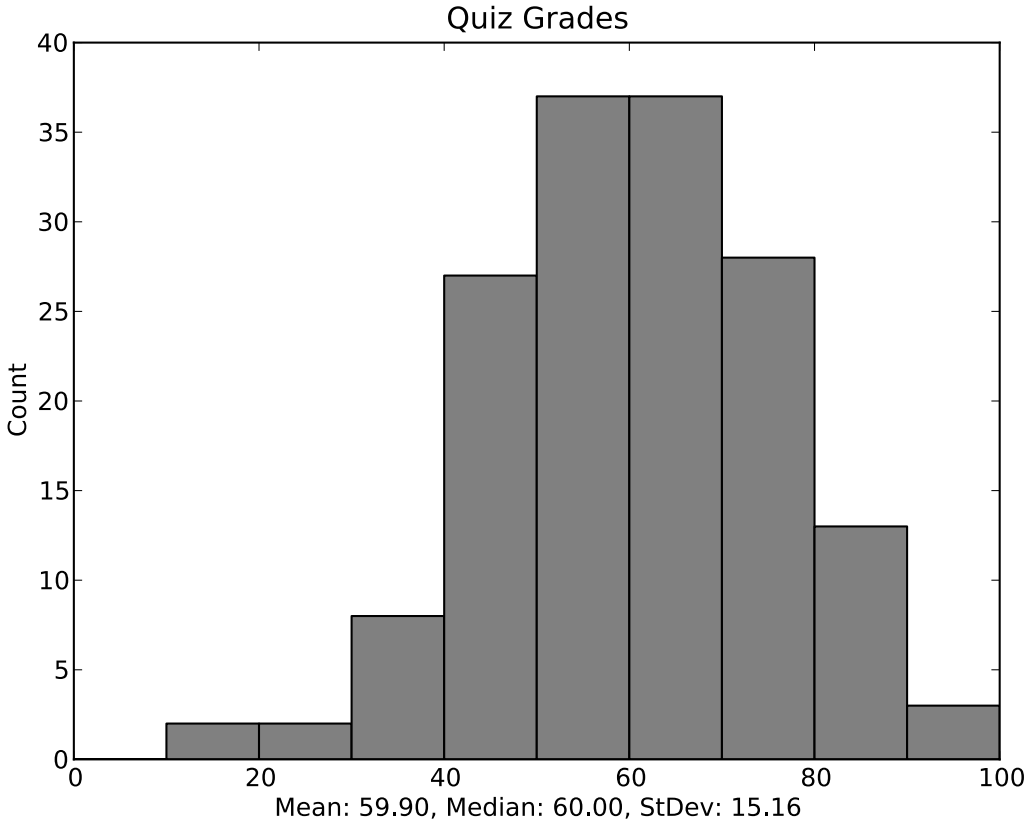

Quiz Grades
Count
0
5
10
15
20
25
30
35
40
0
20
40
60
80
100
Mean: 59.90, Median: 60.00, StDev: 15.16

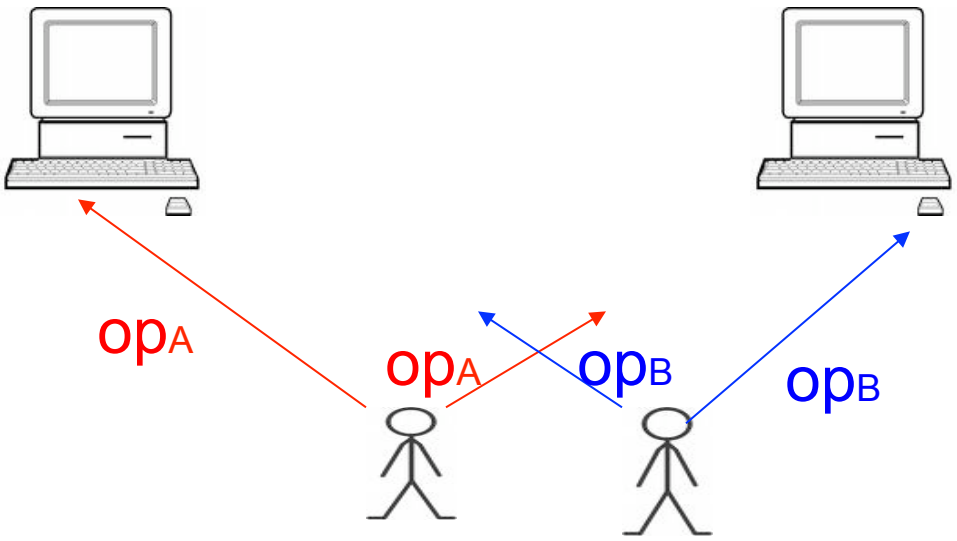
6.033: Replication

Frans Kaashoek
With slides by Jinyang Li and Robert Morris

Fault tolerance ➔replication

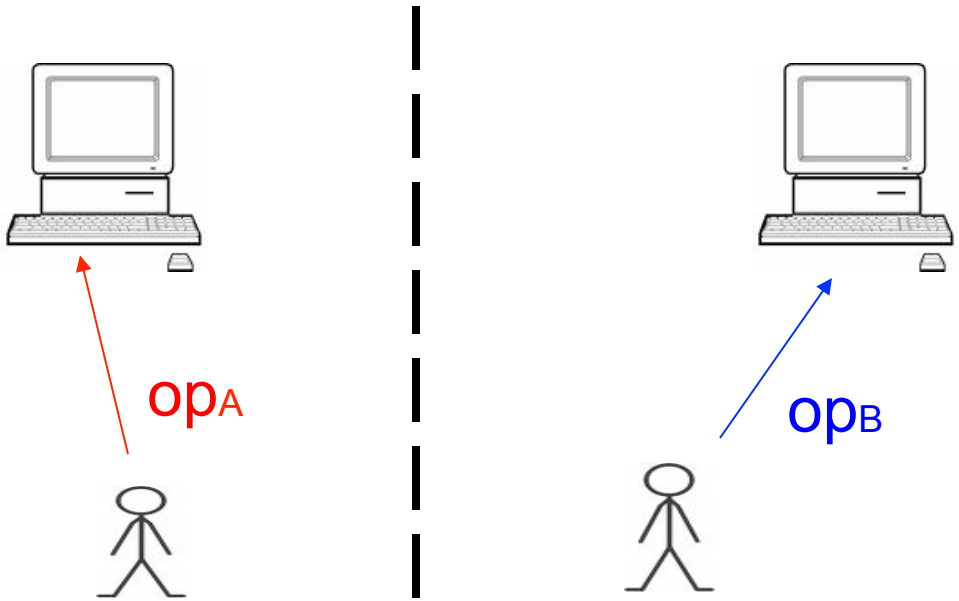
- How to recover from a single failure?
  - Wait for reboot
    - Data is durable, but service is unavailable temporarily
- Can we continue despite failure?
  - Useful for DNS, file server, master in two phase-commit, etc.

Replication



- Send requests to both
- If one is down, send to the other one

Network partition



- Client cannot know if a server is down
- Different clients see different results
- Same client may see state switching back&forth

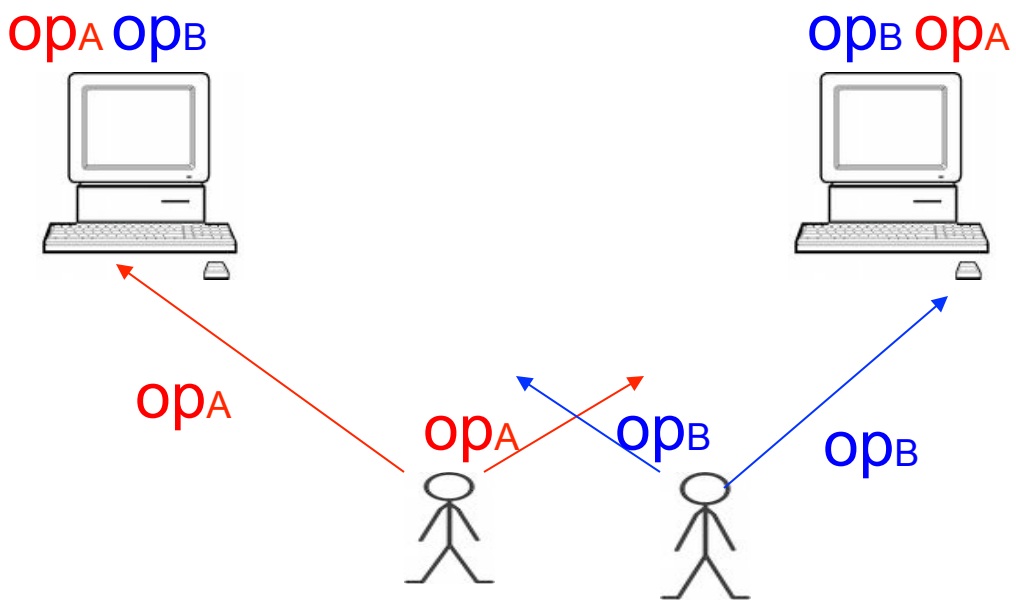
Consistency

- Tolerate inconsistency
  - DNS
- Reconcile later, but may have conflicts
  - Unison
- Single-copy consistency
  - Looks like one copy of the data
  - Each request sees result of previous requests

Replicated state machine (RSM)

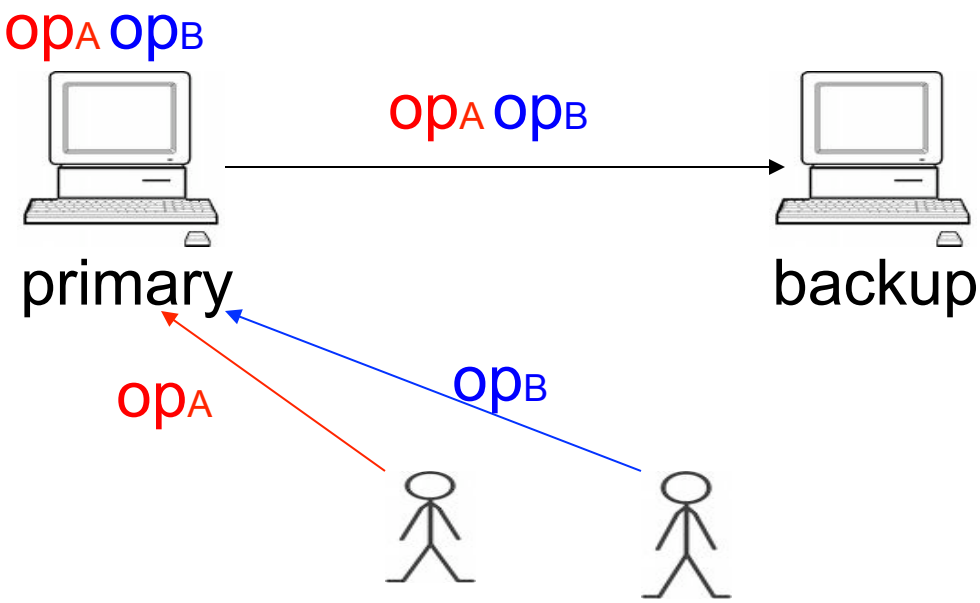
- RSM is a general replication method
- RSM Rules:
  - All replicas start in the same initial state
  - Every replica apply operations in the same order
  - All operations must be deterministic
- All replicas end up in the same state

RSM



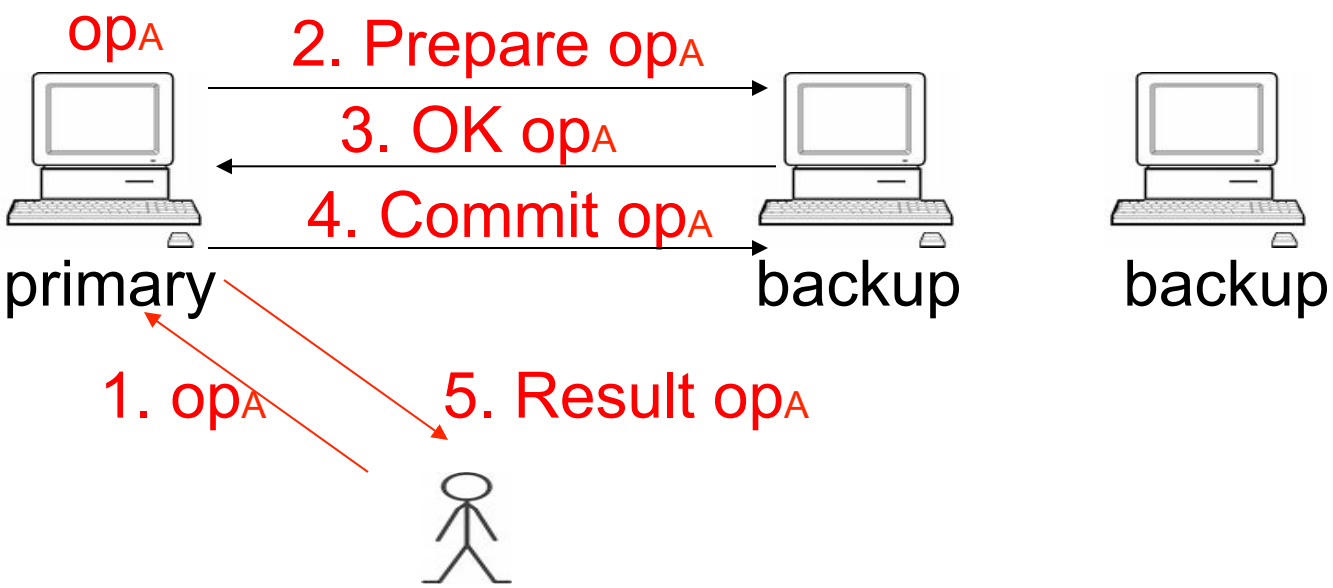
- How to maintain a single order in the face of concurrent client requests?

RSM using primary/backup



- Primary/backup: ensure a single order of ops:
  - Primary orders operations
  - Backups execute operations in order

When does primary respond?



- After backups have committed to op

Challenge: failures

- Primary failure
  - Backups ping primary periodically
  - No response: one backup becomes primary
- What if the network between primary/backup fails?
  - Primary is still running
  - Backup becomes a new primary
  - Two primaries at the same time!
- Maybe partition with majority continues
  - Better agree who is that majority and what last op was

Paxos: fault tolerant agreement

- Paxos lets all nodes agree on the same value despite node failures, network failures and delays
- Extremely useful:
  - e.g. Nodes agree that X is the primary
  - e.g. Nodes agree that Y is the last operation executed
- “Paxos Made Simple” by L. Lamport

Paxos Properties

- Correctness (safety):
  - All nodes agree on the same value
  - The agreed value X has been proposed by some node
- Fault-tolerance:
  - If less than N/2 nodes fail, the rest nodes should reach agreement *eventually w.h.p*
  - Liveness is not *guaranteed*
- Assume: fail-stop

Paxos: general approach

- One (or more) node decides to be the leader
- Leader proposes a value and solicits acceptance from majority
- Leader announces result or try again

Challenges

- What if >1 nodes become leaders simultaneously?
  - What if there is a network partition
- What if a leader crashes after deciding but before announcing results?
  - New leader shouldn't choose a different value ….

Paxos setup

- Each node runs as a *proposer*, *acceptor* and *learner*
- Proposer (leader) proposes a value and solicit acceptence from acceptors
- Leader announces the chosen value to learners

Strawman

- Designate a single node X as acceptor (e.g. one with smallest id)
  - Each *proposer* sends its value to X
  - X decides on one of the values
  - X announces its decision to all *learners*
- Problem?
  - Failure of the single acceptor halts decision
  - Need multiple acceptors!

Strawman 2: multiple acceptors

- Each proposer (leader) propose to all acceptors
- Each acceptor accepts the first proposal it receives and rejects the rest
- If the leader receives positive replies from a majority of acceptors, it chooses its own value
  - There is at most 1 majority, hence only a single value is chosen
- Leader sends chosen value to all learners
- Problem:
  - What if multiple leaders propose simultaneously and there is no majority accepting?

Paxos solution: two phases

- Prepare: agree to an ordered proposal #
- Accept: agree to value for proposal #
  - Each acceptor may accept multiple proposals
- Why do proposals have number #?
  - May need multiple rounds (e.g., leader fails)
  - Later rounds should supersede earlier rounds, but if a proposal with value v is *chosen*, all higher proposals have value v

Paxos state

- Acceptor maintains across reboots:
  - n_a, v_a: highest accept # and its corresponding accepted value
  - n_p: highest prepare # seen
- Proposer maintains:
  - my_n: my proposal # in current Paxos
- Each round of Paxos has an instance #

Proposer

- PROPOSE(v)

  choose $my_n > n_p$

  send PREPARE(my_n) to all nodes

  if PREPARE_OK(n_a, v_a) from majority **then**

    v_a = v_a with highest n_a, or choose own v otherwise

    send ACCEPT (n_a, v_a) to all

    if ACCEPT_OK(n_a) from majority **then**

      # done

      send DECIDED(v_a) to all

Acceptor

- PREPARE(n)

  If $n > n_p$

  $n_p = n$

  reply <PREPARE_OK, n_a,v_a>

- ACCEPT(n, v)

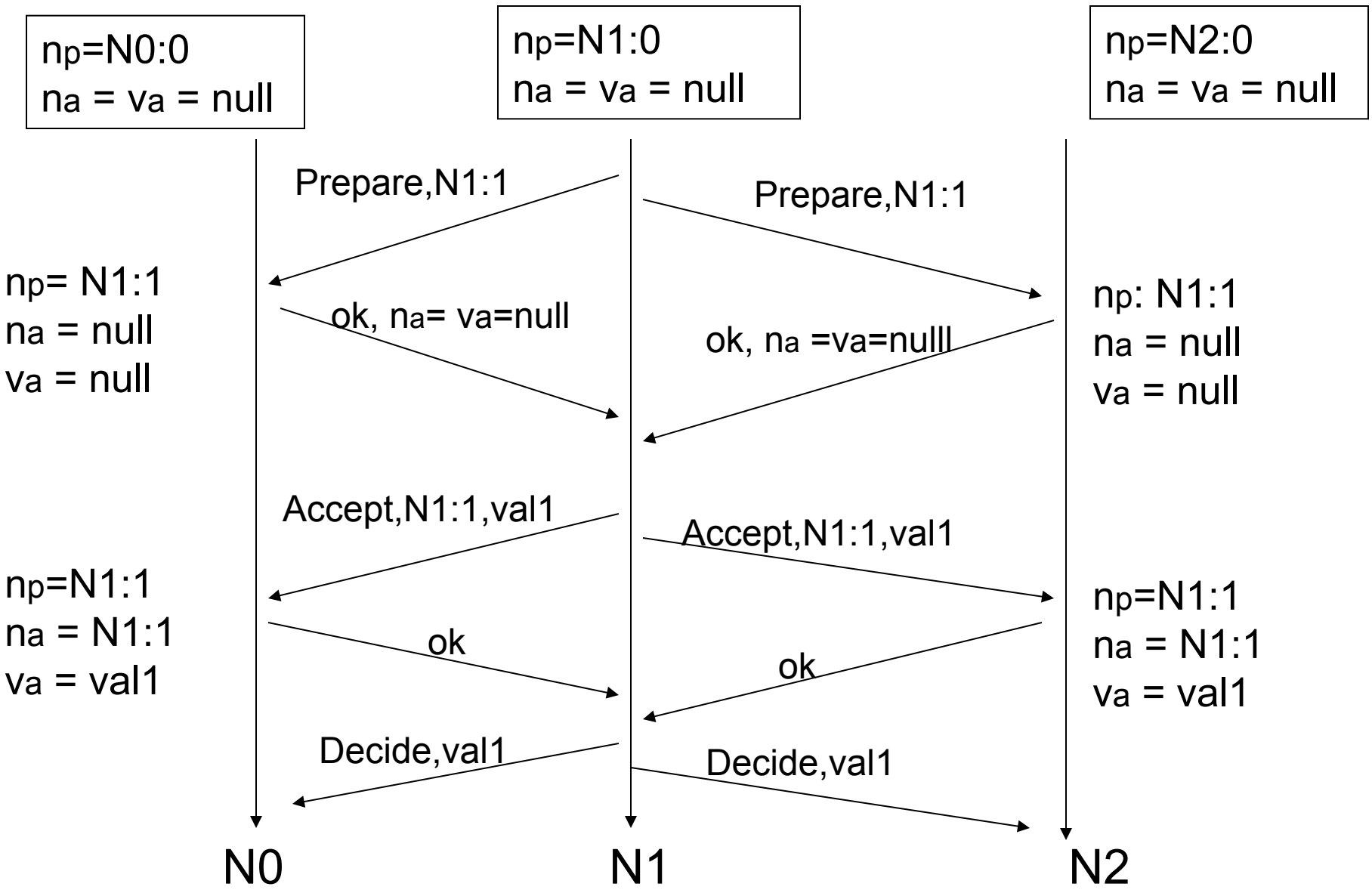
  If $n >= n_p$

  $n_a = n$

  $v_a = v$

  reply with <ACCEPT_OK, n>

Paxos operation: 3 phase example



Paxos properties

- When is the value V chosen?

  When majority of acceptors records n_a/v_a

- What if an acceptor doesn't hear accept announcement?

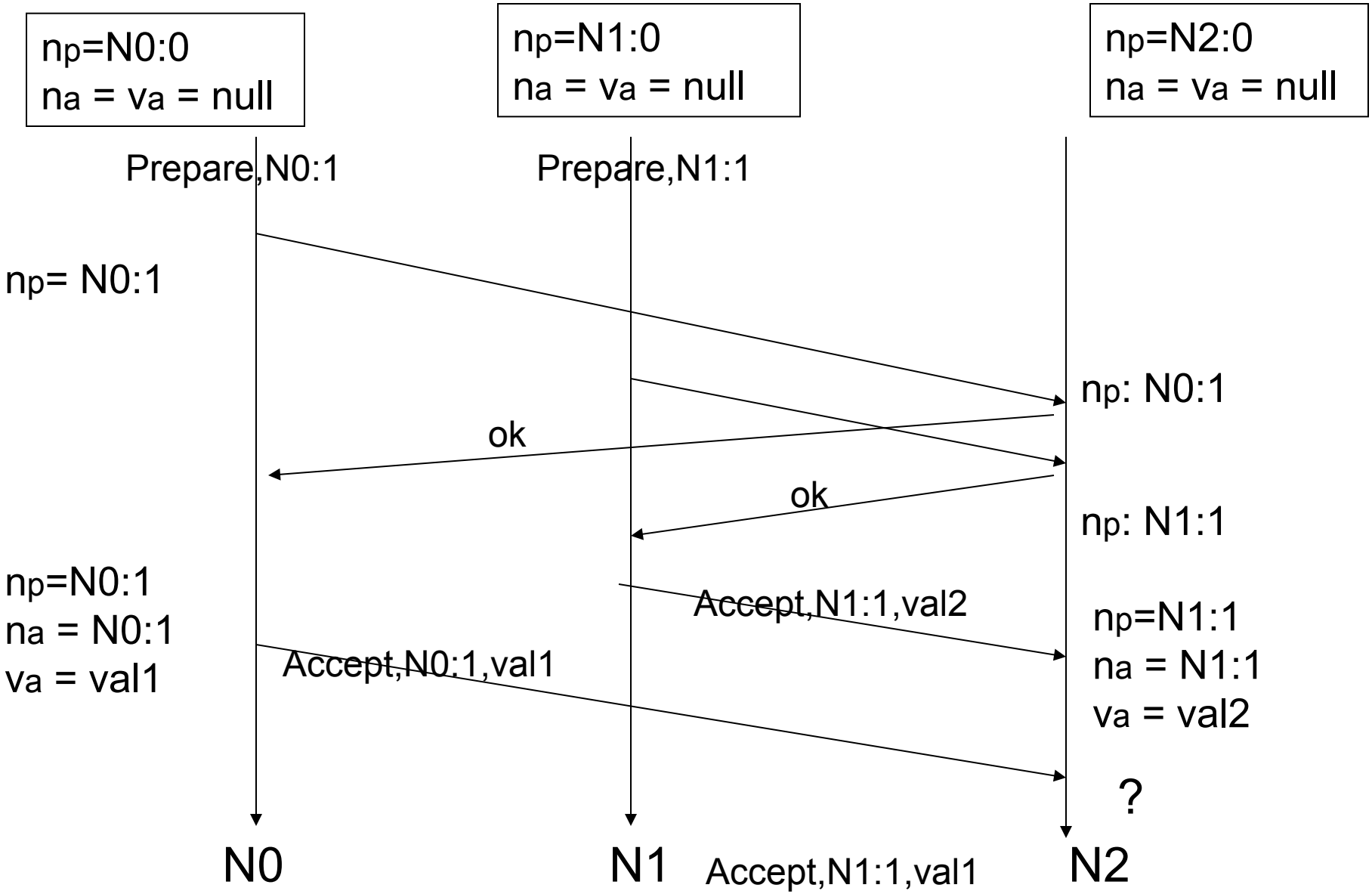
  Can start new round

  Will choose v_a, if majority accepted

Understanding Paxos

- What if more than one leader is active?
- Suppose two leaders use different proposal number, N0:10, N1:11
- Can both leaders see a majority of prepare-ok?

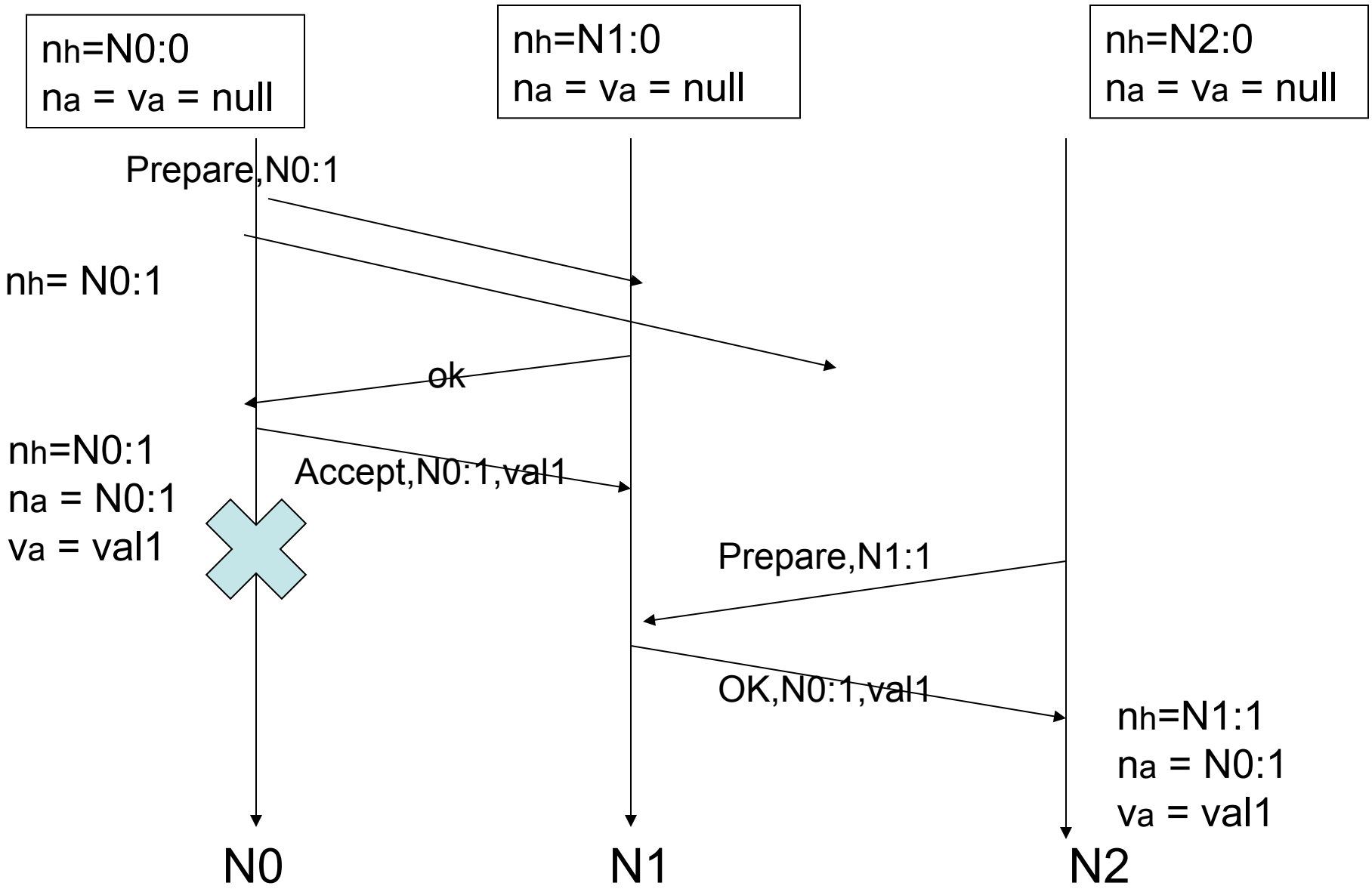
Scenario 1


np=N0:0
na = va = null
np=N1:0
na = va = null
np=N2:0
na = va = null
Prepare,N0:1
Prepare,N1:1
np= N0:1
np: N0:1
ok
ok
np: N1:1
np=N0:1
na = N0:1
va = val1
Accept,N1:1,val2
np=N1:1
na = N1:1
va = val2
Accept,N0:1,val1
?
N0
N1
Accept,N1:1,val1
N2

Understanding Paxos

- What if leader fails while sending accept?

Scenario 2


nh=N0:0
na = va = null
nh=N1:0
na = va = null
nh=N2:0
na = va = null
Prepare,N0:1
nh= N0:1
ok
nh=N0:1
na = N0:1
va = val1
Accept,N0:1,val1
Prepare,N1:1
OK,N0:1,val1
nh=N1:1
na = N0:1
va = val1
N0
N1
N2

Understanding Paxos

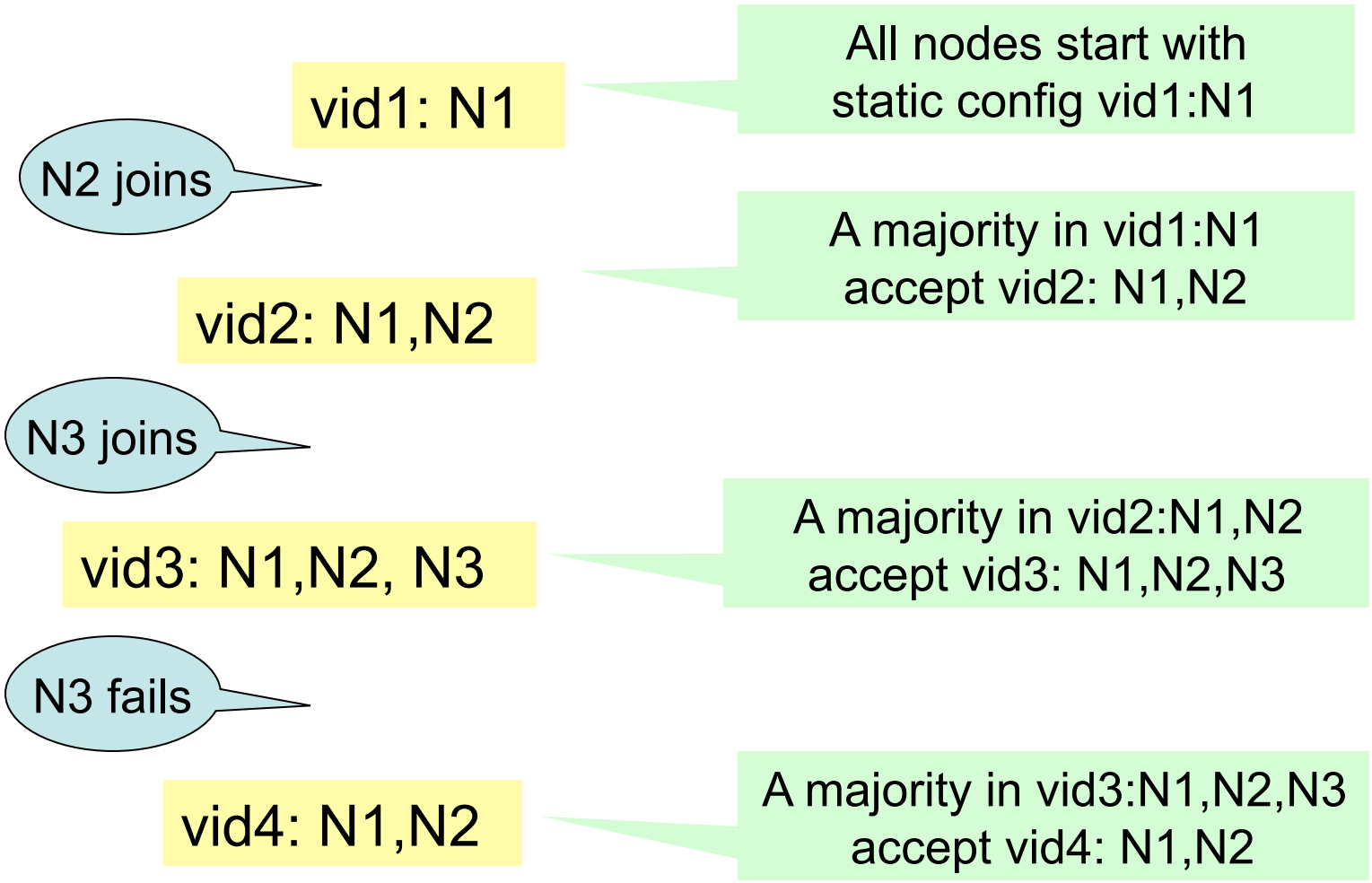
- What if acceptor fails after accepting?
  - For example, N1
  - If N0 and N1 down, N2 must wait
  - If N0 or N1 reboots, must use N0's val1!
    - Must log accepts + proposals (!), etc. on disk

Using Paxos for RSM

- RSM requires consistent replica membership
  - Membership: <primary, backups>
  - RSM goes through a series of membership changes

  <vid-0, primary, backups><vid-1, primary, backups> ..
- Use Paxos to agree on the <primary, backups> for a particular vid
  - vid == paxos instance #

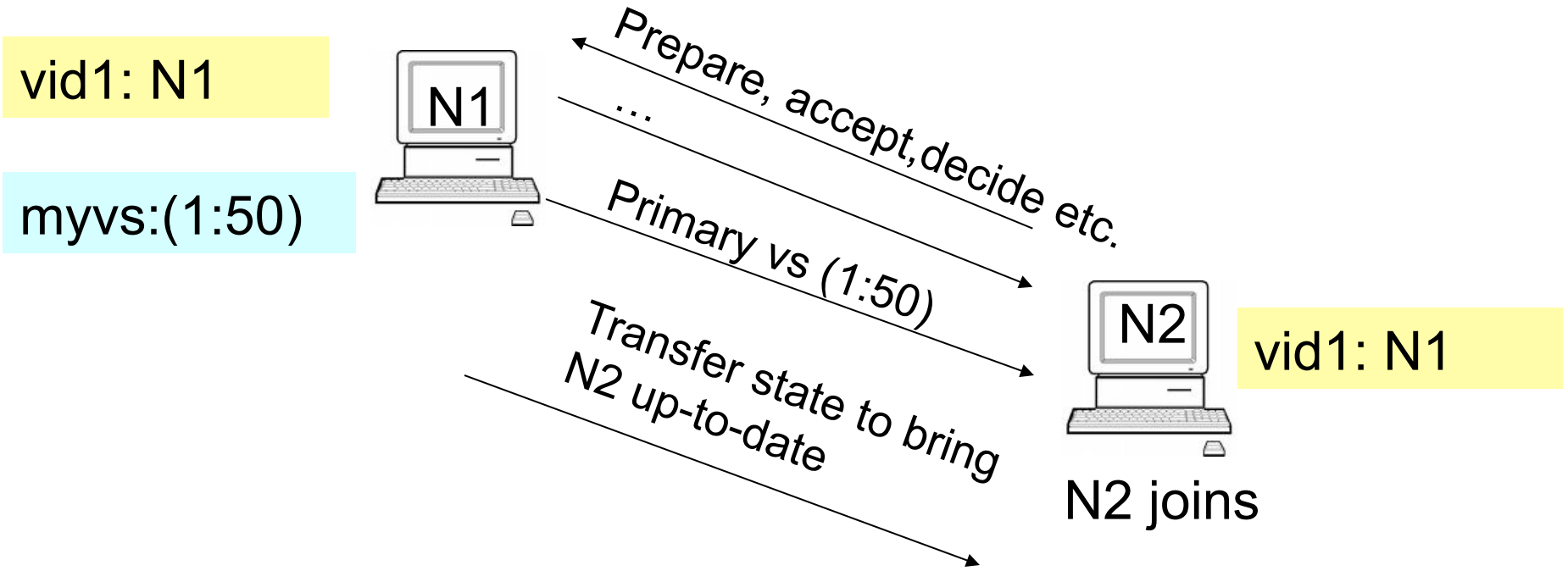
Example


vid1: N1
All nodes start with
static config vid1:N1
N2 joins
A majority in vid1:N1
accept vid2: N1,N2
vid2: N1,N2
N3 joins
vid3: N1,N2, N3
A majority in vid2:N1,N2
accept vid3: N1,N2,N3
N3 fails
vid4: N1,N2
A majority in vid3:N1,N2,N3
accept vid4: N1,N2

Viewstamp replication

- All ops have a viewstamp vs = (instance #, seq #)
- To execute an op with vs, a replica must have executed all ops < vs
- A newly joined replica need to transfer state to ensure its state reflect executions of all ops < vs

Example



- Primary in new view is last primary, if alive
- Otherwise, backup with highest view stamp
- Resume responding to client after backups and primary are in sync

Many optimizations/issues

- Send read ops only to primary
- Garbage collect logs
- …
- What if replicas lie (i.e., Byzantine)?

- Much more to say, take 6.824
  - You implement a RSM w. Paxos

Summary

- Fault tolerance -> replication
- Consistency
- Replicated state machine
- Paxos to achieve consensus
  - Hard case: network partition