Understanding Consensus Using Go

- Aditya Relangi

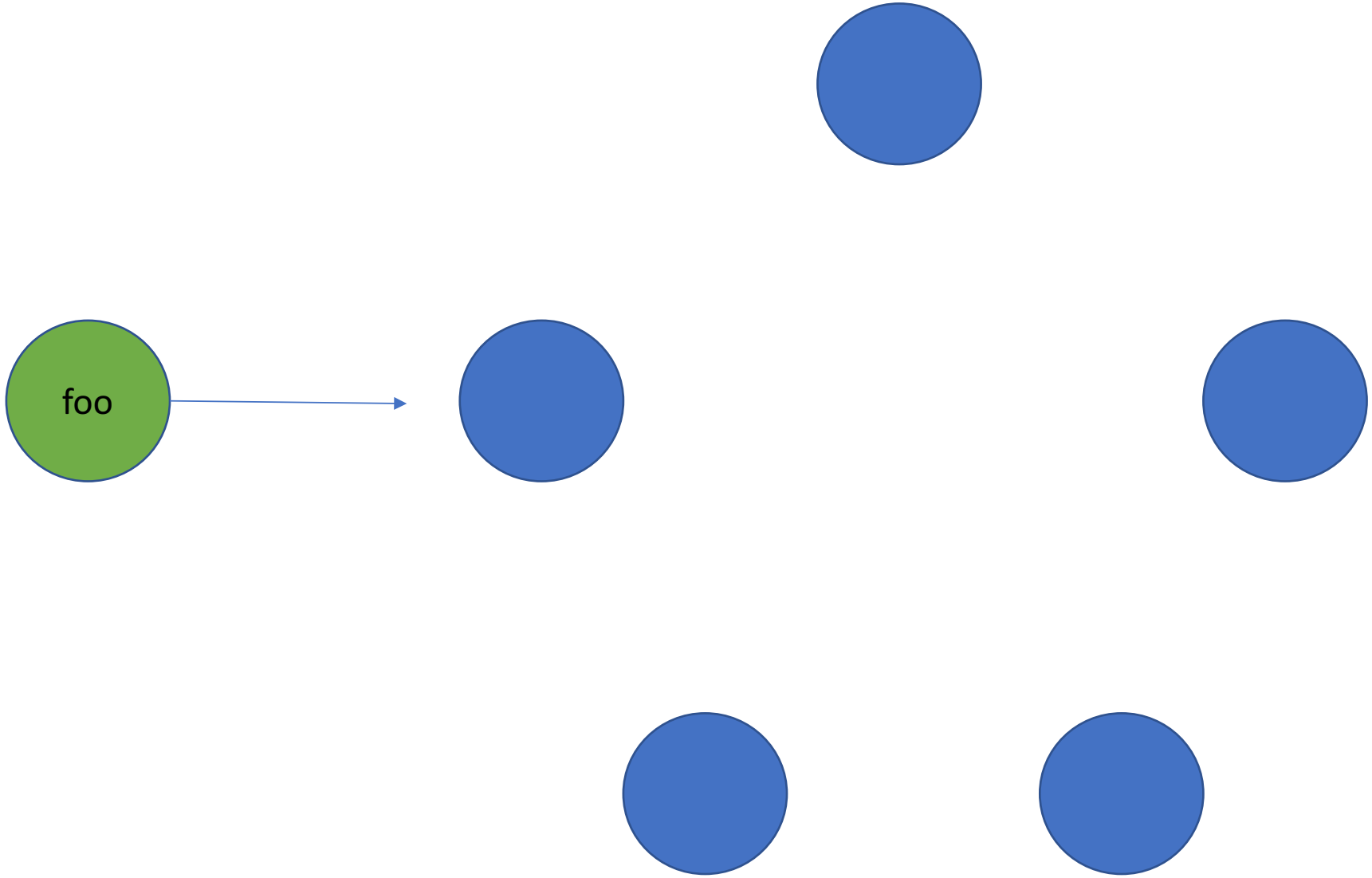
Outline

- Single Node System
- Multi Node System
- Failure Modes
- What is Consensus?
- Single Commit? Is that even feasible?
- Two Phase Commit
- Three Phase Commit
- Paxos

Single Node System

Single Node System

Multi Node System



Failure Modes

Fail – Stop

Fail – Recover

Byzantine Failures

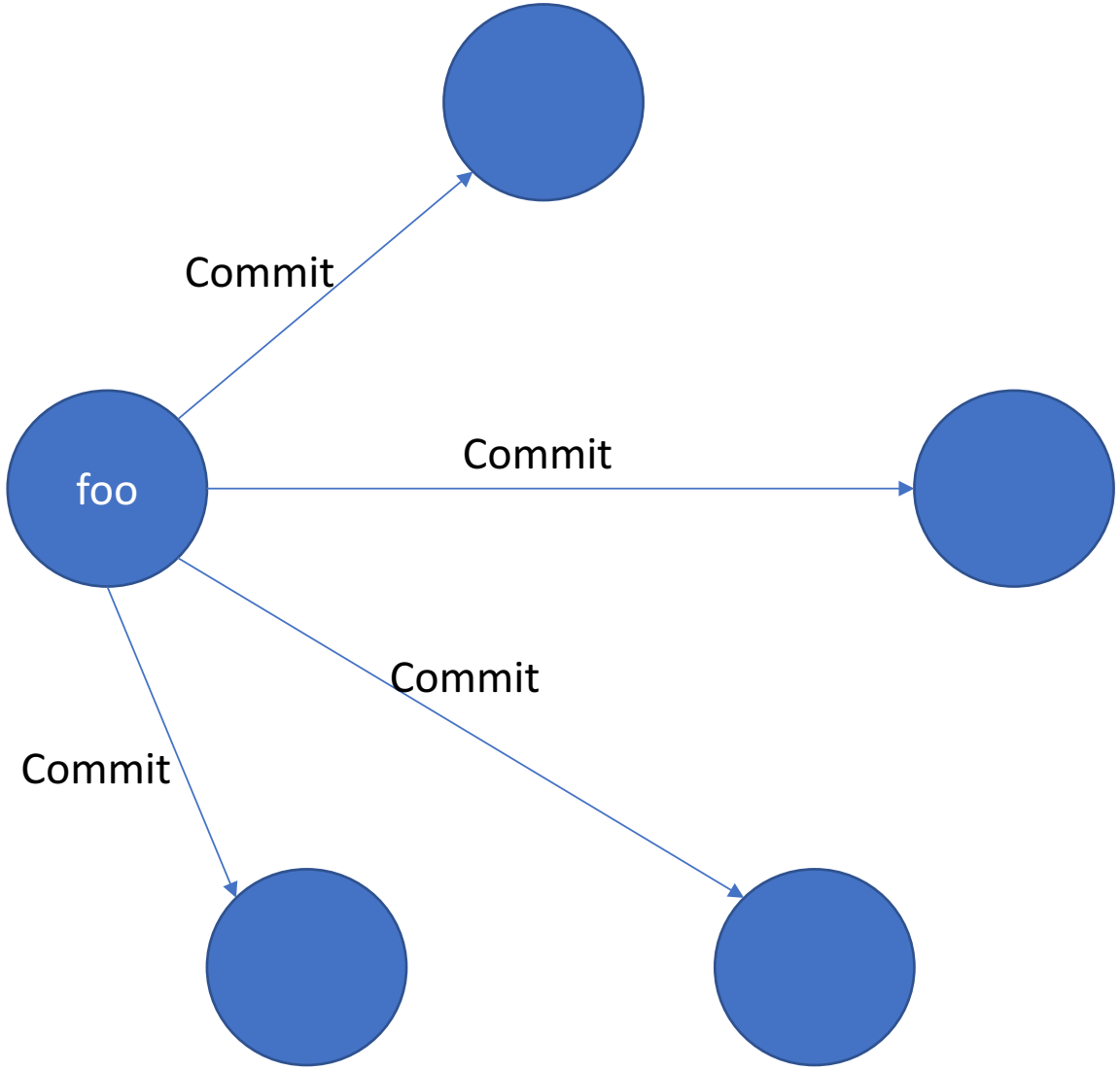
The *consensus problem* is the problem of getting a set of nodes in a distributed system to agree on something – it might be a value, a course of action

Define a Solution to the Consensus Problem

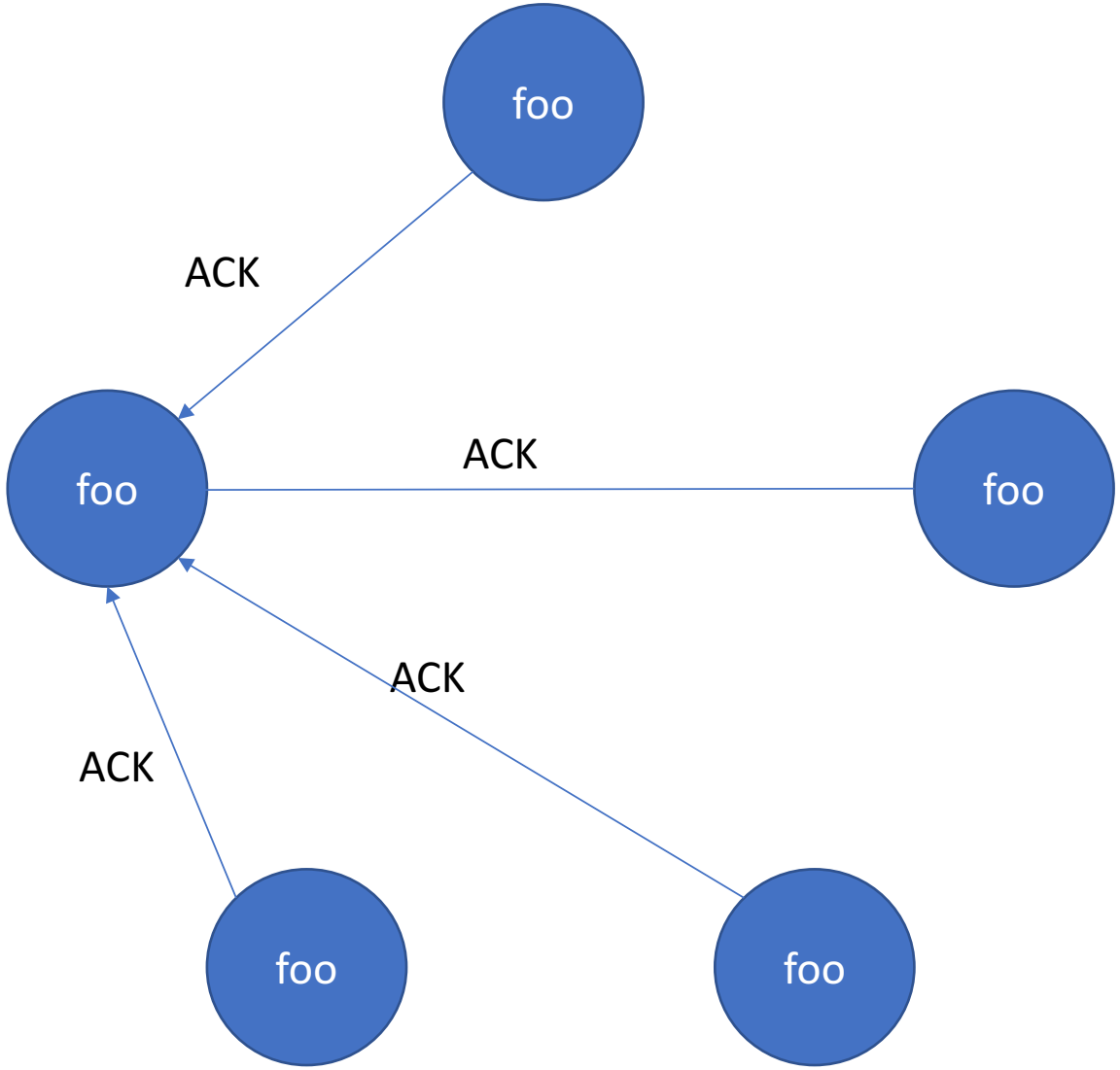
Given a set of N nodes in a distributed System

- Agreement – all nodes in N decide on the same value
- Validity – The value that is decided upon must have been proposed by some node in N
- Termination – all nodes eventually decide

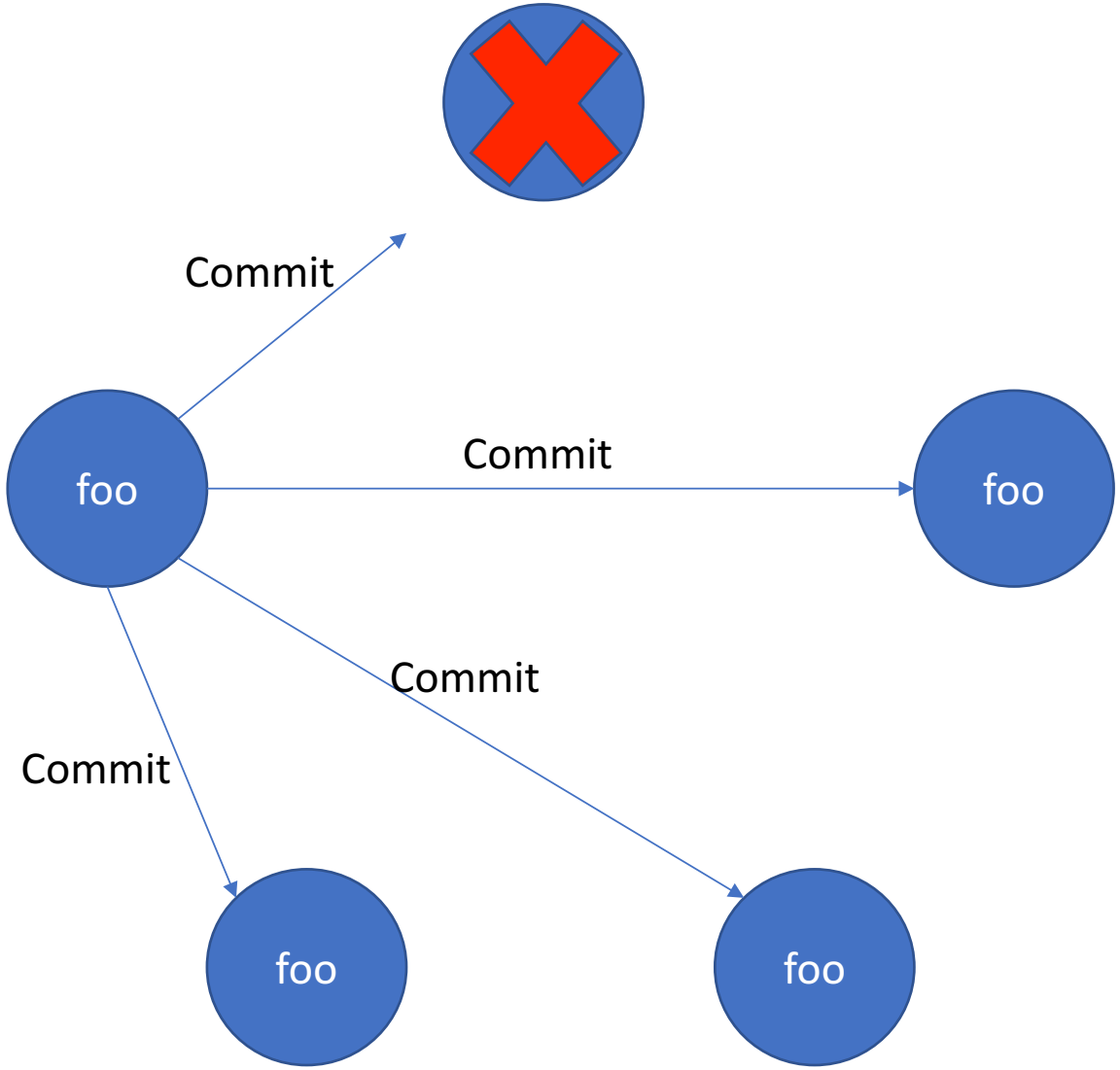
Single Commit



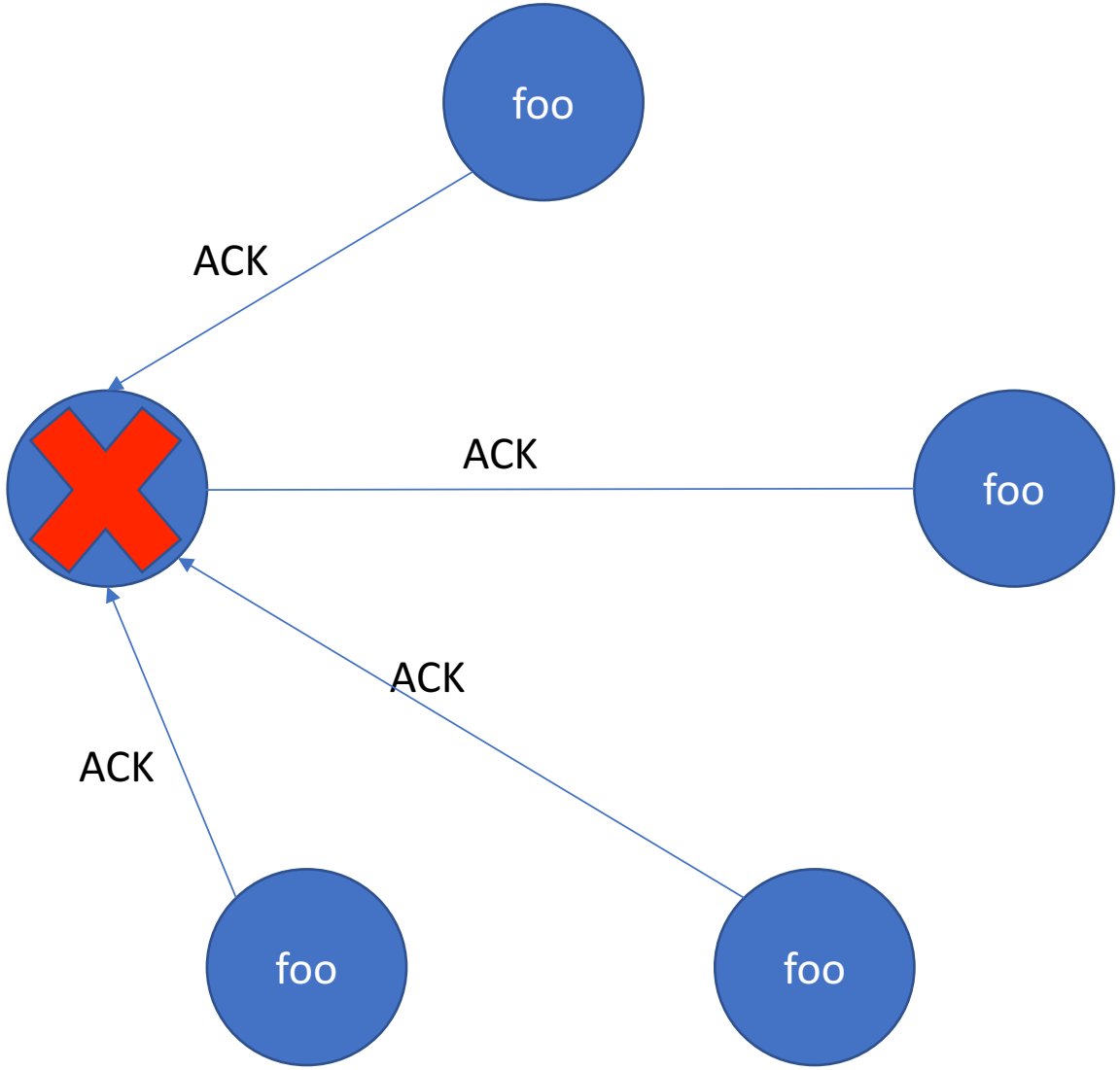
Single Commit



Single Commit

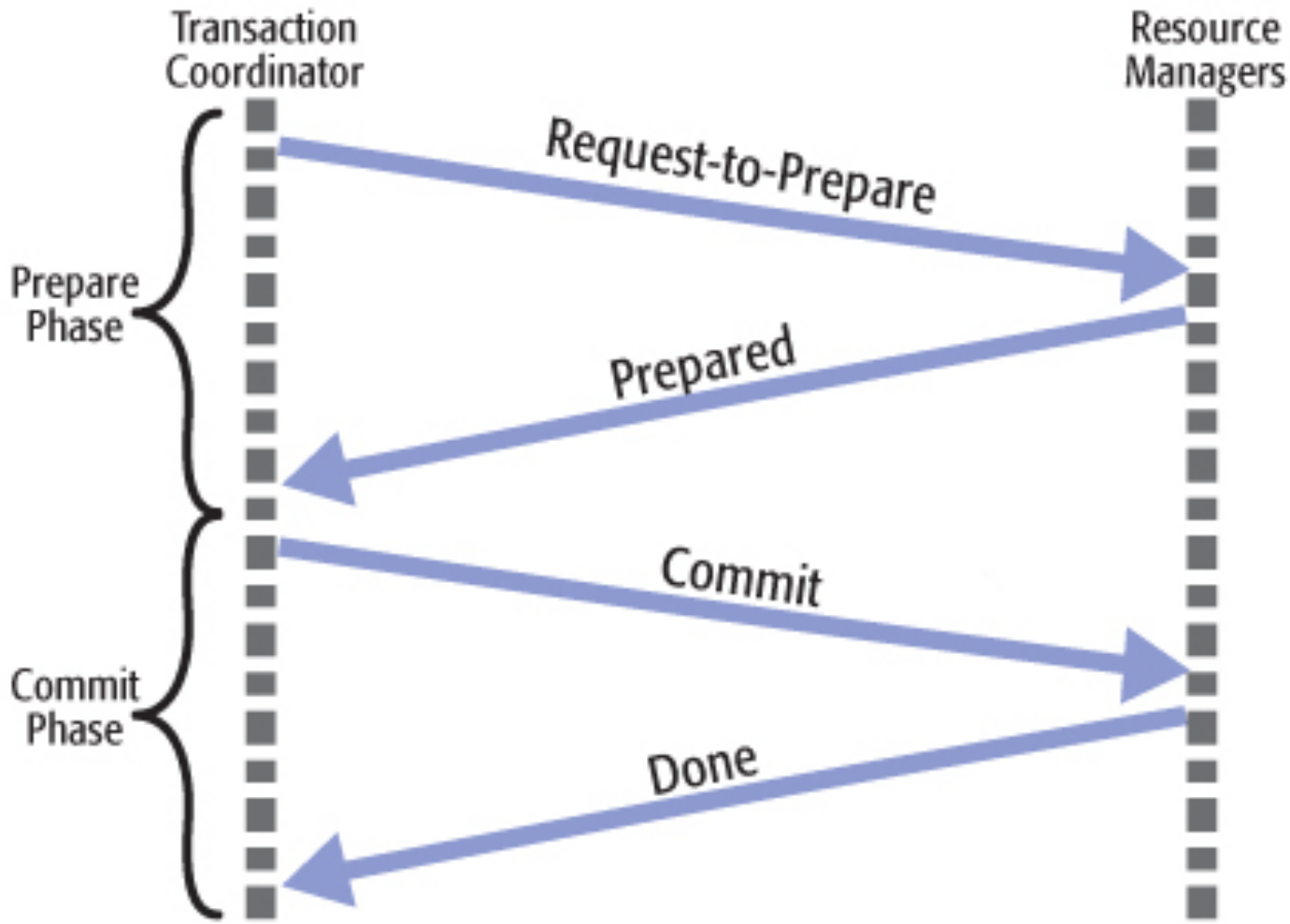


Single Commit

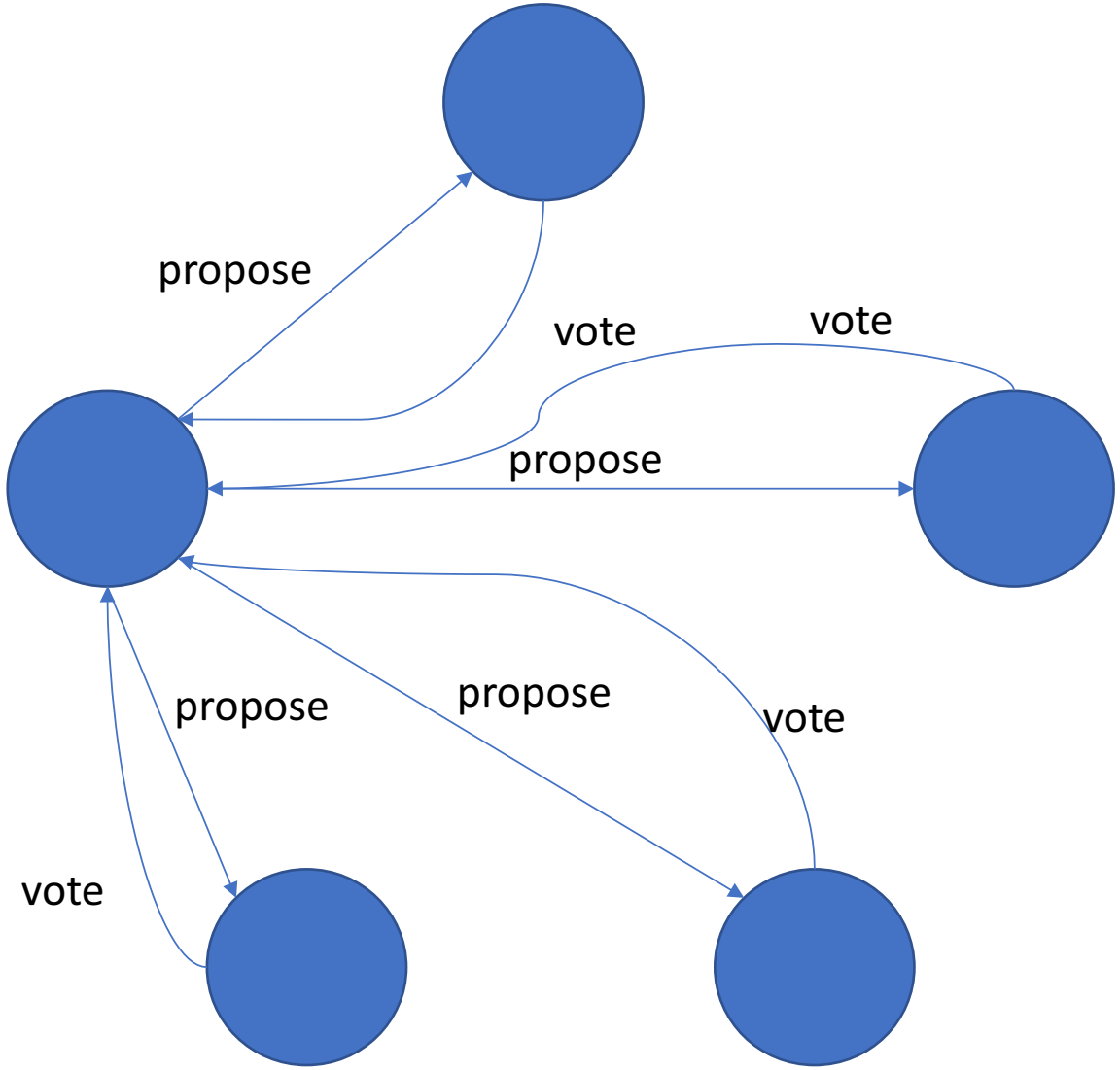


Two Phase Commit

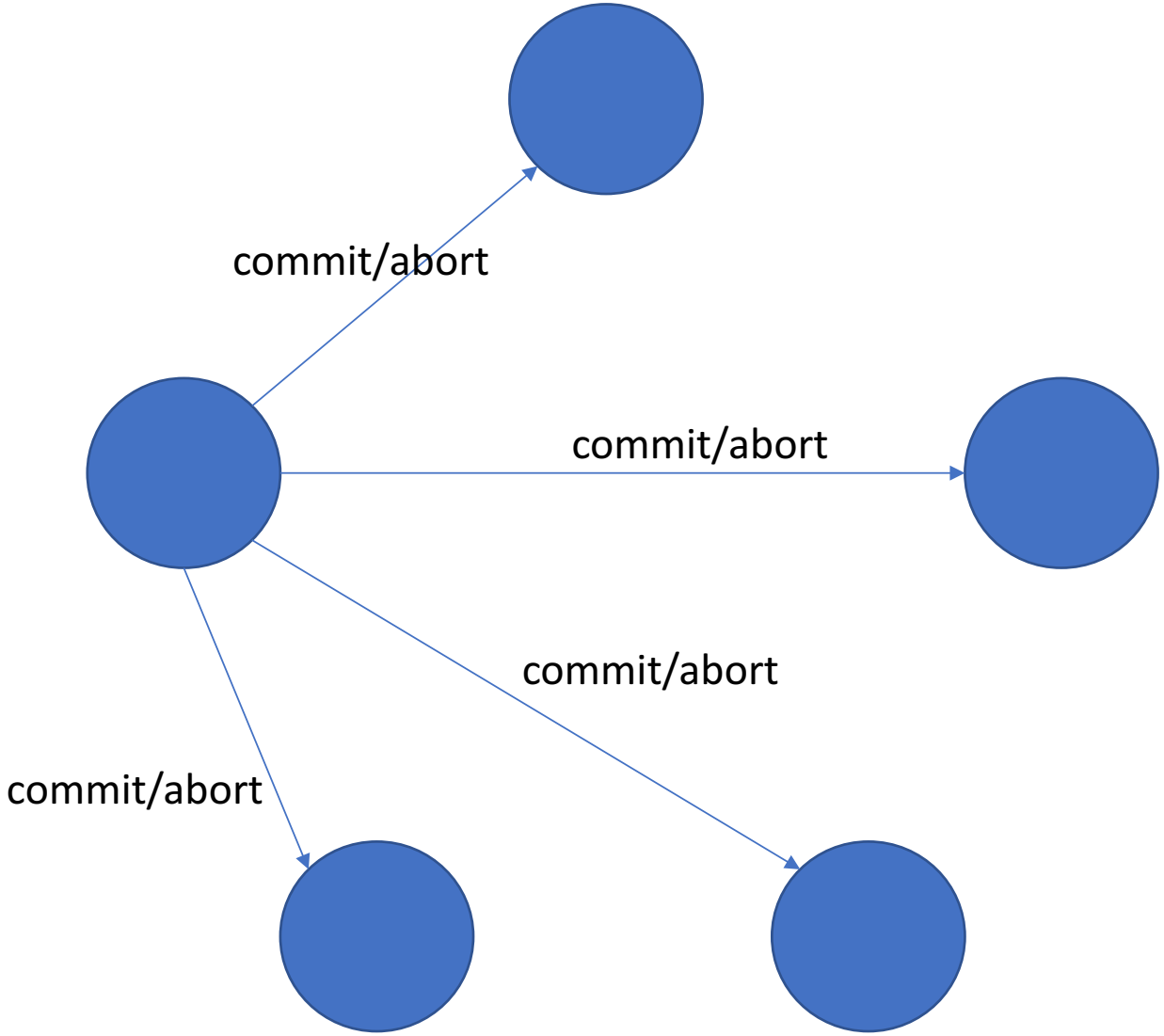
- The first *proposal* phase involves proposing a value to every participant in the system and gathering responses.
- The second *commit-or-abort* phase communicates the result of the vote to the participants and tells them either to go ahead and decide or abort the protocol.


Transaction Coordinator
Resource Managers
Request-to-Prepare
Prepared
Commit
Done
Prepare Phase
Commit Phase

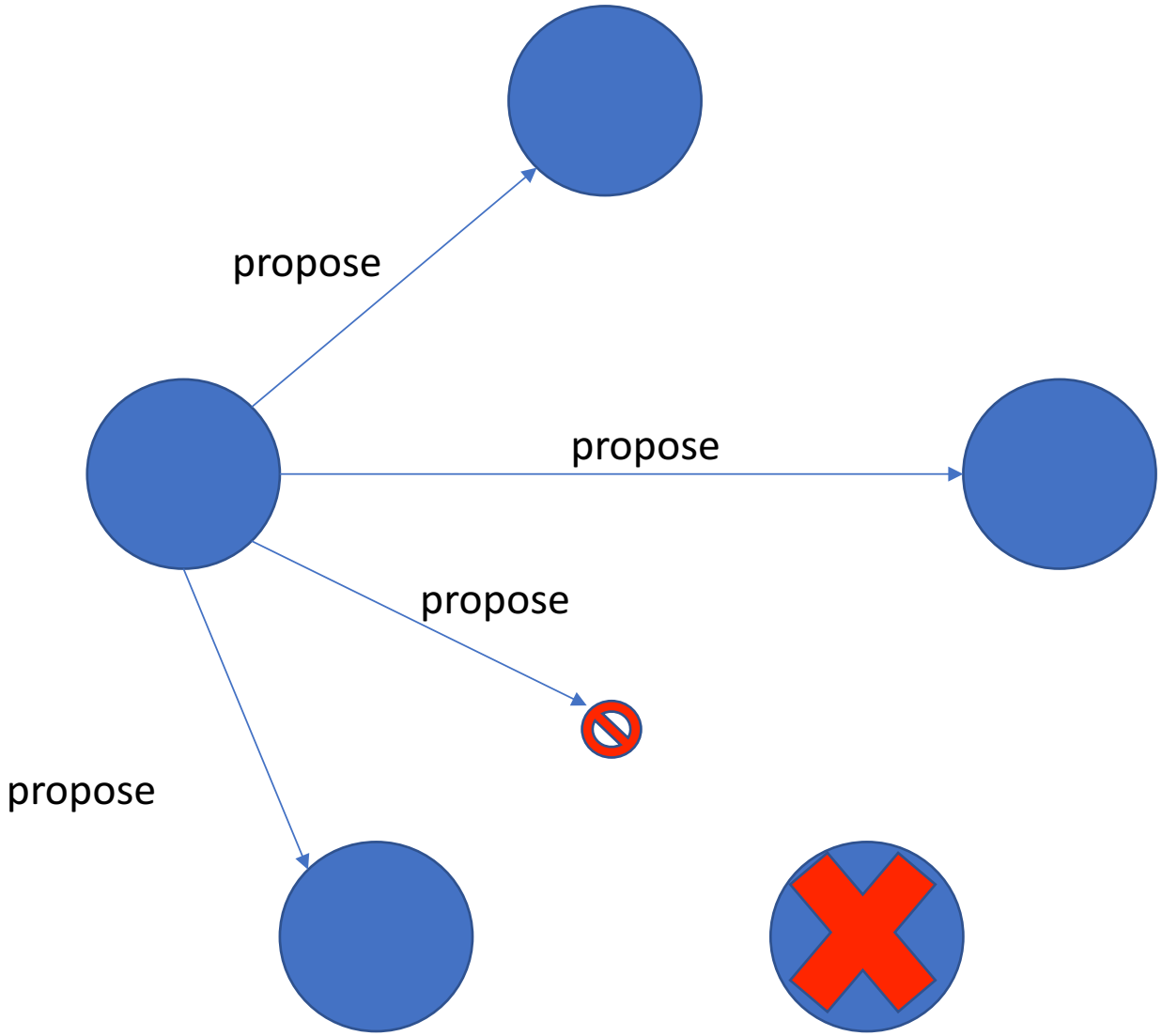
Two Phase Commit – Proposal Phase



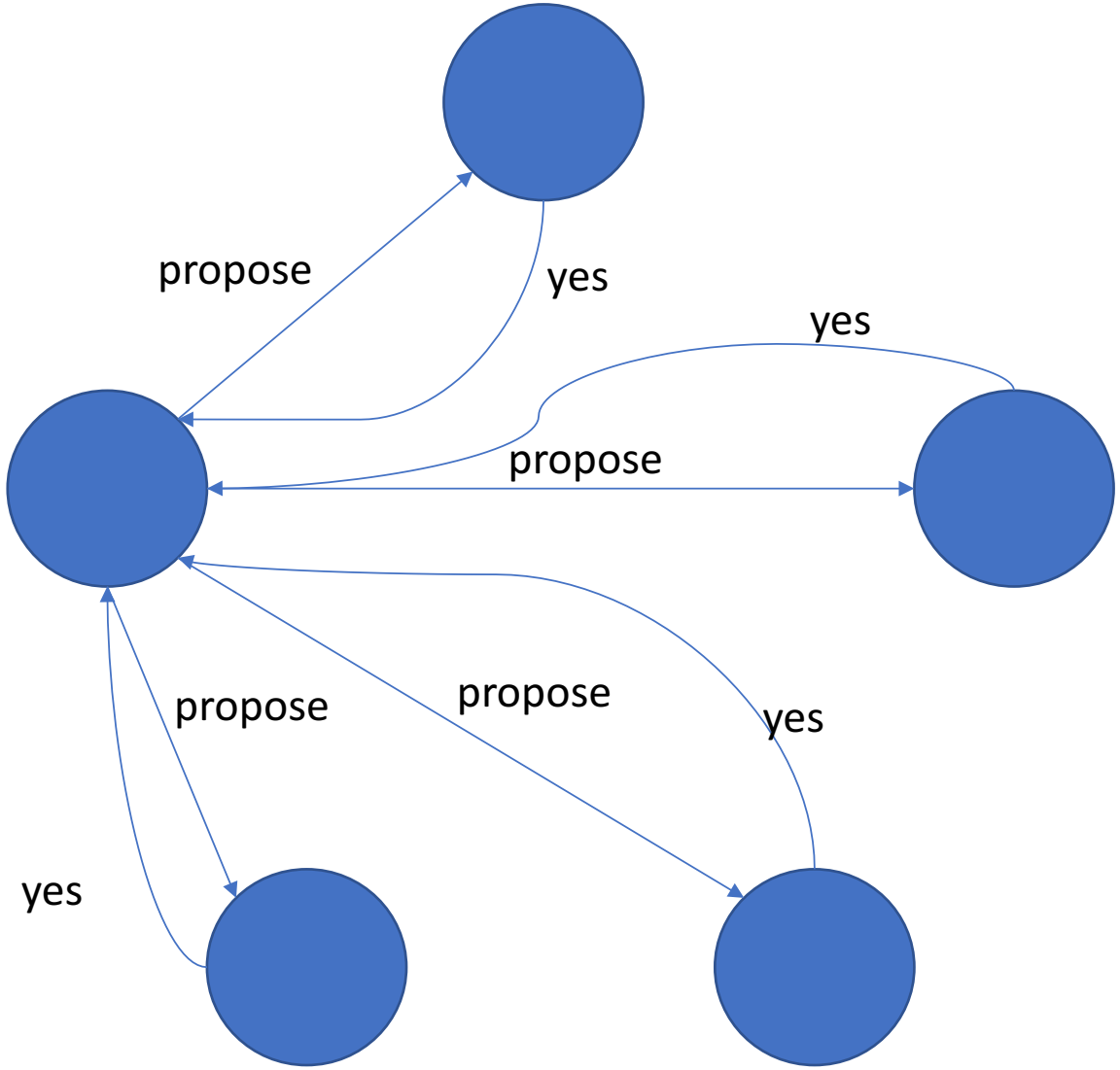
Two Phase Commit – Commit/Abort Phase



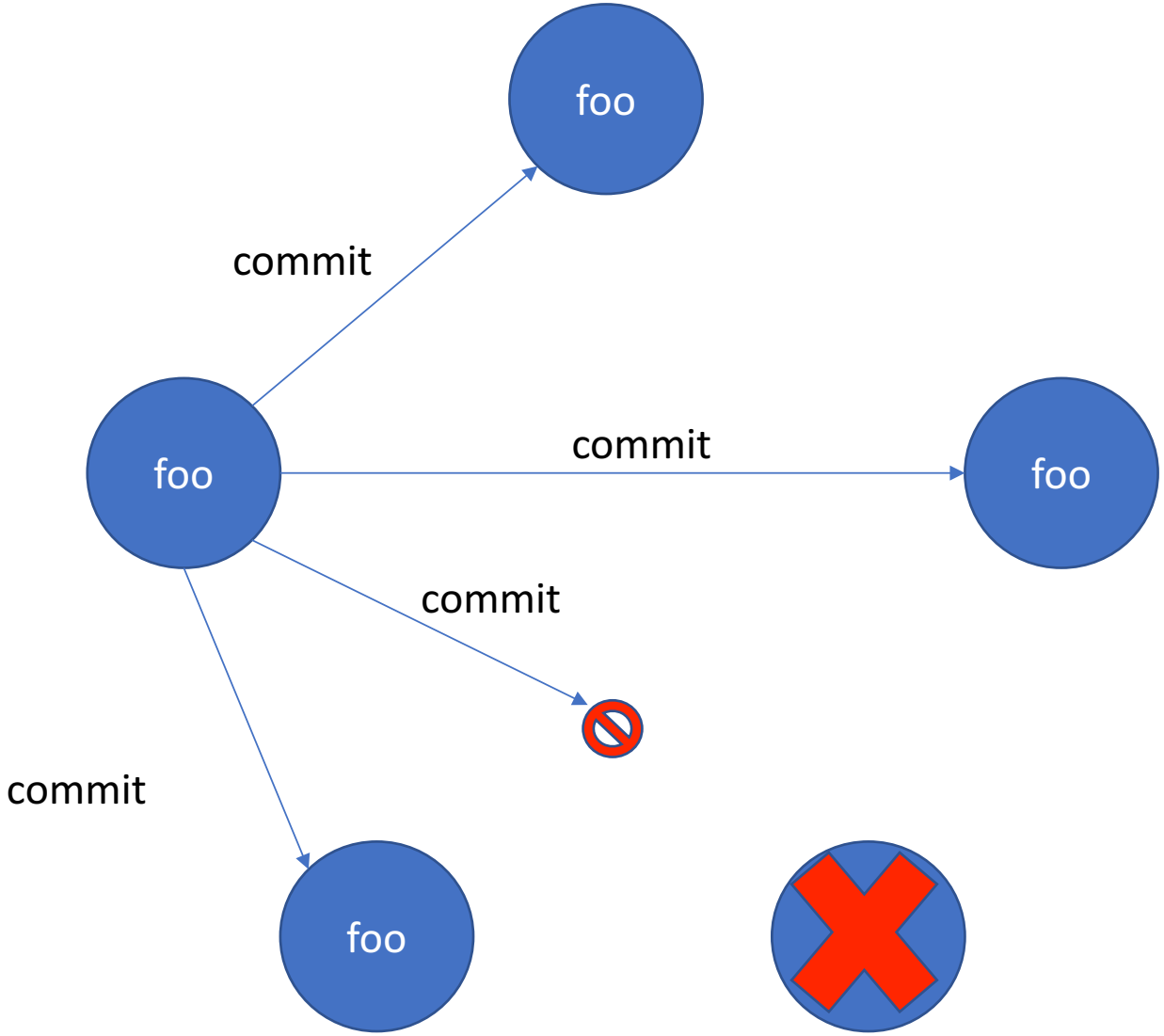
Two Phase Commit – Node Failure



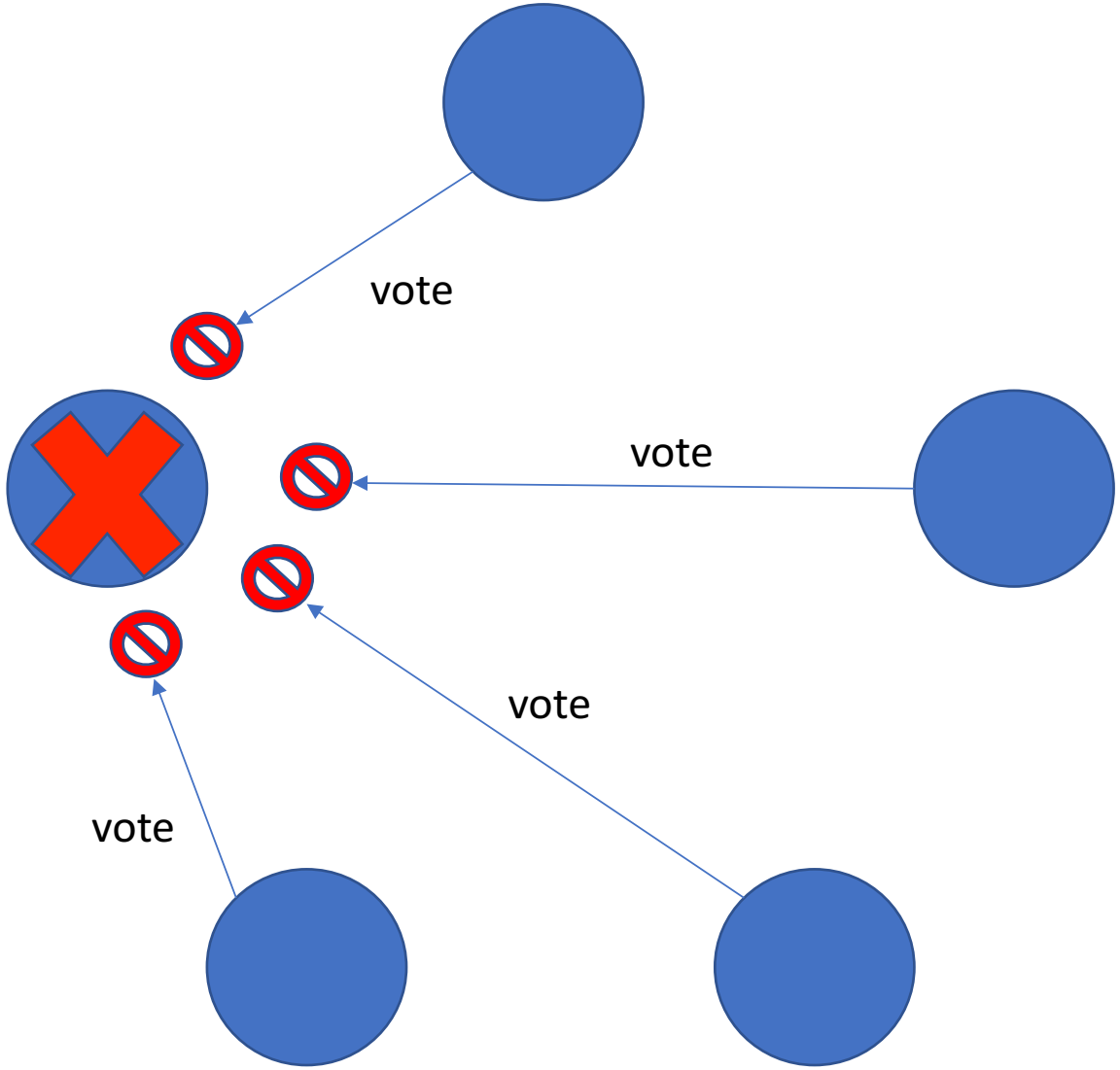
Two Phase Commit – Proposal Phase



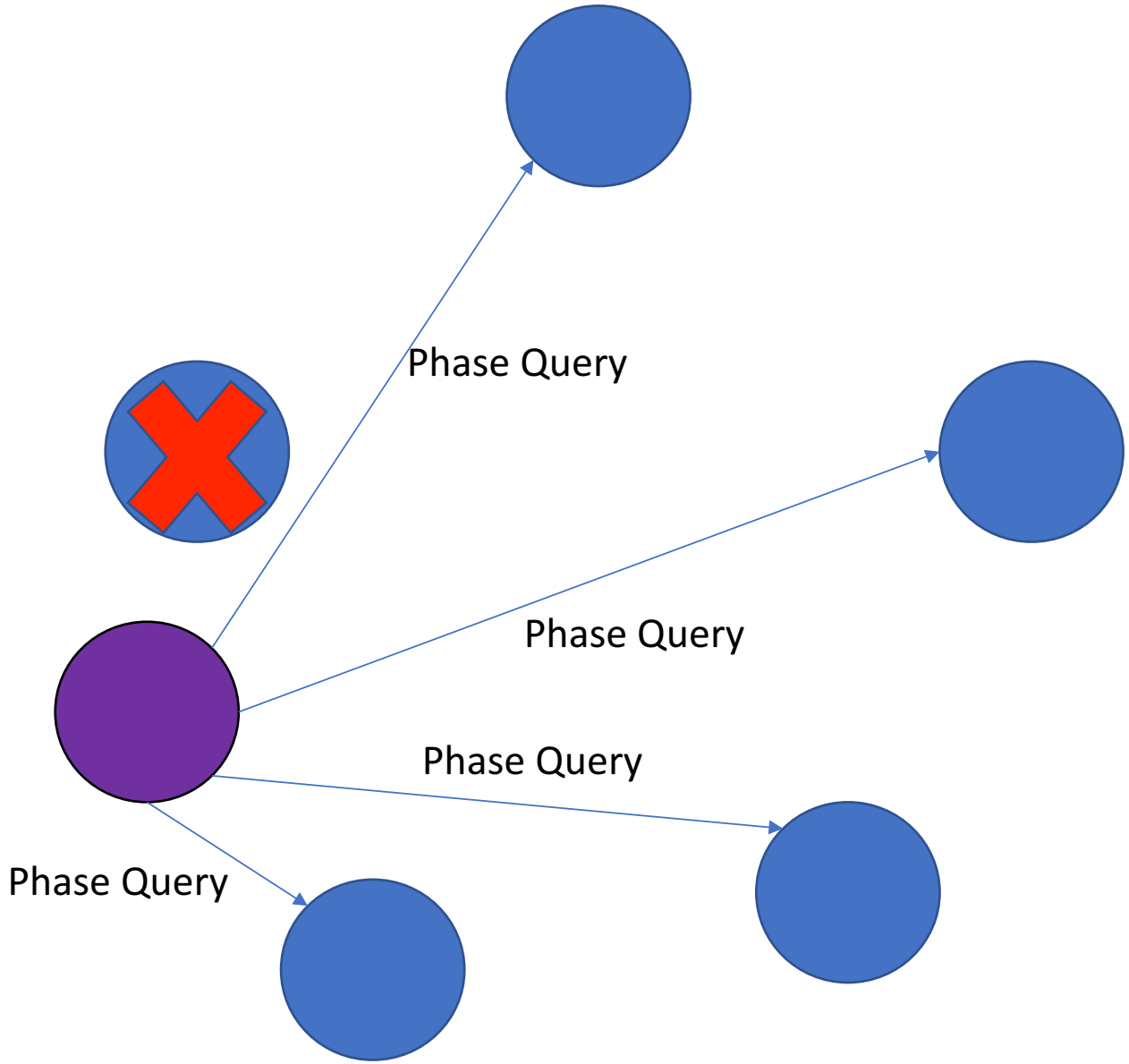
Two Phase Commit – Node Failure



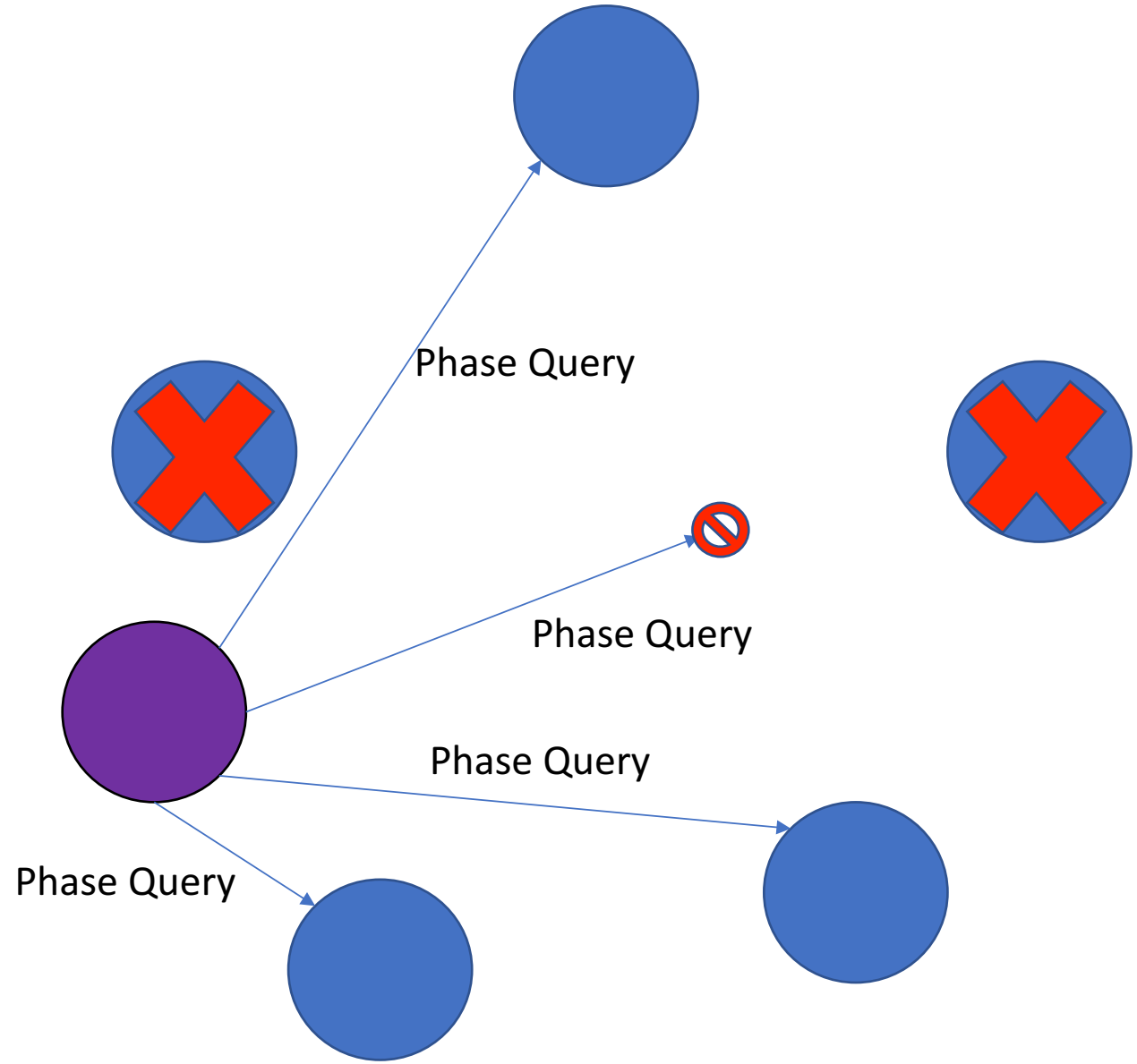
Two Phase Commit – Coordinator Failure



Two Phase Commit – Watchdog Takeover

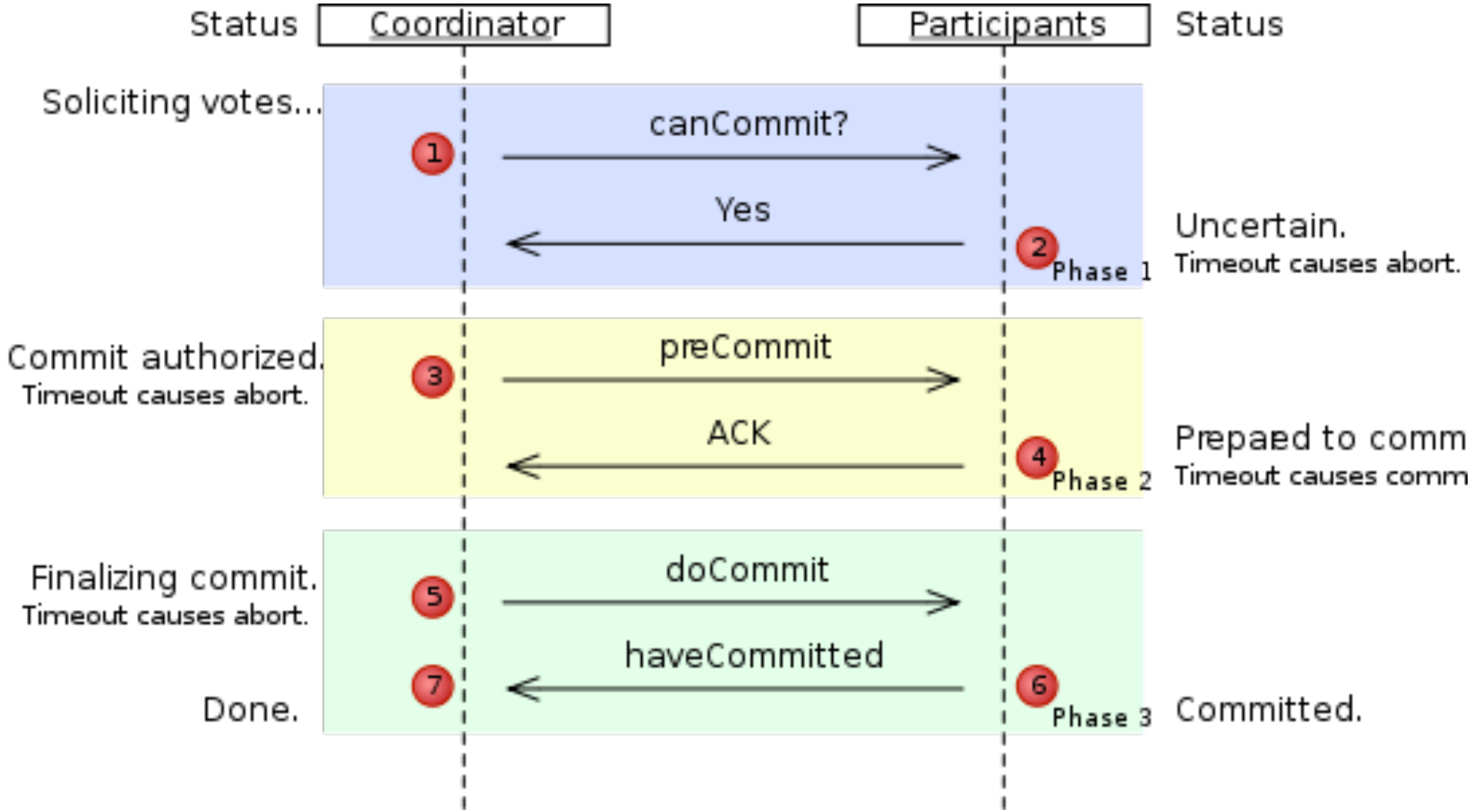


Two Phase Commit – Another node fails

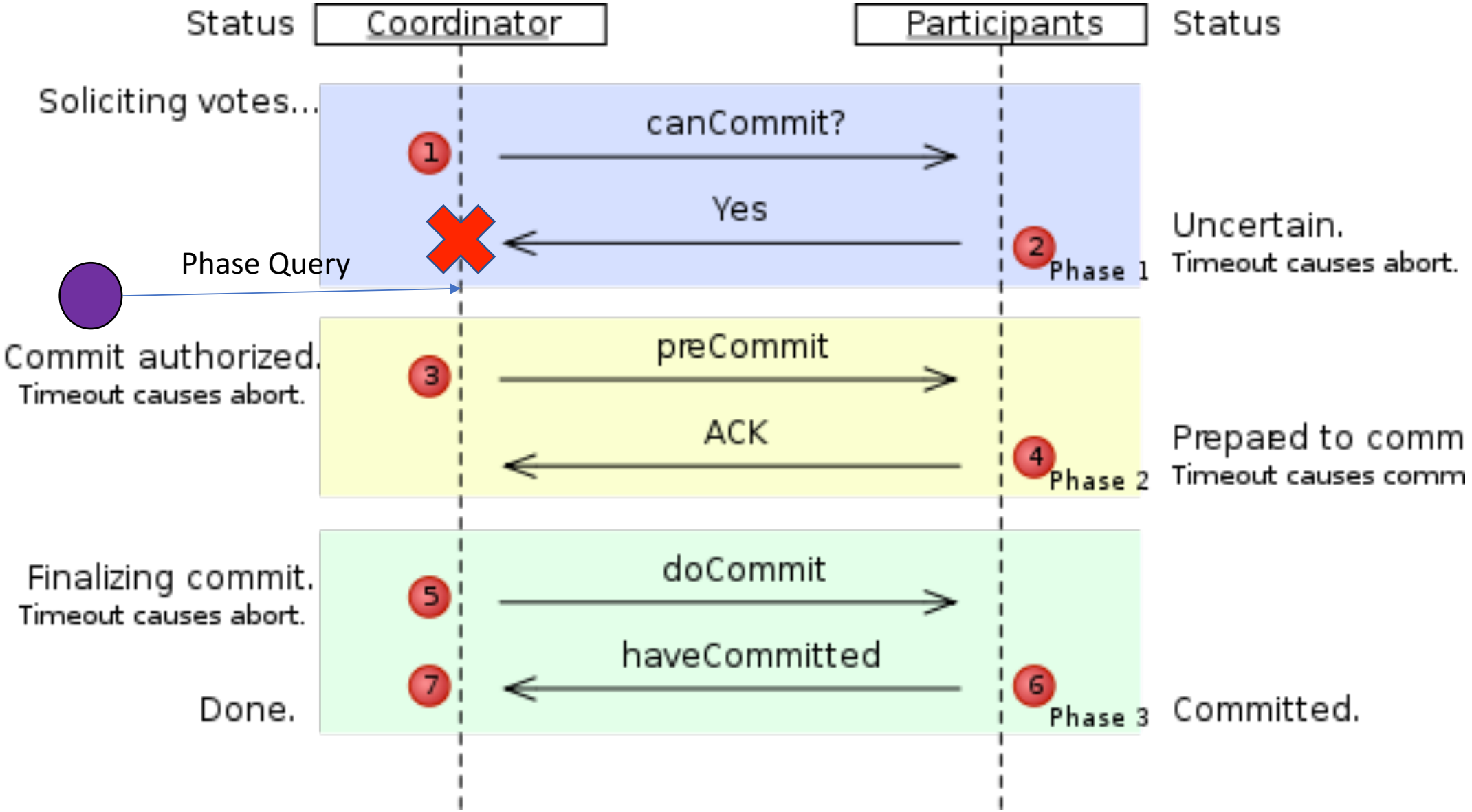


Three Phase Commit

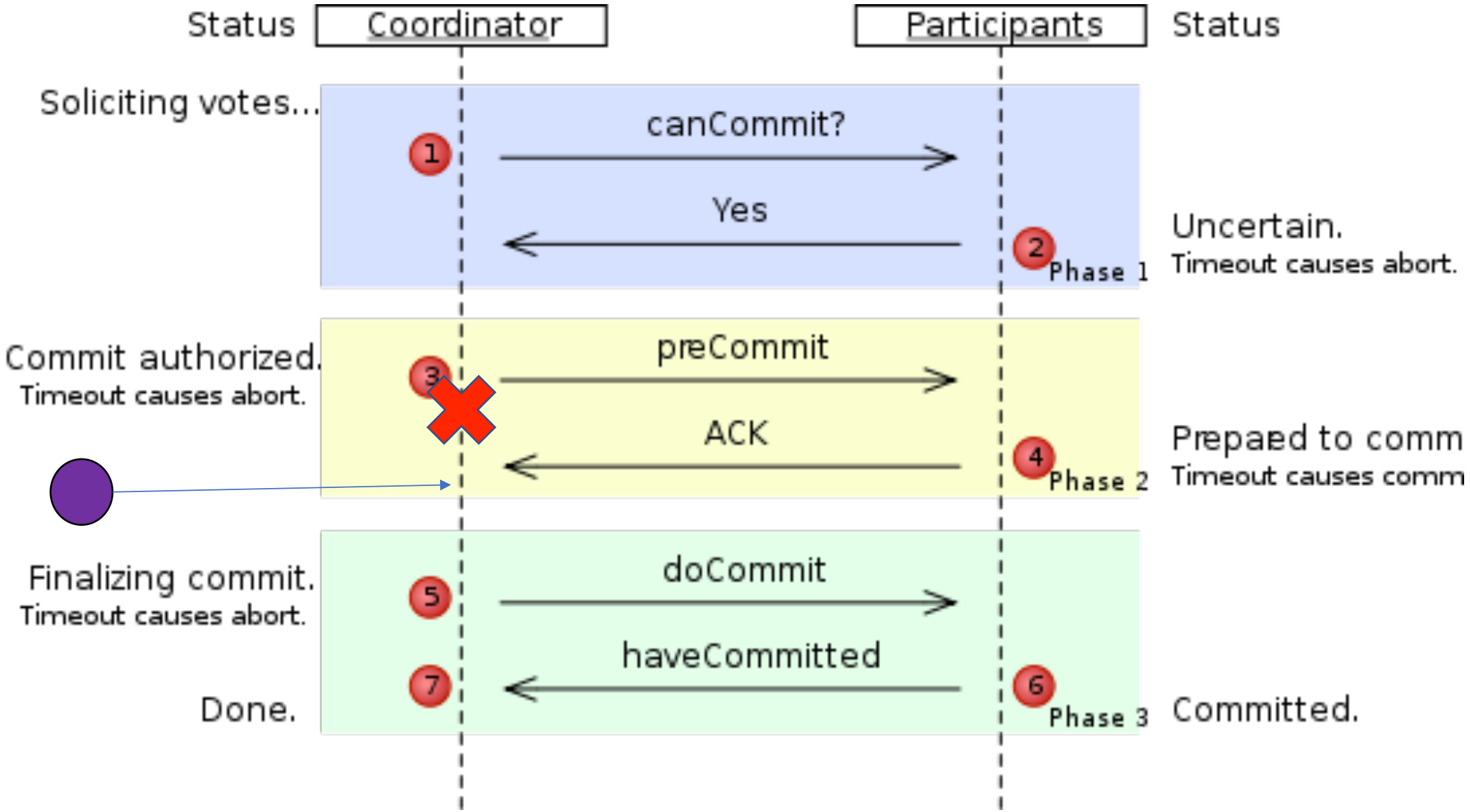
- The first *proposal* phase involves proposing a value to every participant in the system and gathering responses.
- The second *prepare to commit* phase communicates the result of the vote to the participants and tells take necessary locks, but crucially do not do any work that cannot be undone.
- The last *commit-or-abort* phase tells the participants to either go ahead and commitor abort the protocol.


Status
Coordinator
Participants
Status
Soliciting votes...
canCommit?
1
Yes
2
Phase 1
Uncertain.
Timeout causes abort.
Commit authorized.
Timeout causes abort.
preCommit
3
ACK
4
Phase 2
Prepaed to comm
Timeout causes comm
Finalizing commit.
Timeout causes abort.
doCommit
5
haveCommitted
7
Done.
6
Phase 3
Committed.

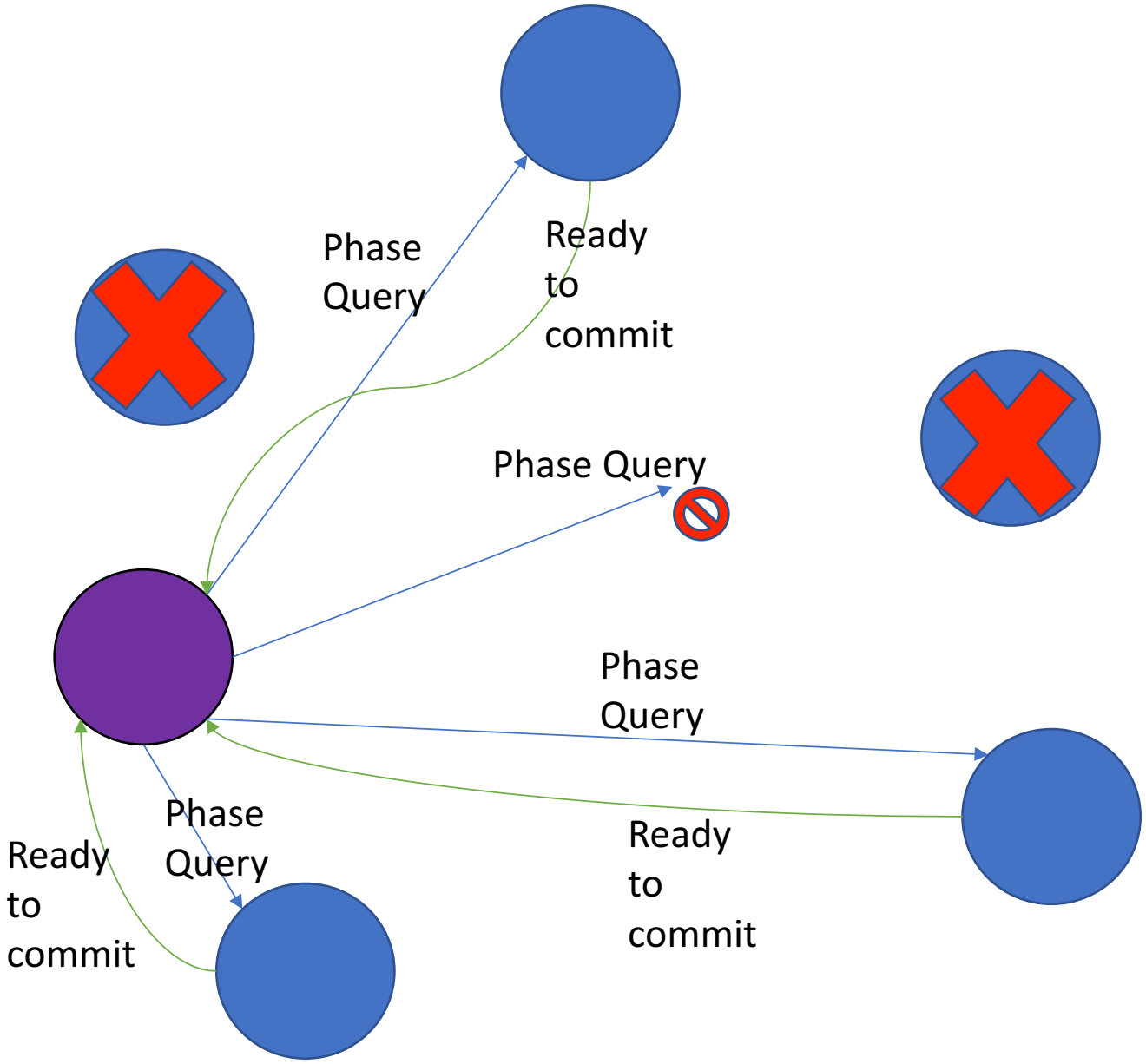
3PC – Coordinator Failure – After Phase 1



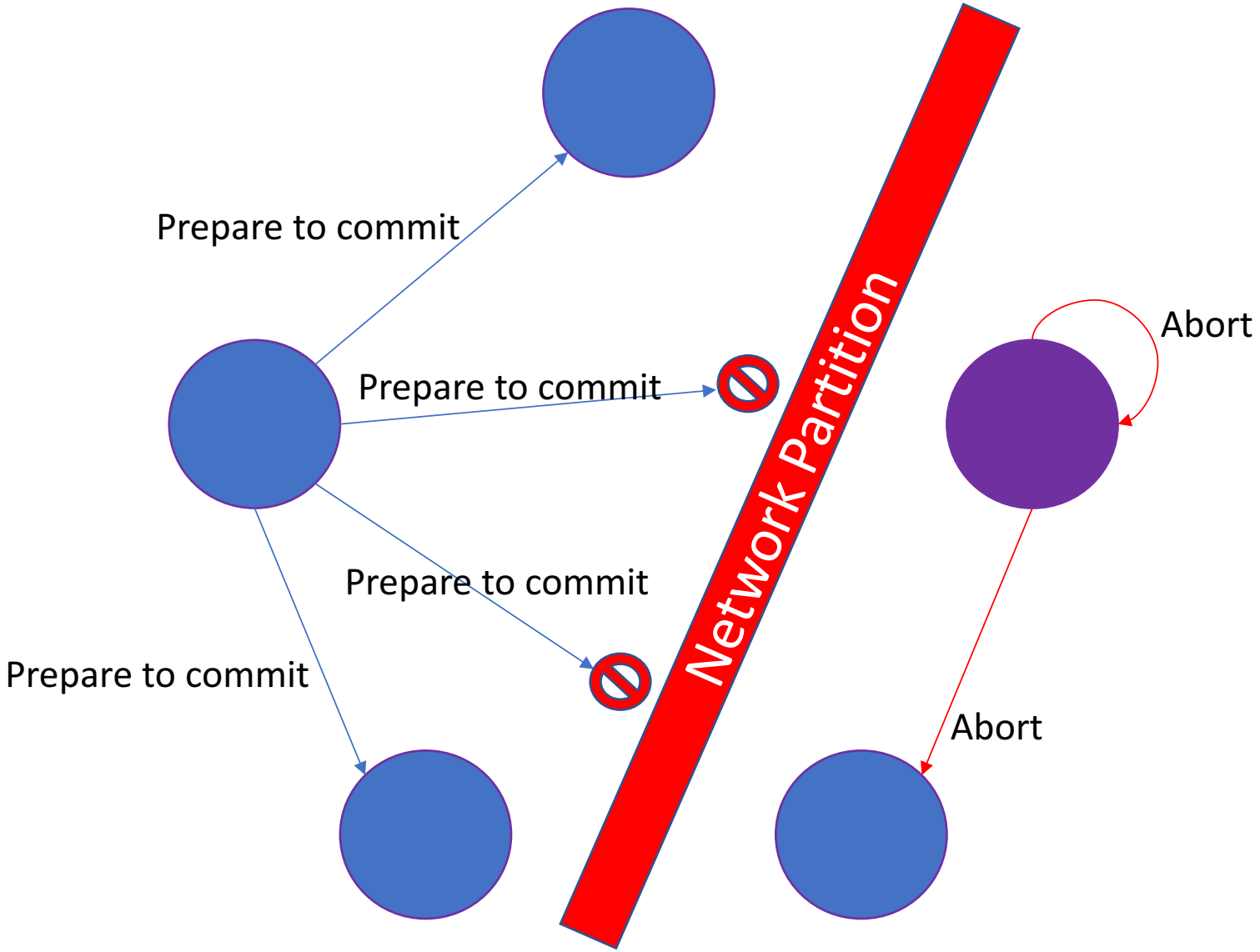
3PC – Coordinator Failure – After Phase 2



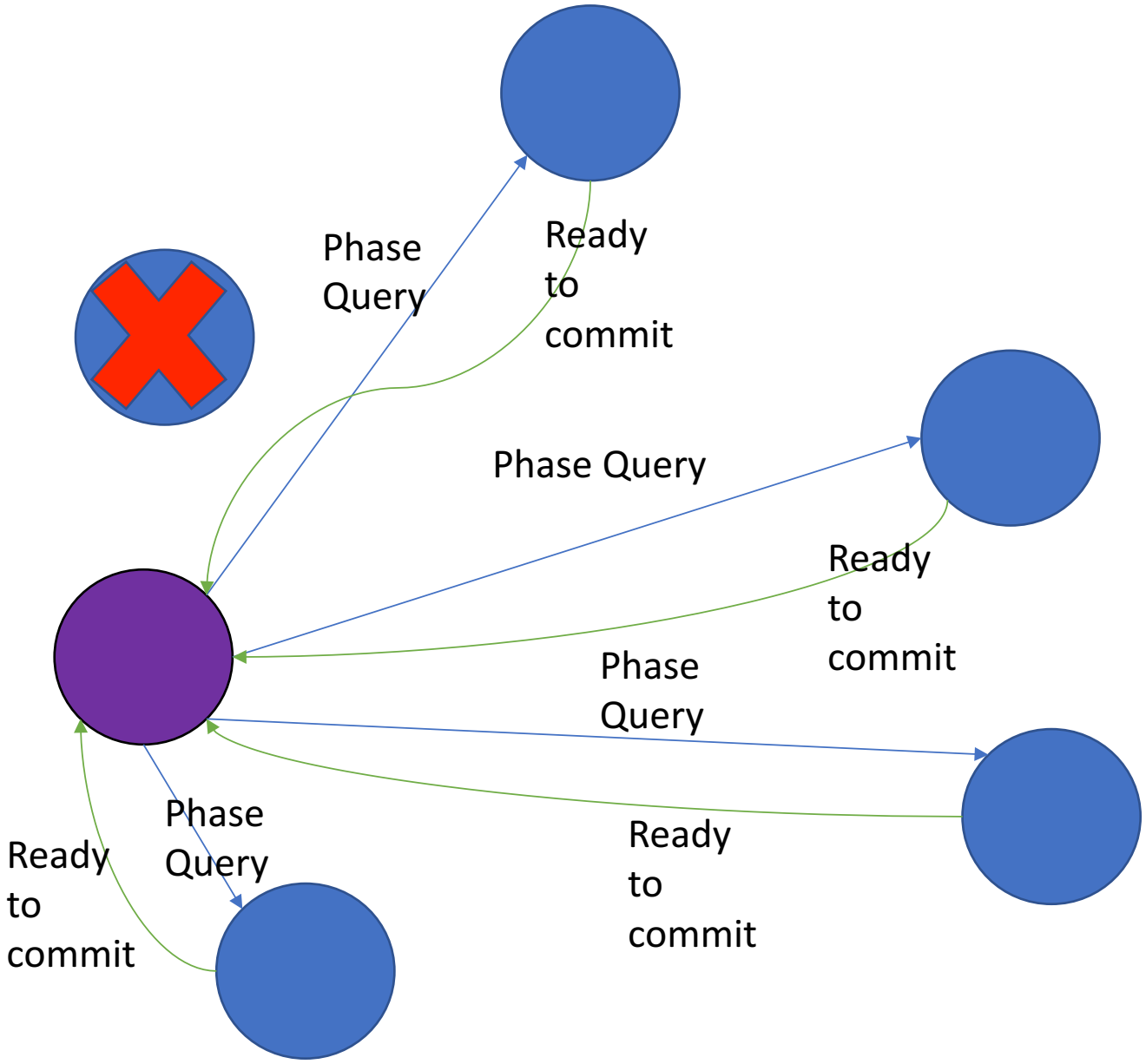
3PC – Coordinator and Node Failure – After Phase 2



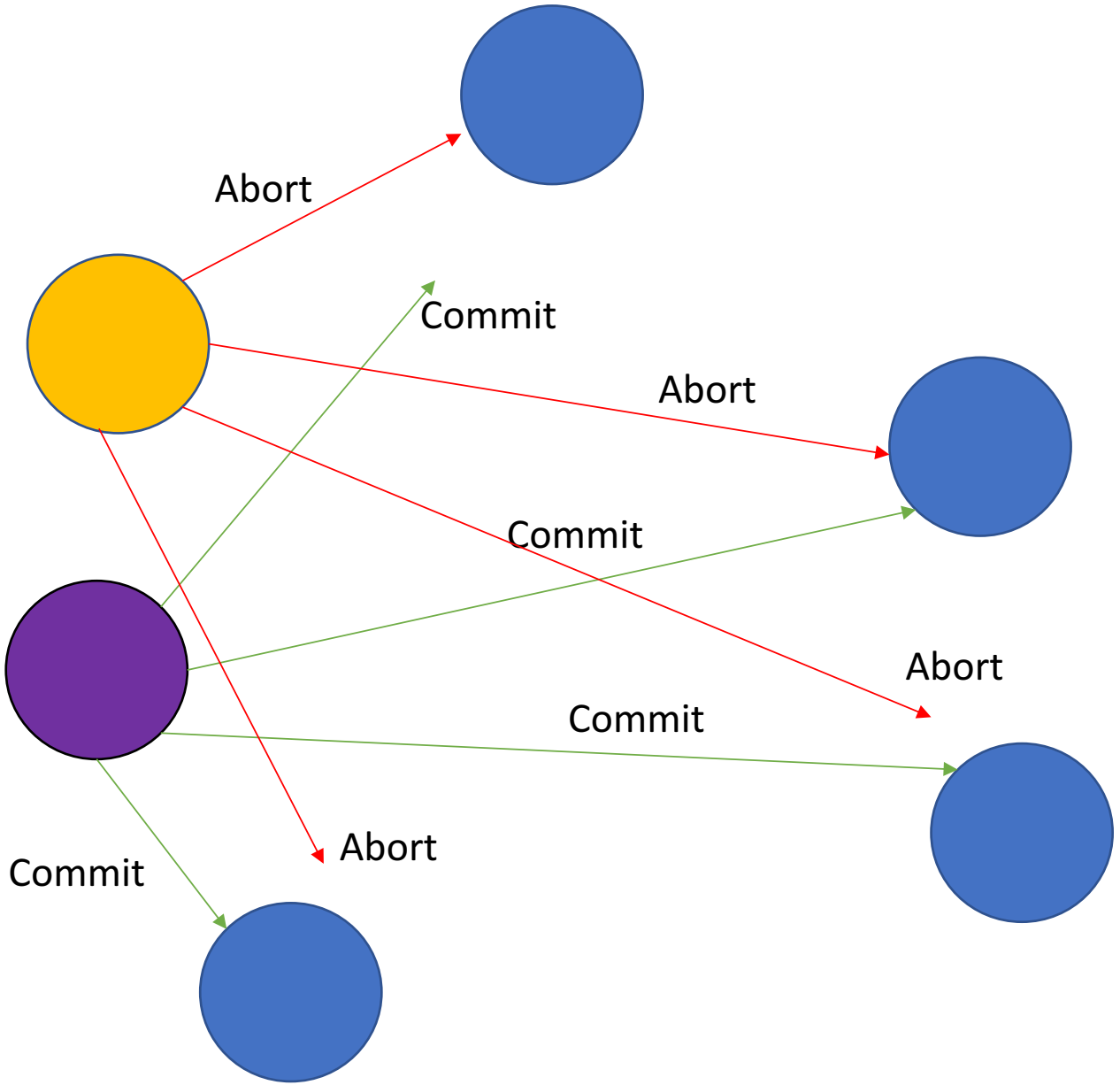
Network Partition



3PC – Coordinator Fail Recover



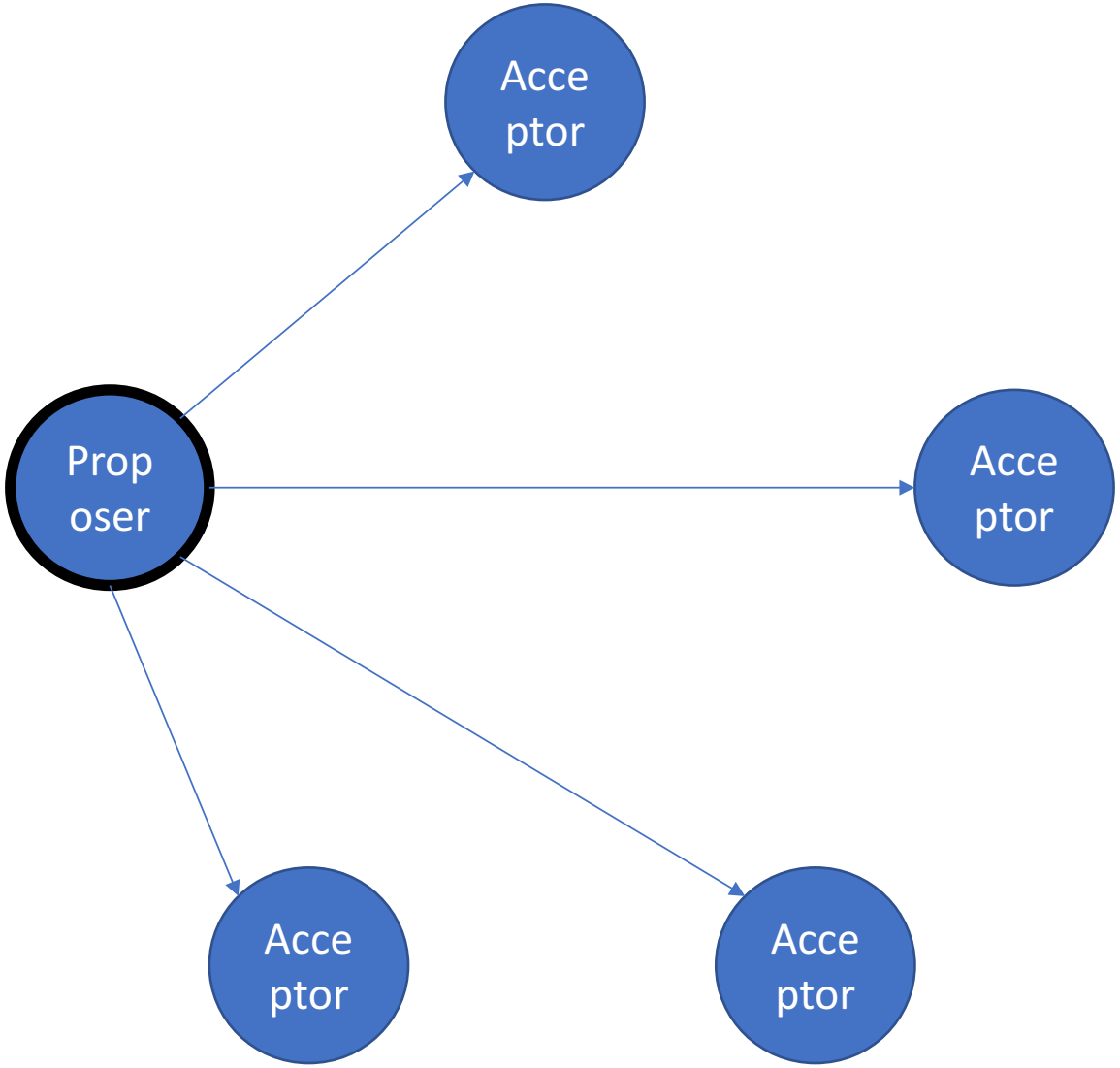
3PC – Coordinator Fail Recover



Paxos

- Paxos is a little similar to 2PC in the way it operates, but introduces two additional concepts
  - *Quorum*
  - *Sequencing*

Paxos

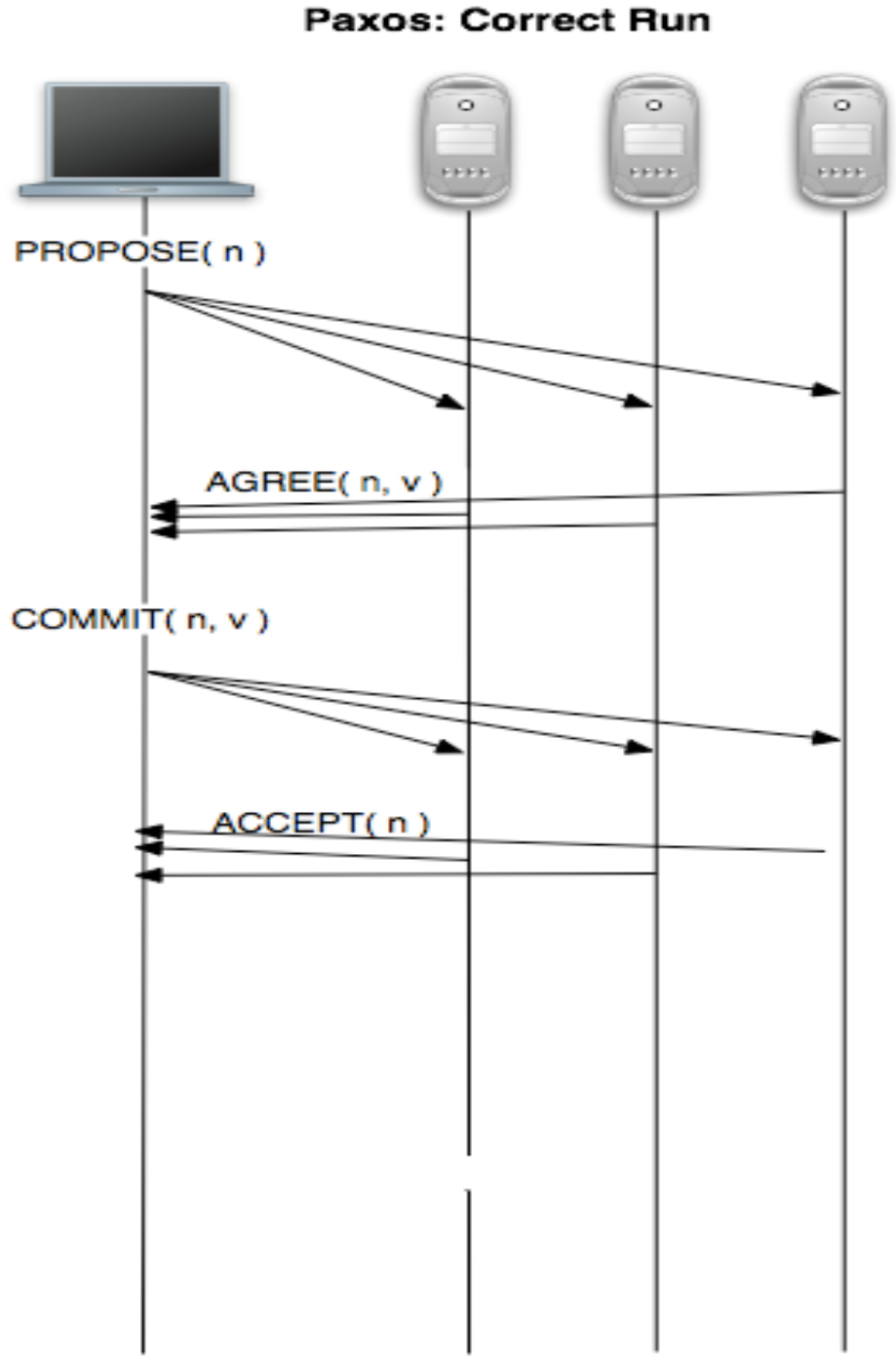


PROPOSERS:

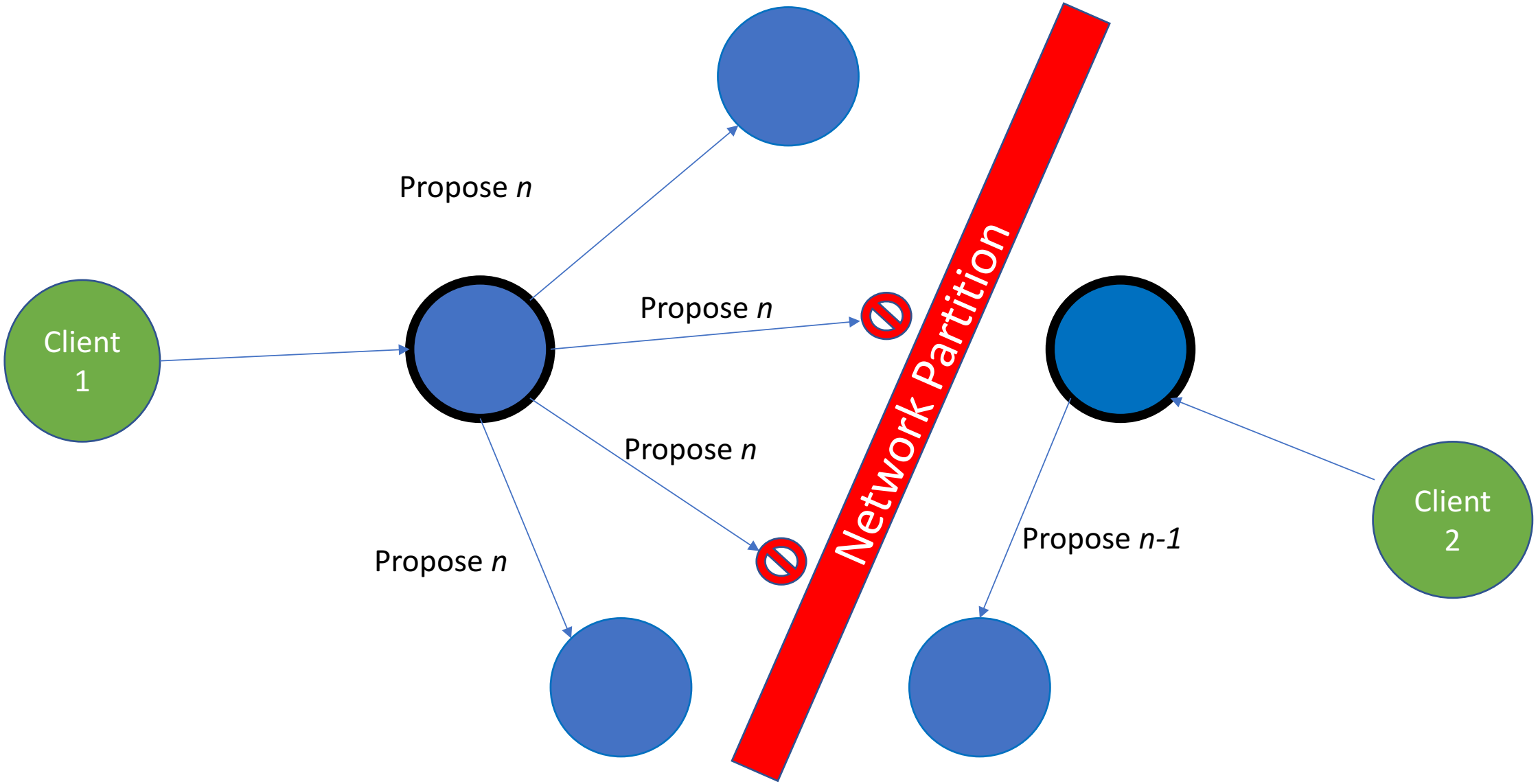
1. Submit a proposal numbered *n* to a majority of acceptors. Wait for a majority of acceptors to reply.
2. If the majority reply ‘agree’, they will also send back the value of any proposals they have already accepted. Pick one of these values, and send a ‘commit’ message with the proposal number and the value. If no values have already been accepted, use your own. If instead a majority reply ‘reject’, or fail to reply, abandon the proposal and start again.
3. If a majority reply to your commit request with an ‘accepted’ message, consider the protocol terminated. Otherwise, abandon the proposal and start again.

ACCEPTORS:

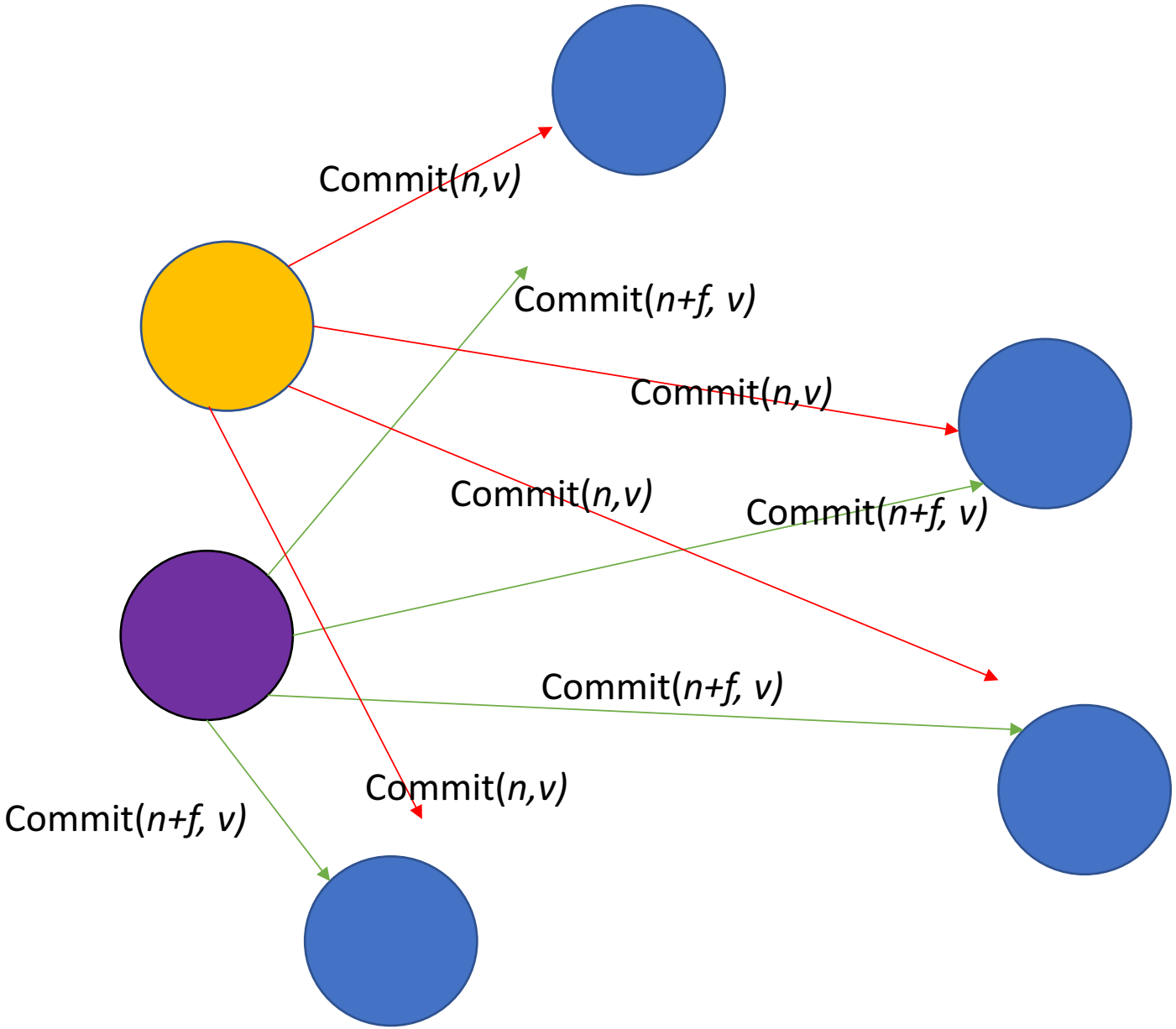
- Once a proposal is received, compare its number to the highest numbered proposal you have already agreed to. If the new proposal is higher, reply ‘agree’ with the value of any proposals you have already accepted. If it is lower, reply ‘reject’, along with the sequence number of the highest proposal.
- When a ‘commit’ message is received, accept it if a) the value is the same as any previously accepted proposal and b) its sequence number is the highest proposal number you have agreed to. Otherwise, reject it.


Paxos: Correct Run
PROPOSE(n)
AGREE(n, v)
COMMIT(n, v)
ACCEPT(n)

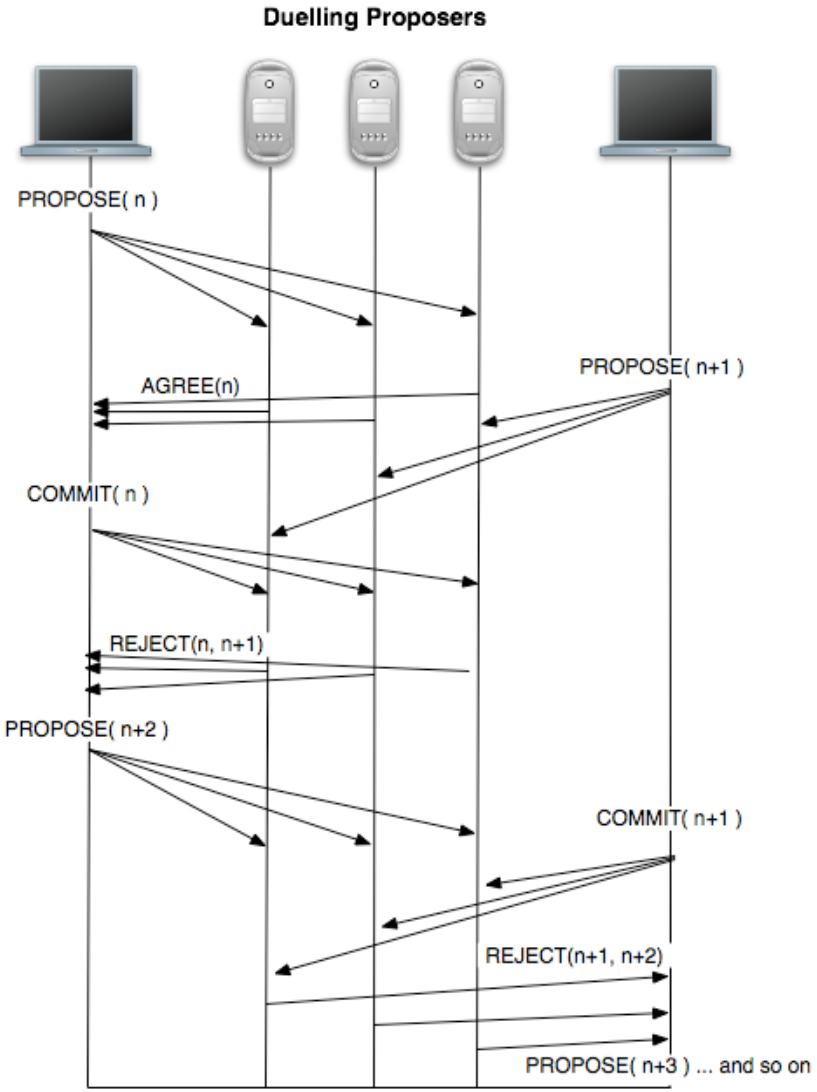
Network Partition



Paxos Fail Recover



Problems With Paxos?



Questions?

References

- The part-time parliament, Leslie Lamport
- Paxos Made Simple, Leslie Lamport
- Paxos Made Live, T Chandra et al
- In Search of an Understandable Consensus Algorithm, Diego Ongaro et al.,
- http://thesecretlivesofdata.com/