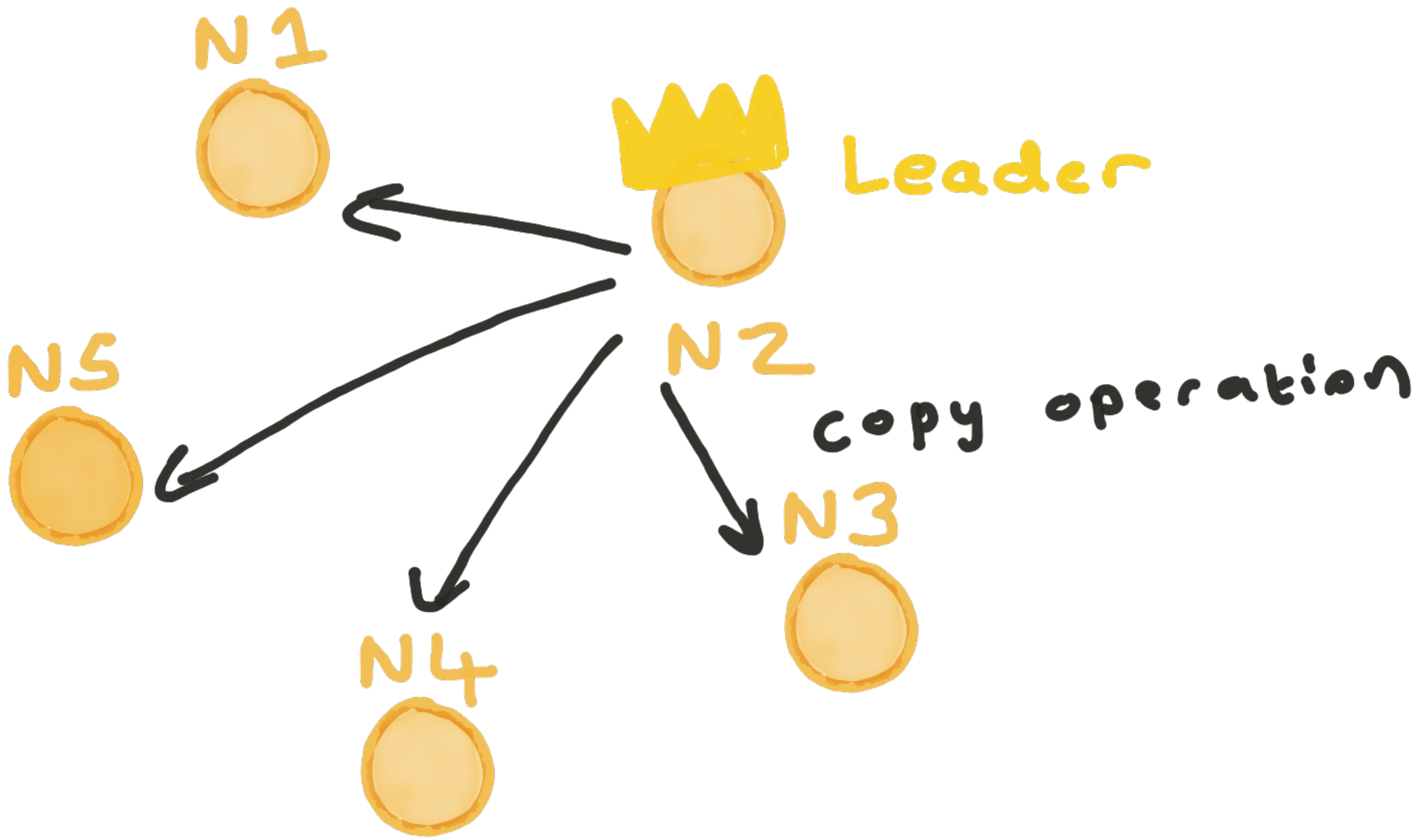
2. Nodes send operations to leader, the leader orders them and replicates them to the other nodes.

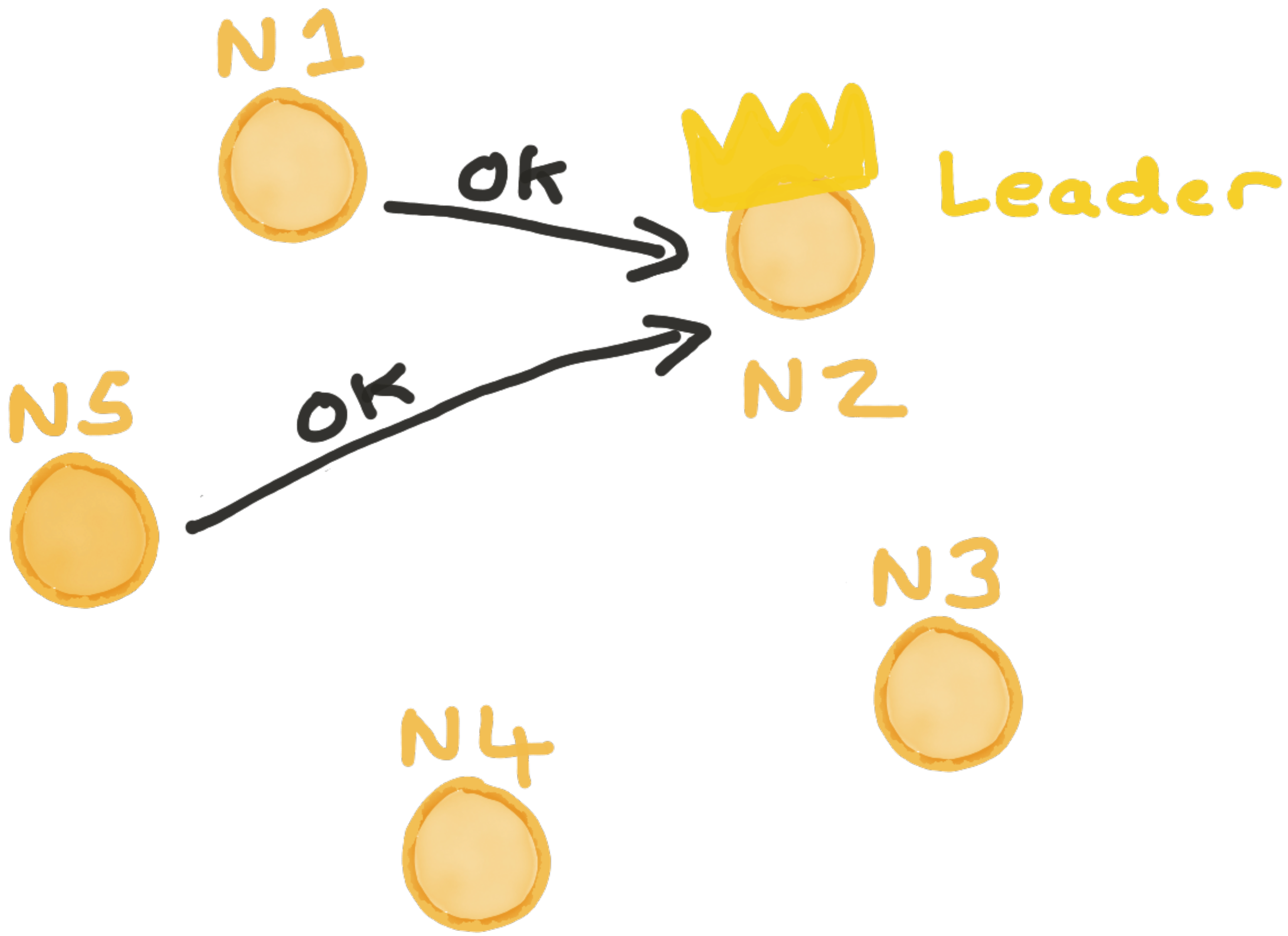
3. Once an operation has been replicated to a majority of nodes then it can be applied to the state machines.

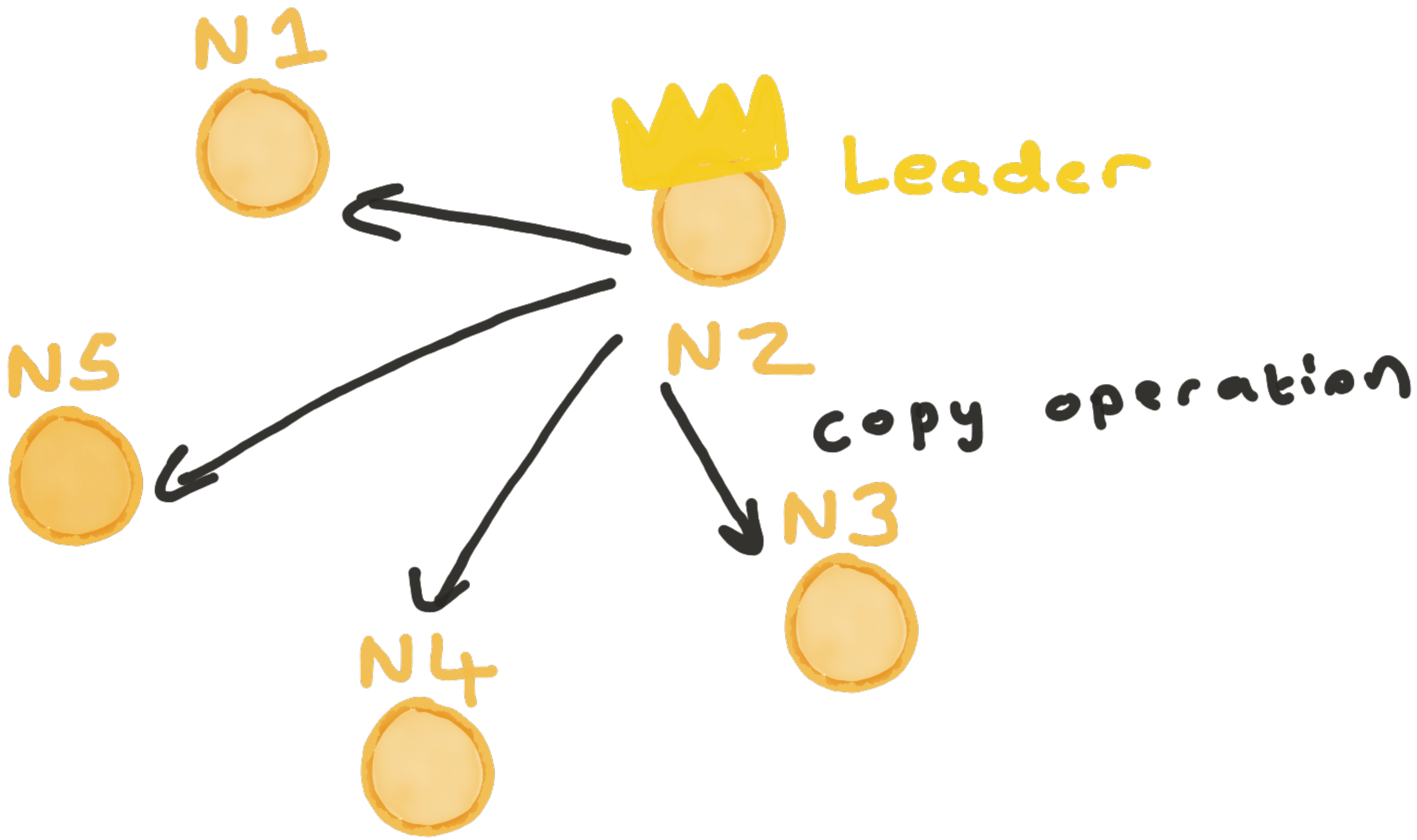
4. If the leader fails, it is replaced by another node. This process requires agreement from a majority of nodes.

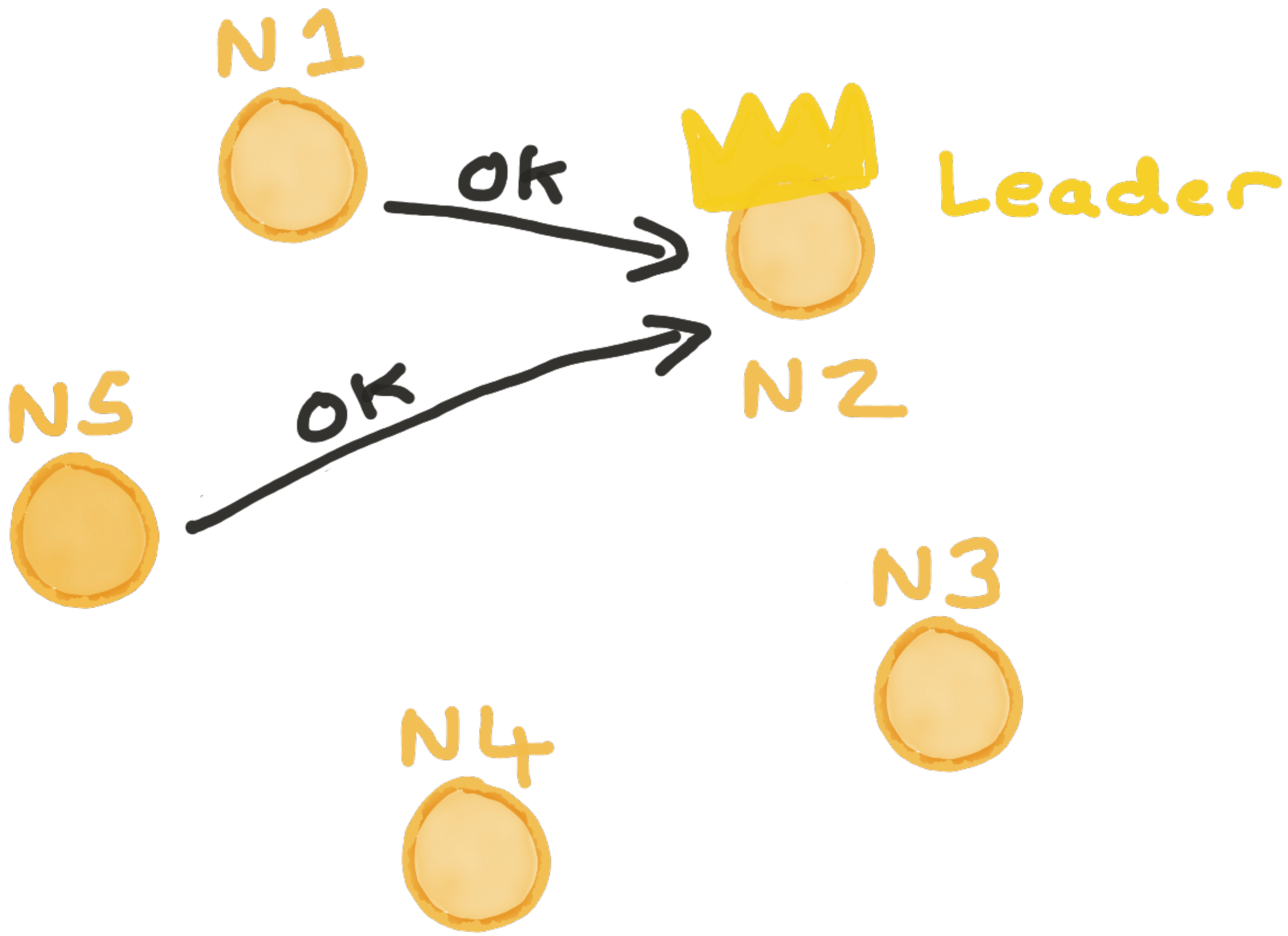

N1
N2
N5
N3
N4

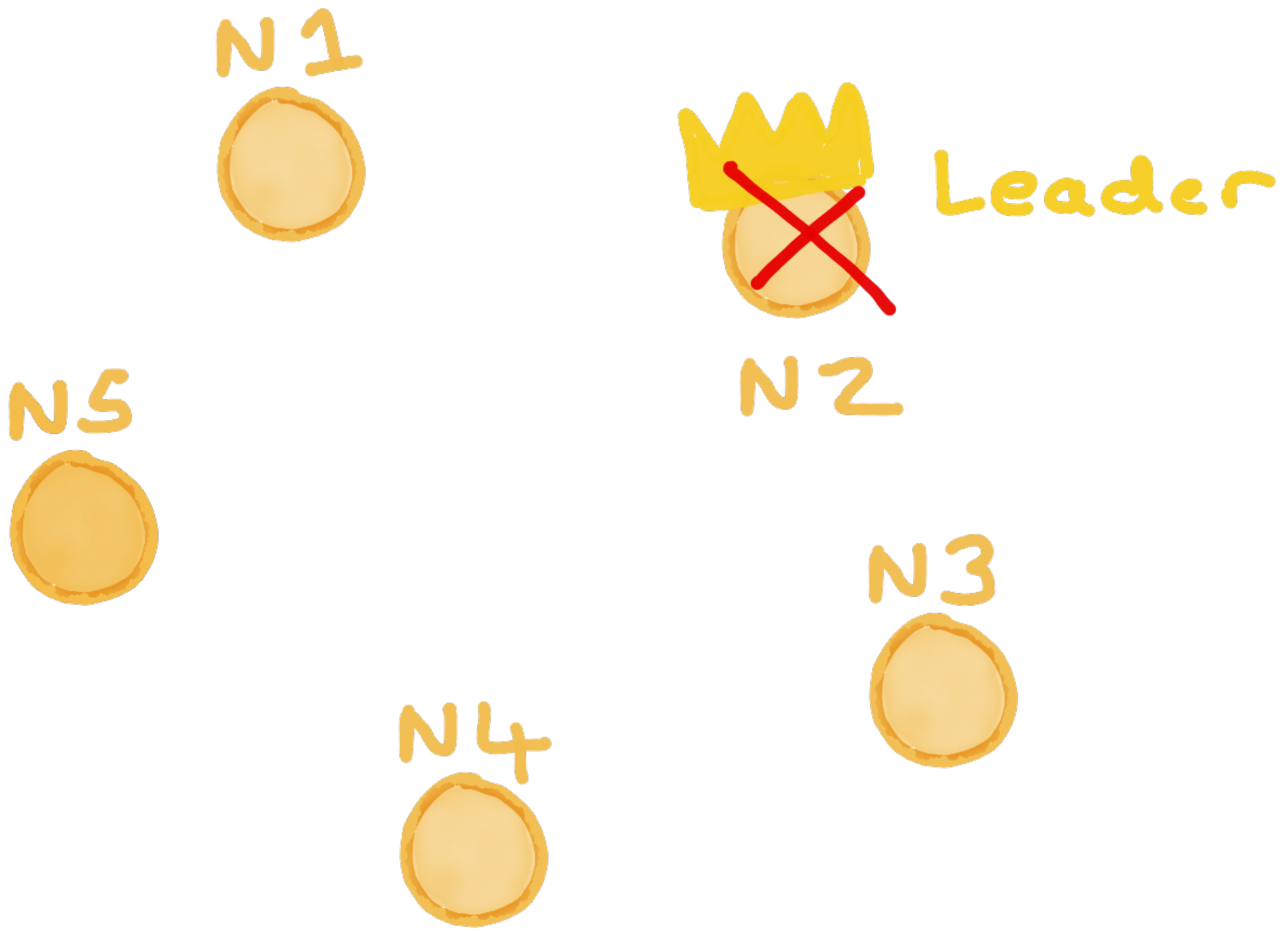

N1
Leader
N2
N5
N3
N4

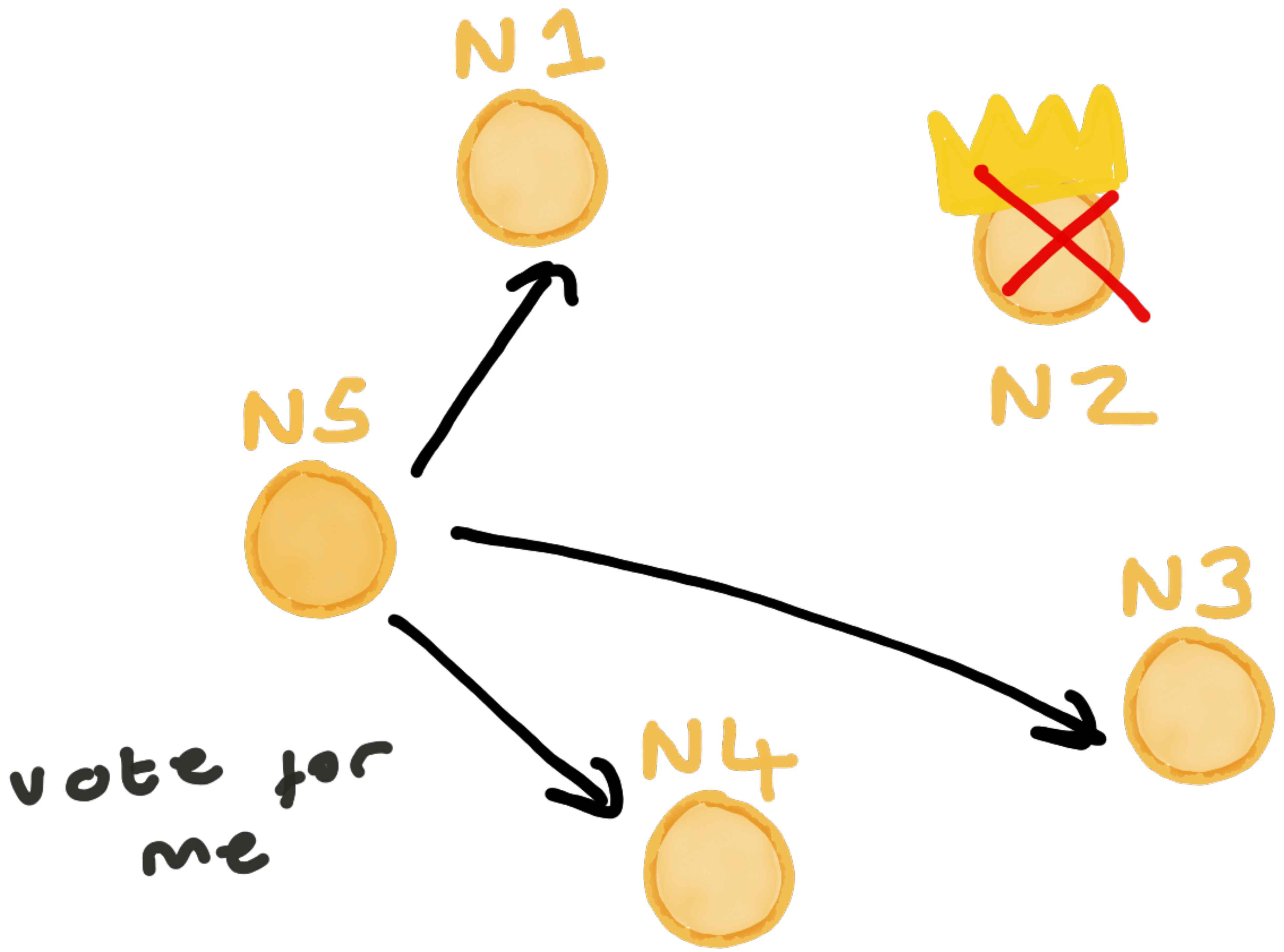

N1
Leader
N5
N2
copy operation
N3
N4

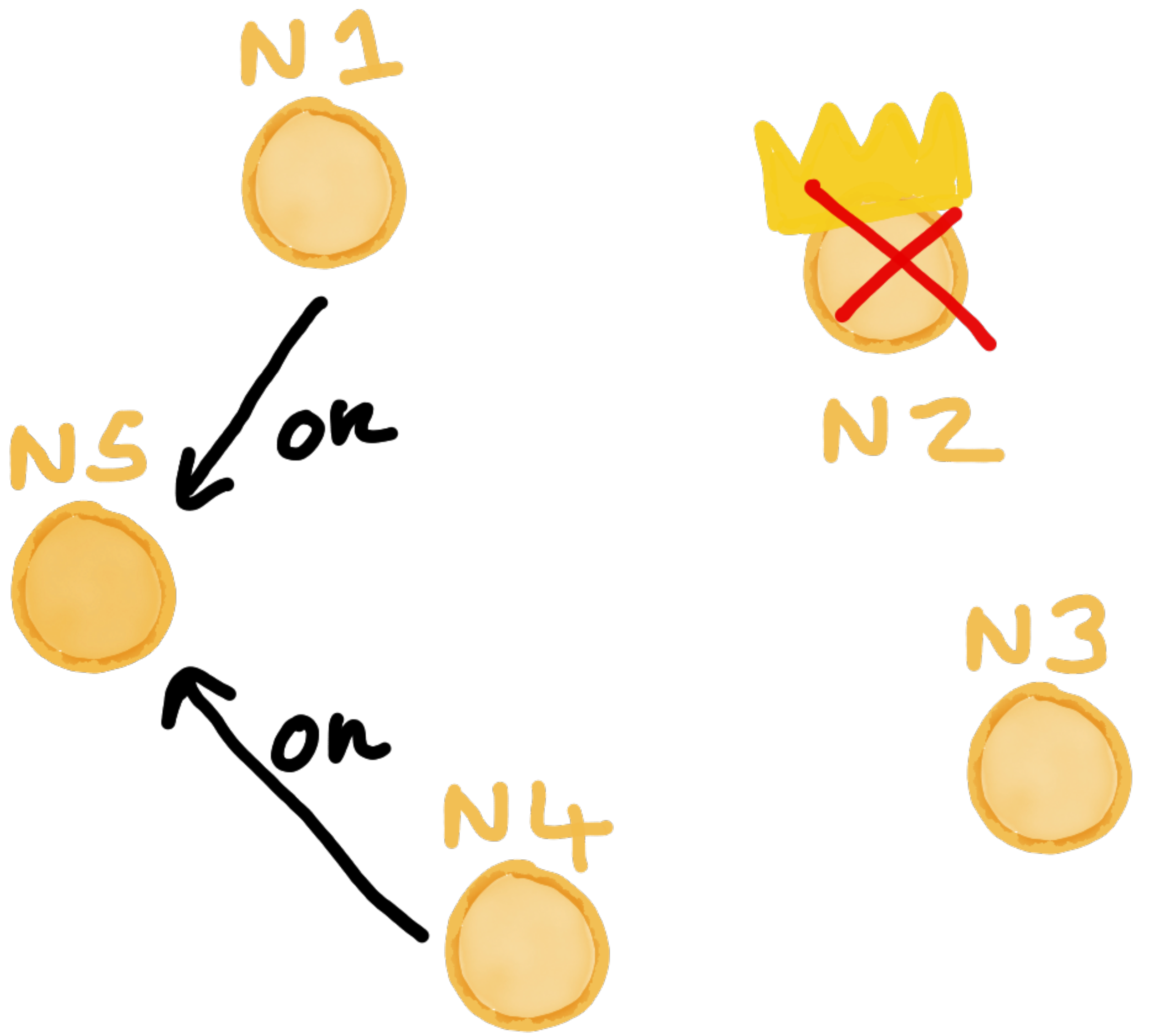

N1
OK
Leader
OK
N2
N5
N3
N4

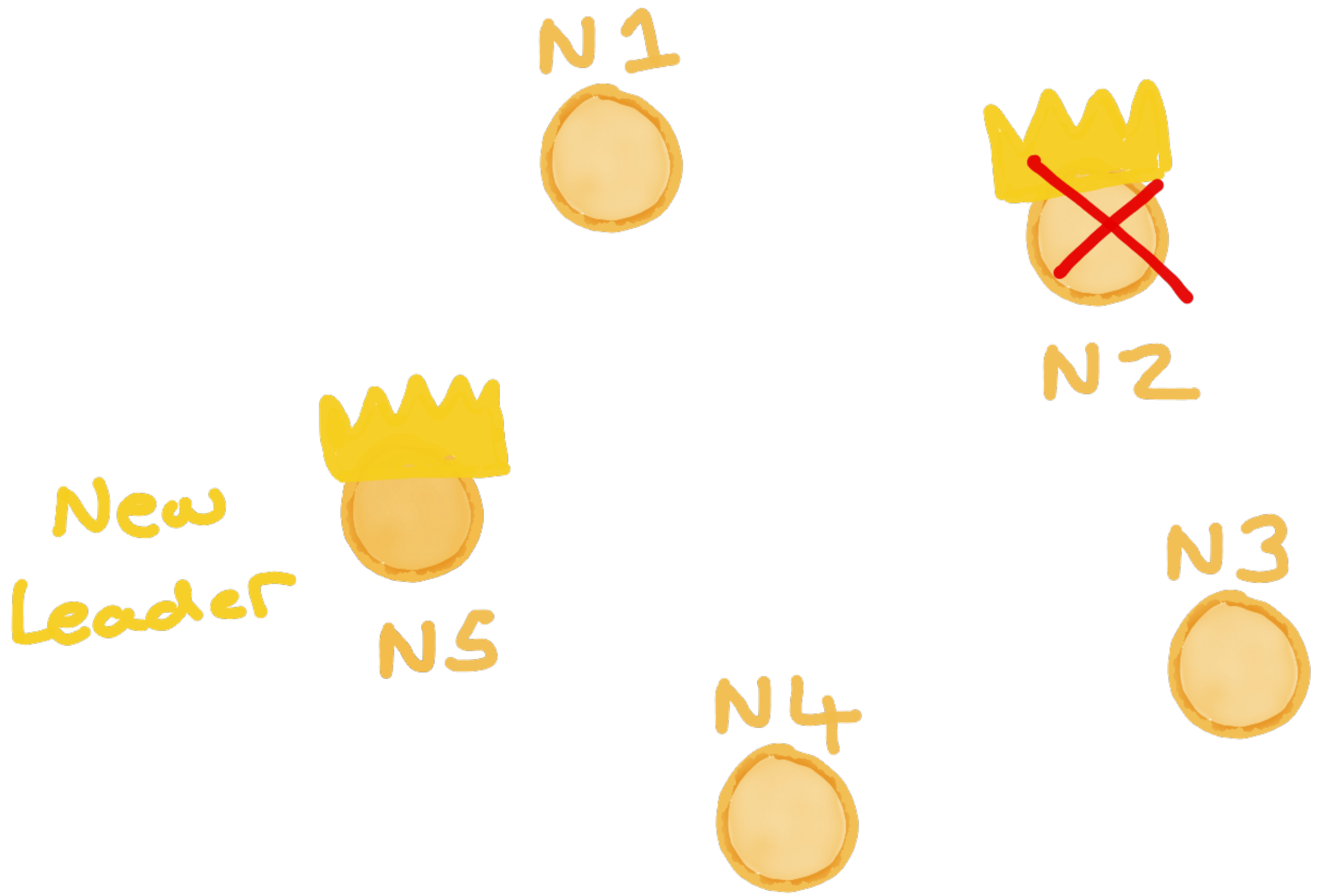

N1
Leader
N5
N2
copy operation
N3
N4

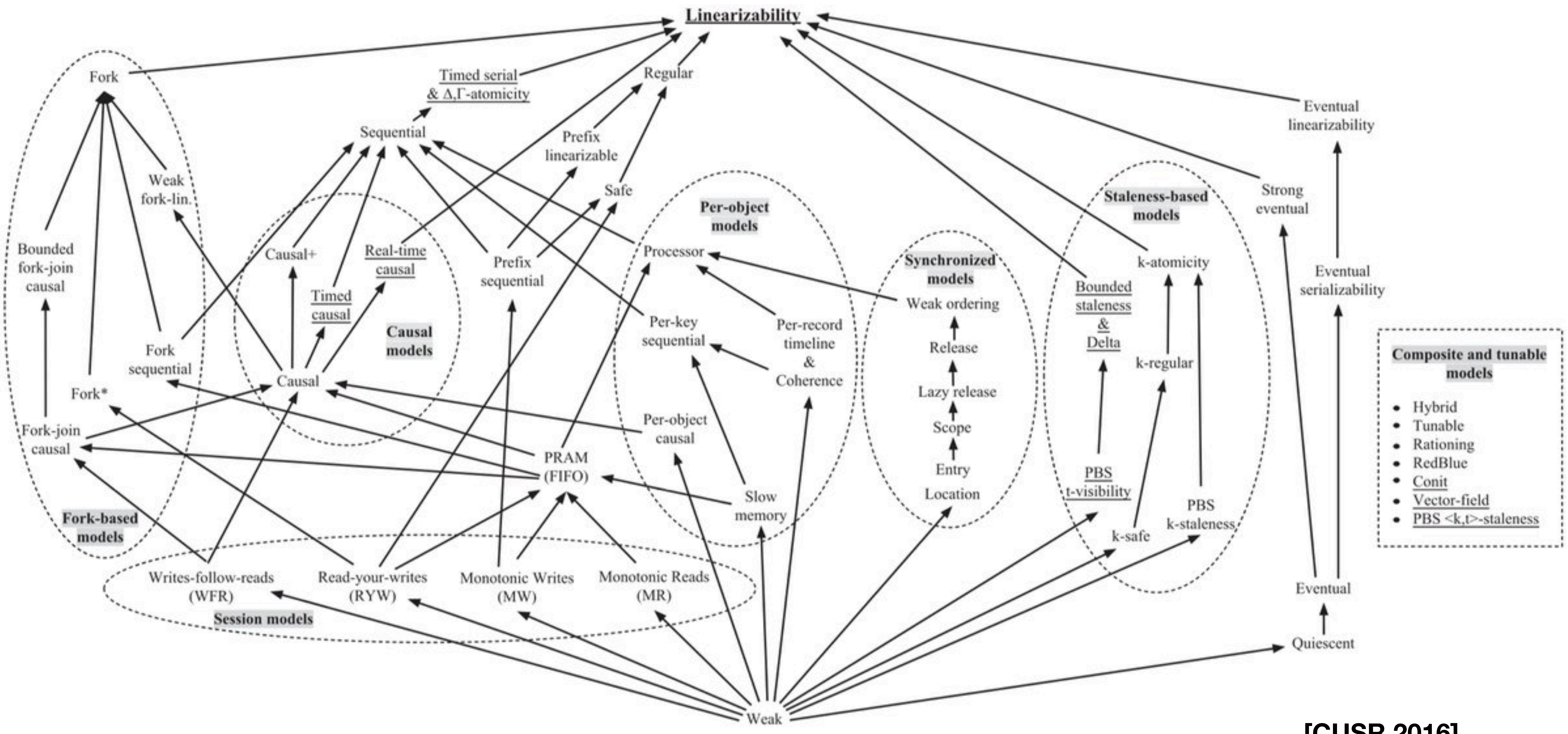

N1
OK
Leader
N5
OK
N2
N3
N4


N1
Leader
N2
N5
N3
N4


N1
N2
N5
N3
N4
vote for me


N1
N2
N5
on
on
N4
N3


N1
N2
New Leader
N5
N3
N4



Linearizability
Fork
Timed serial & Δ,Γ-atomicity
Regular
Sequential
Prefix linearizable
Safe
Weak fork-lin.
Bounded fork-join causal
Causal+
Real-time causal
Timed causal
Prefix sequential
Per-object models
Processor
Causal models
Fork sequential
Fork*
Causal
Per-key sequential
Per-record timeline & Coherence
Per-object causal
Fork-join causal
PRAM (FIFO)
Slow memory
Fork-based models
Synchronized models
Weak ordering
Release
Lazy release
Scope
Entry
Location
Staleness-based models
k-atomicity
Bounded staleness & Delta
k-regular
PBS t-visibility
k-safe
PBS k-staleness
Eventual linearizability
Strong eventual
Eventual serializability
Composite and tunable models
Hybrid
Tunable
Rationing
RedBlue
Conit
Vector-field
PBS <k,t>-staleness
Writes-follow-reads (WFR)
Read-your-writes (RYW)
Monotonic Writes (MW)
Monotonic Reads (MR)
Session models
Eventual
Quiescent
Weak

[CUSR 2016]

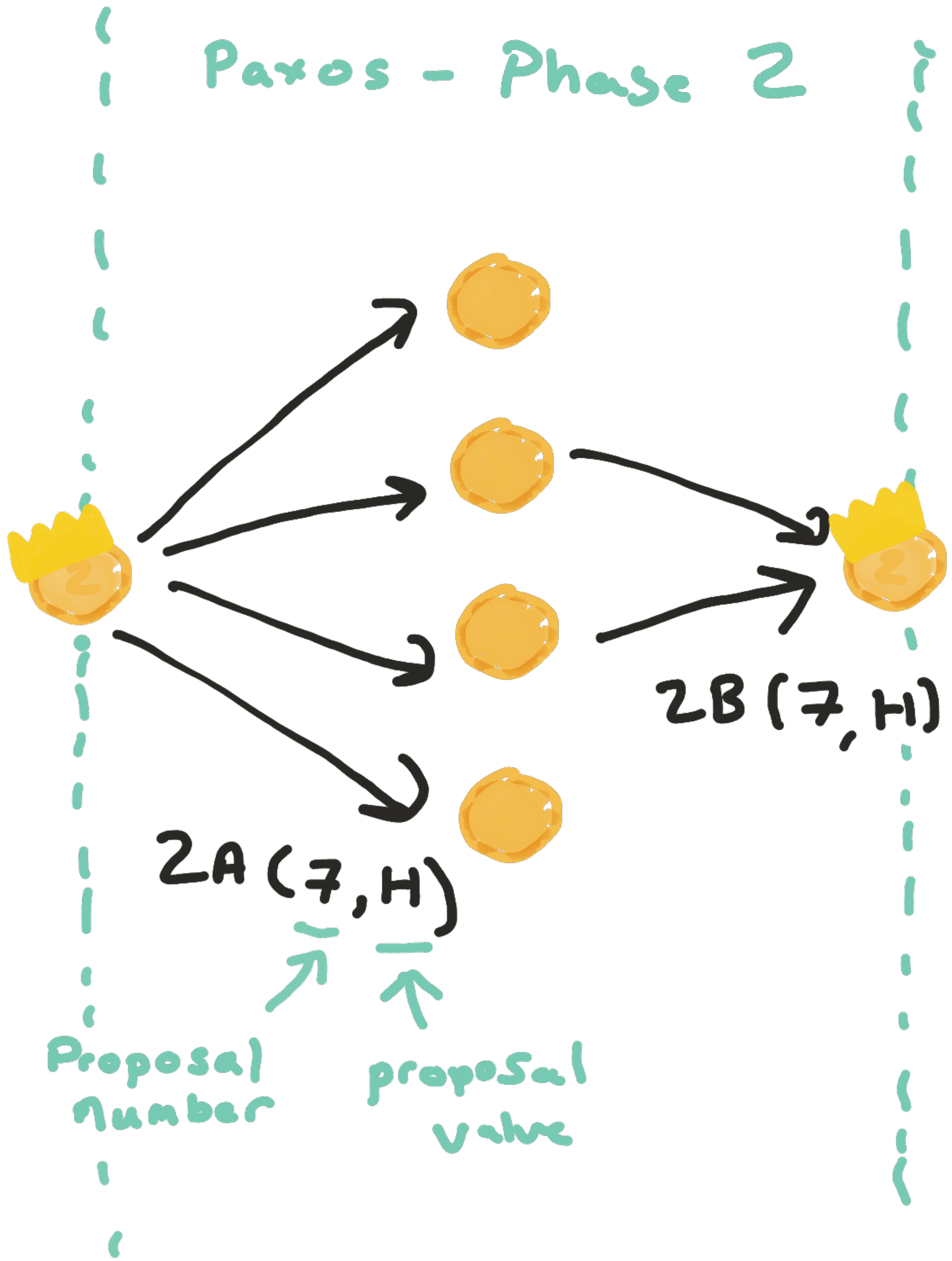
Distributed yet highly centralised

The leader is a single point of serialisation.

However, it also:

- Limits throughput as it becomes a bottleneck
- Increases latency (3 steps instead of 2)

Alternatives have been proposed but have seen little adoption.


Paxos - Phase 2
2A (7, H)
2B (7, H)
Proposal number
proposal value

Fast Paxos

Fast Paxos is similar to Paxos, except that some proposal numbers are *fast*.

These use quorums of 3/4 of nodes instead of 1/2 of nodes.

However, any node can send a value directly to the other nodes, bypassing the leader.


Distrib. Comput. (2006) 19:79–103
DOI 10.1007/s00446-006-0005-x

ORIGINAL ARTICLE

Fast Paxos

Leslie Lamport

Received: 20 August 2005 / Accepted: 5 April 2006 / Published online: 8 July 2006

Abstract As used in practice, traditional consensus algorithms require three message delays before any process can learn the chosen value. Fast Paxos is an extension of the classic Paxos algorithm that allows the value to be learned in two message delays. How and why the algorithm works are explained informally, and a TLA$^+$ specification of the algorithm appears as an appendix.

Keywords Consensus · Fault tolerance · Distributed algorithms · Paxos

1 Introduction

The consensus problem requires a set of processes to choose a single value. This paper considers the consensus problem in an asynchronous message-passing system subject to non-Byzantine faults. A solution to this problem must never allow two different values to be chosen despite any number of failures, and it must eventually choose a value if enough processes are nonfaulty and can communicate with one another.

In the traditional statement of the consensus problem, each process proposes a value and the chosen value must be one of those proposed values. It is not hard to see that any solution requires at least two message delays before any process learns what value has been chosen [3]. A number of algorithms achieve this delay in the best case. The classic Paxos algorithm [7,9] is popular because it achieves the optimal delay in the normal case when used in practical systems [12].

The apparently optimal number of message delays required by traditional consensus algorithms is illusory – an artifact of the traditional problem statement in which values are chosen by the same processes that propose them. In many applications, values are not proposed by the same processes that choose the value. For example, in a client/server system, the clients propose the next command to be executed and the servers choose one proposed command. When a traditional consensus algorithm is used in such a system, three message delays are required between when a client proposes a command and when some process learns which command has been chosen.

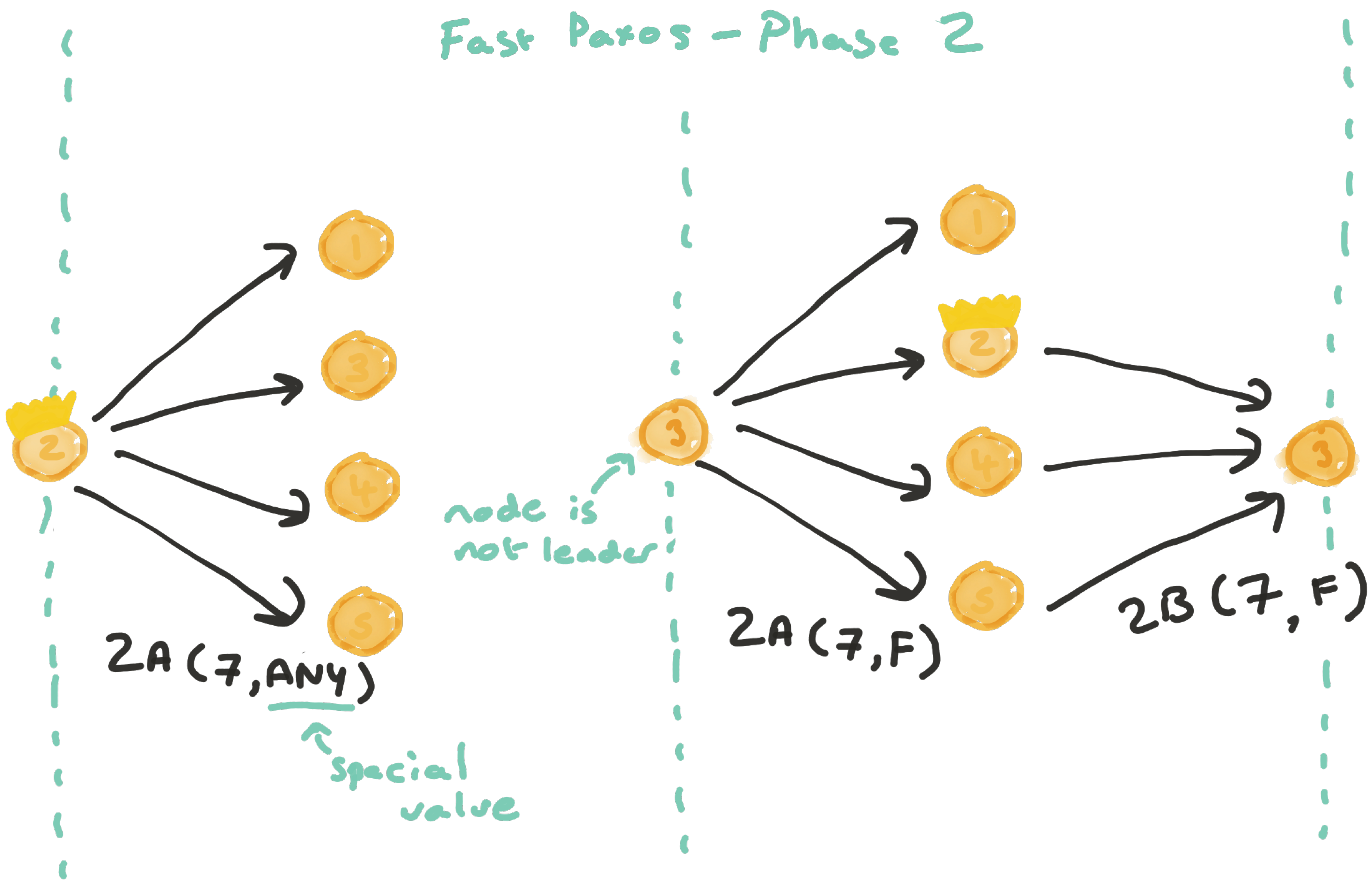
A fast consensus algorithm is one in which a process can learn the chosen value within two message delays of when it is proposed, even if values are proposed and chosen by different sets of processes. It has been shown that no general consensus algorithm can guarantee learning within two message delays if competing proposals collide – that is, if two different values are proposed concurrently [11]. A fast consensus algorithm therefore cannot always be fast in the event of collision.

Fast Paxos is a fast consensus algorithm that is a variant of classic Paxos. In the normal case, learning occurs in two message delays when there is no collision and can be guaranteed to occur in three message delays even with a collision. Moreover, it can achieve any desired degree of fault tolerance using the smallest possible number of processes.

The basic idea behind Fast Paxos also underlies an earlier algorithm of Brasileiro et al. [1]. However, they considered only the traditional consensus problem, so they failed to realize that their algorithm could be easily

L. Lamport (✉)
Microsoft Research,
1065 La Avenida,
Mountain View, CA 94043, USA

[DC 2006]

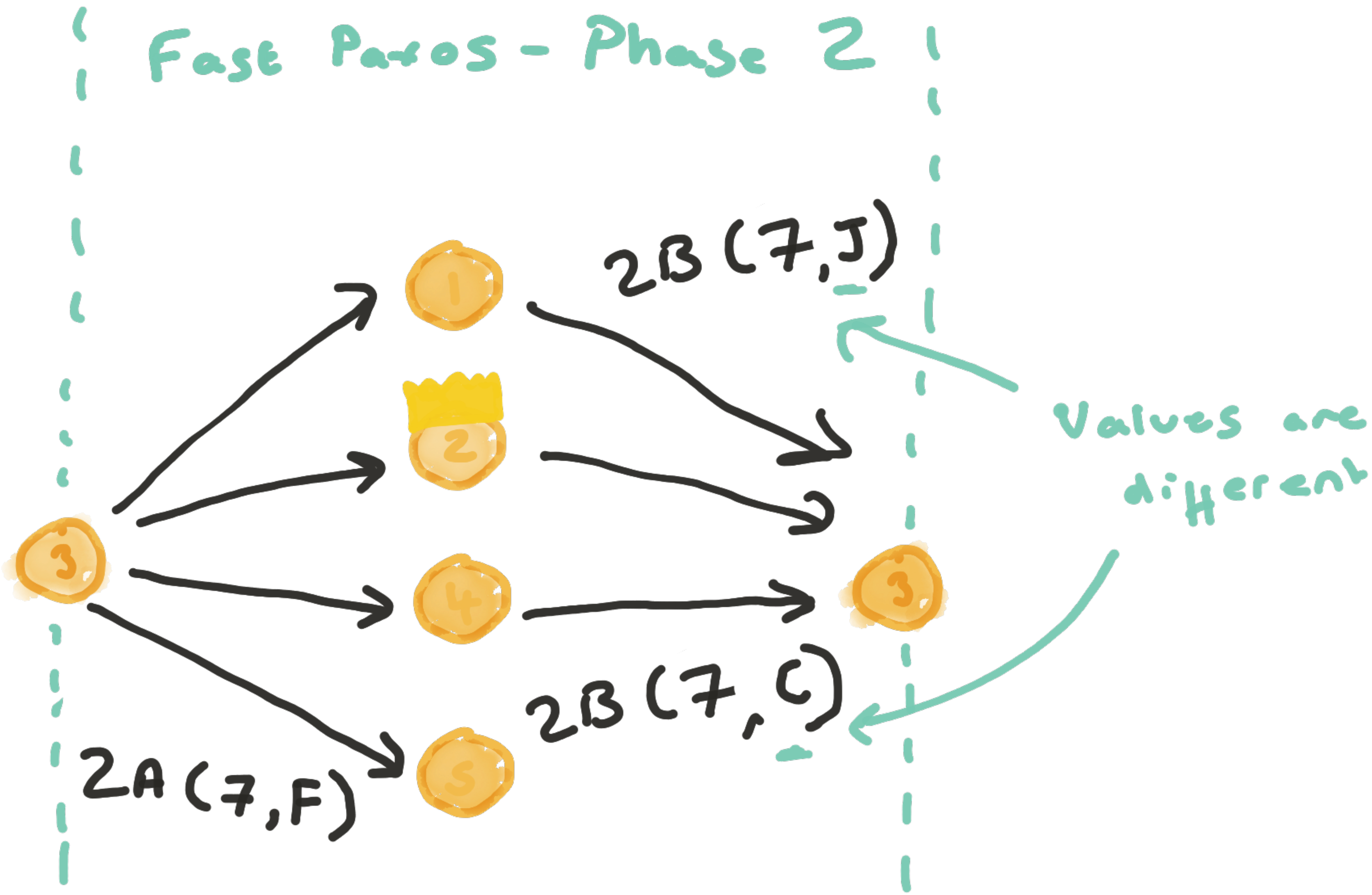


Fast Paxos - Phase 2
2A (7, ANY)
special value
node is not leader
2A (7, F)
2B (7, F)

New algorithm, new problems

Fast Paxos solves our issue with leaders but it introduces the issue of conflicts.

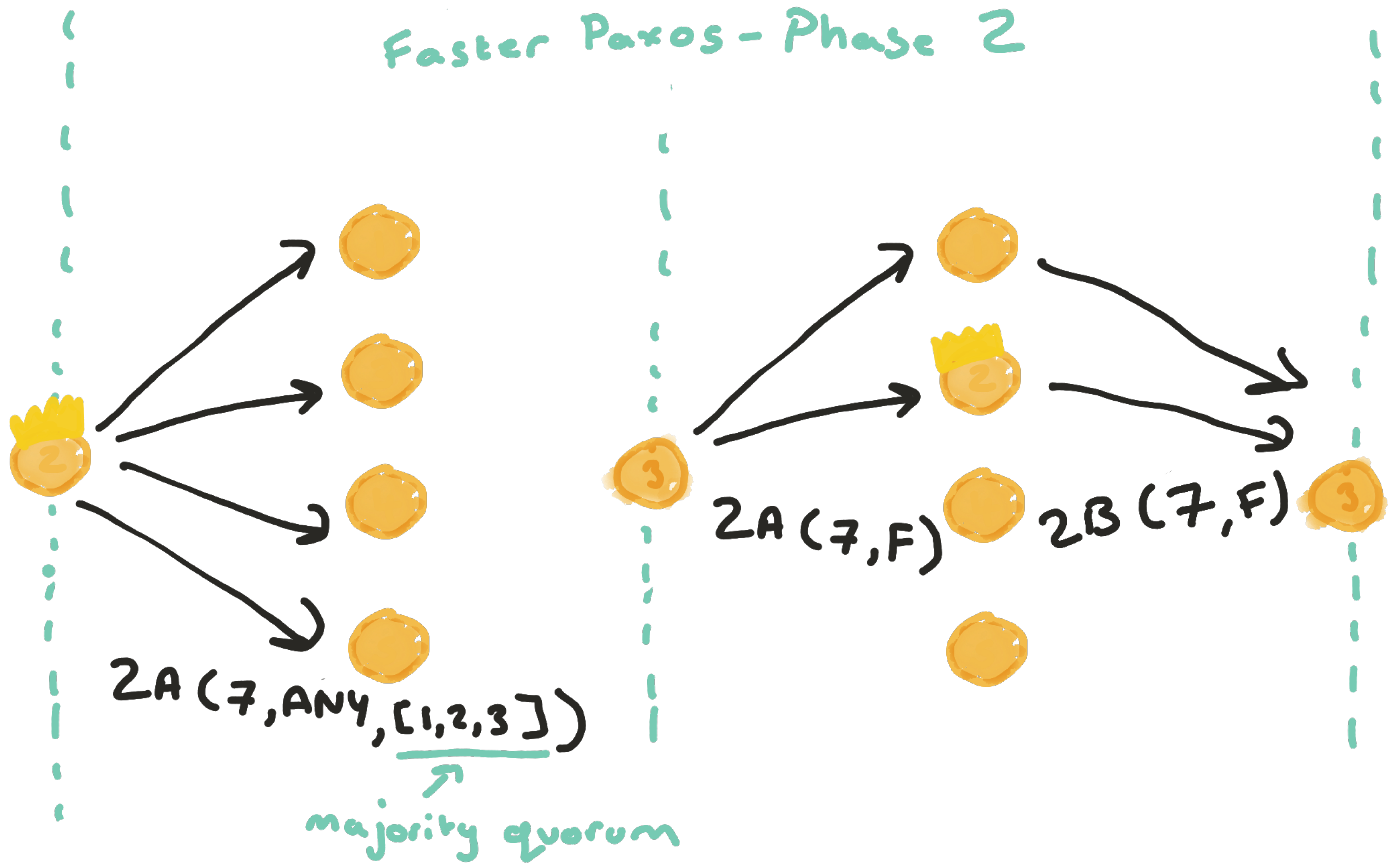
Recovering from conflicts is expensive.



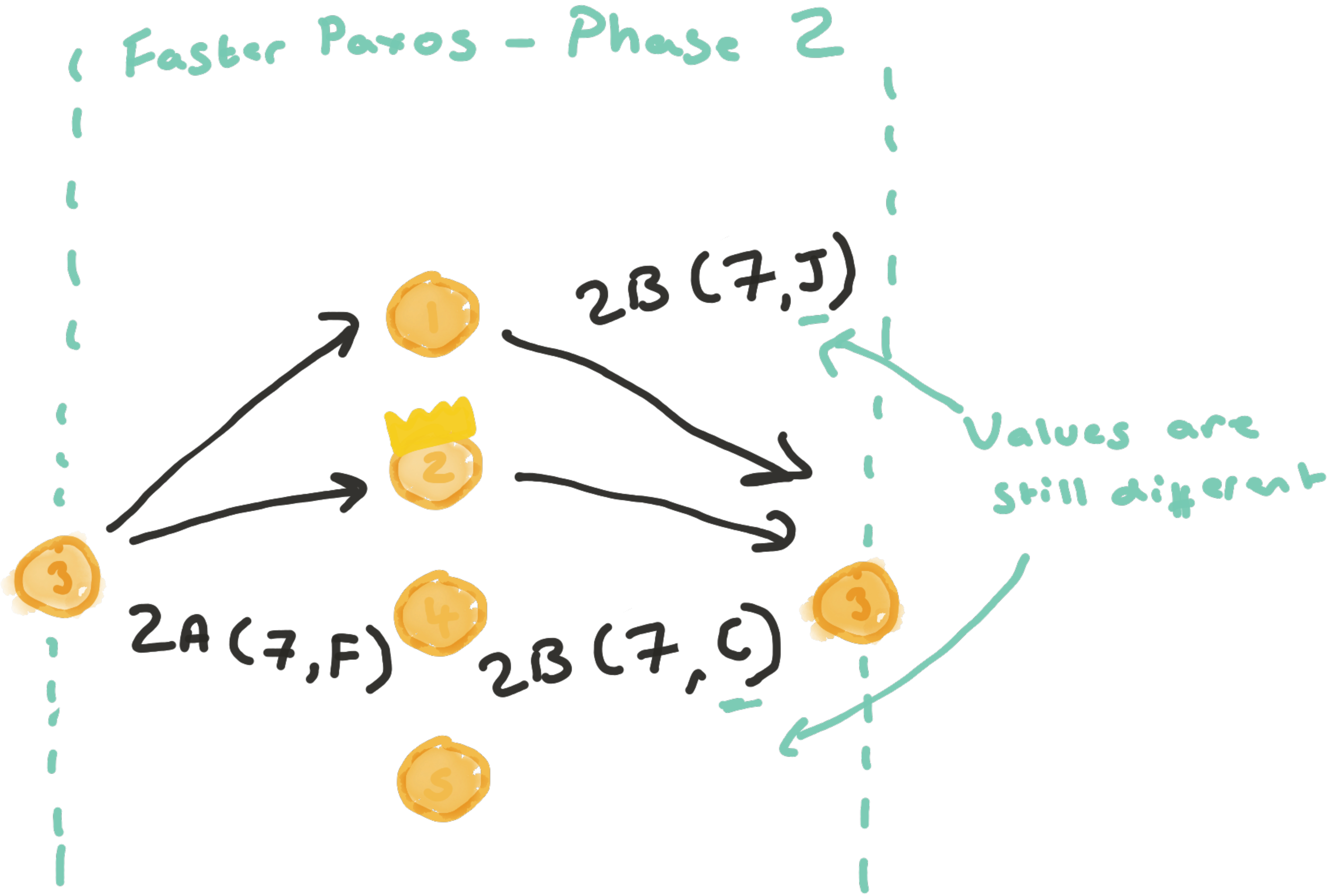
Introducing Faster Paxos

Faster Paxos is similar to Paxos, except that a leader can choose to allow any node to can send values directly to a chosen majority quorum of nodes.

Faster Paxos - Phase 2



Faster Paxos - Conflicts



Requirements of consensus

The aim of distributed consensus is to decide a single value.

Validity - The decided value must have been be proposed by some node.

Agreement - All node must learn the same decided value.

Termination - Eventually, all nodes must learn the decided value.

Is this what we actually want?

Requirements of faster consensus

The aim of distributed consensus is to decide non-empty ordered set of values.

Validity - Each decided value must have been be proposed by some node.

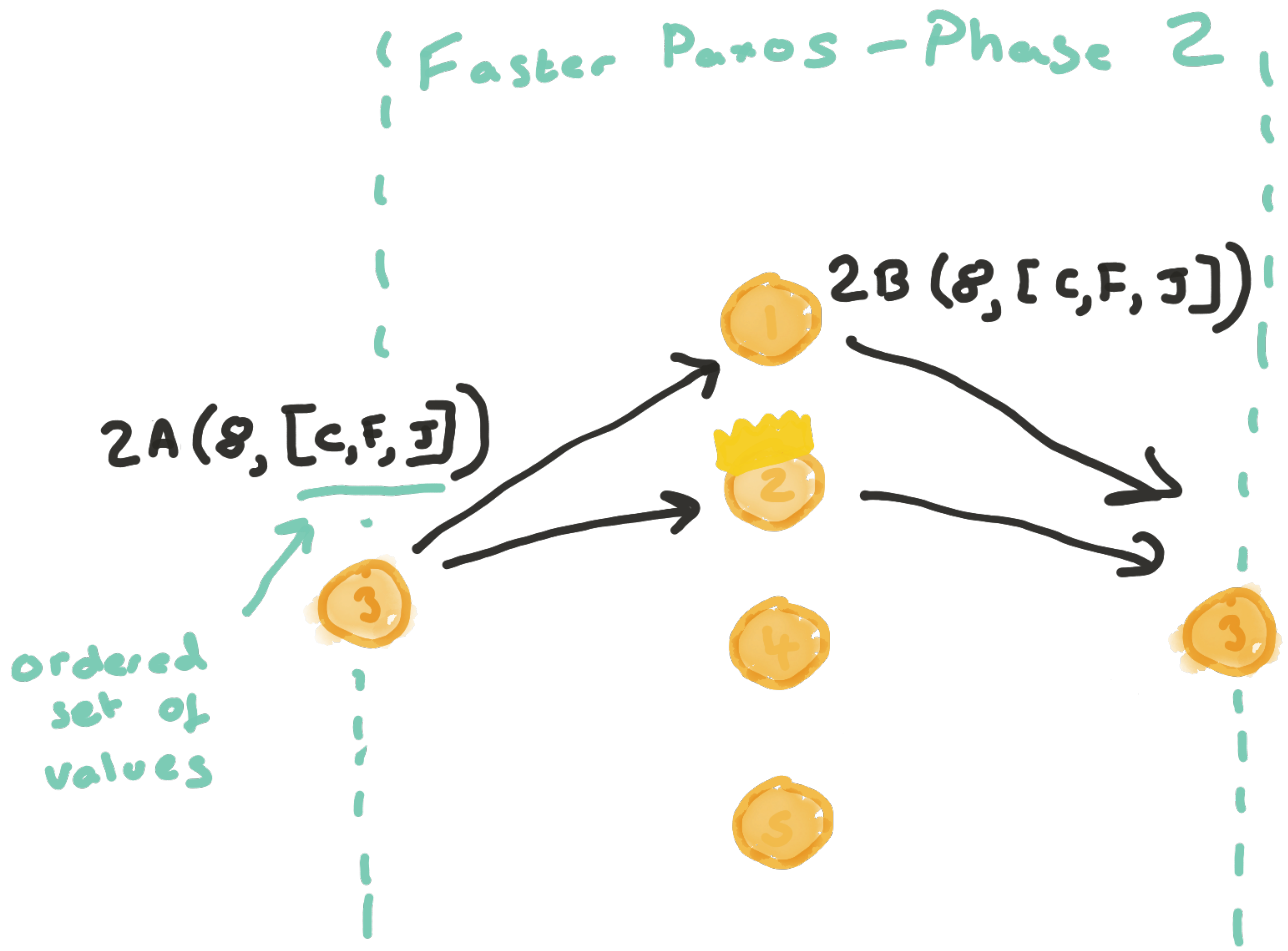
Agreement - All node must learn the same ordered set of decided values.

Termination - Eventually, all nodes must learn the set of decided values.

Faster Paxos - Conflict resolution

When a conflict is detected, each node retries with the set of values it has seen.

Each node will see the same values so next time the proposals will not conflict.



Summary

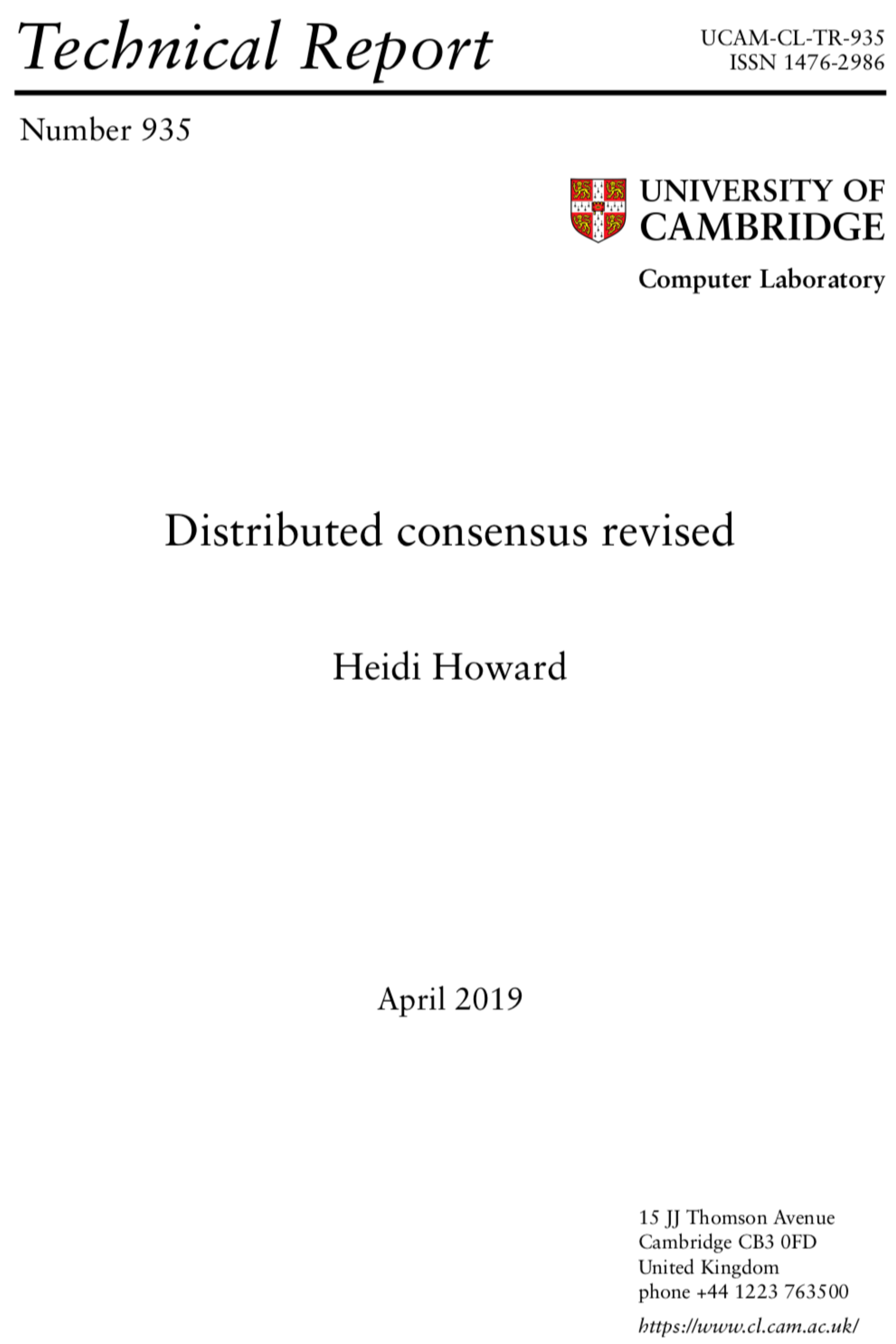
Faster Paxos solves consensus with optimal latency in the absence of conflicts/failures. Any conflict are resolved in 2 additional steps.

Faster Paxos however would not be suit to systems where:

- Performance is not a concern.
- Node failures and/or network partitions are very common.
- One node is the source of most options.

Faster Paxos is one of many options

[Flexible Paxos: Quorum intersection revisited, 2016]



[Distributed consensus revised, 2018]

[A generalised solution to distributed consensus, 2019]

Q & A

Heidi Howard @ Cambridge University
heidi.howard@cl.cam.ac.uk