

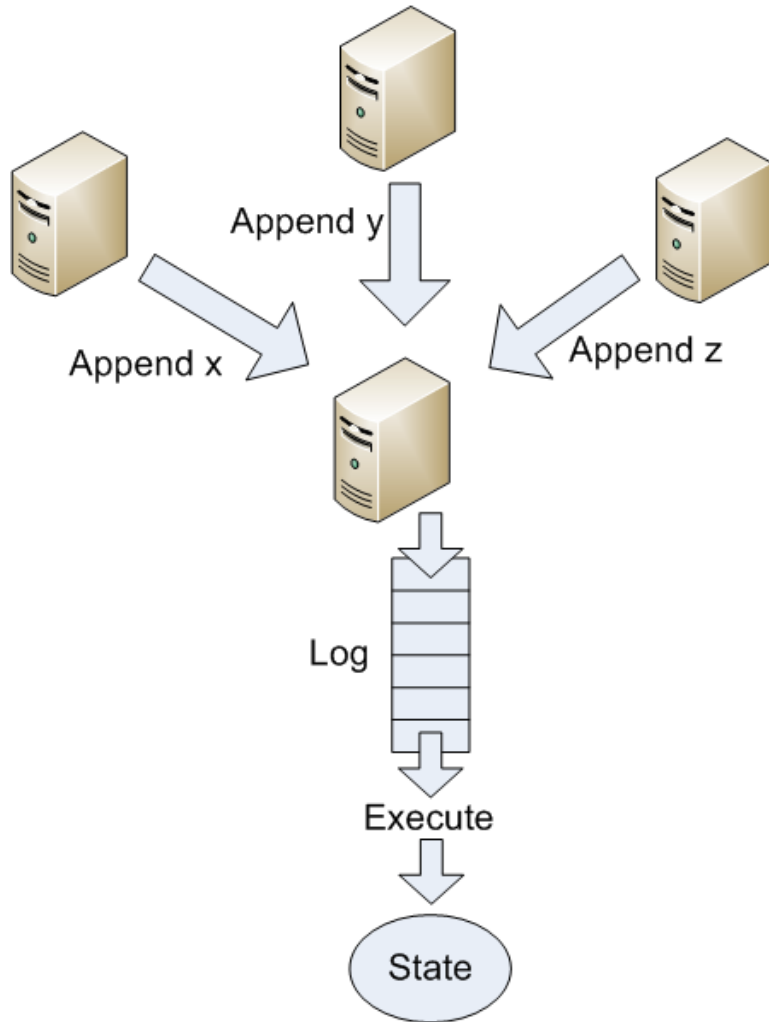
# Infrastructure Technologies for Large-Scale Service-Oriented Systems

Kostas Magoutis

magoutis@cse.uoi.gr

<http://www.cse.uoi.gr/~magoutis>

# Order on state updates



# Paxos algorithm

- Way to build fault-tolerant distributed systems
  - Replicated state machines (RSM)
- Consensus via message exchange
  - Asynchronous: no timing guarantees
  - Network can delay, reorder, lose (but not corrupt) packets
- Can guarantee safety
  - Replicas will agree on a single value
- Need additional assumptions to ensure progress

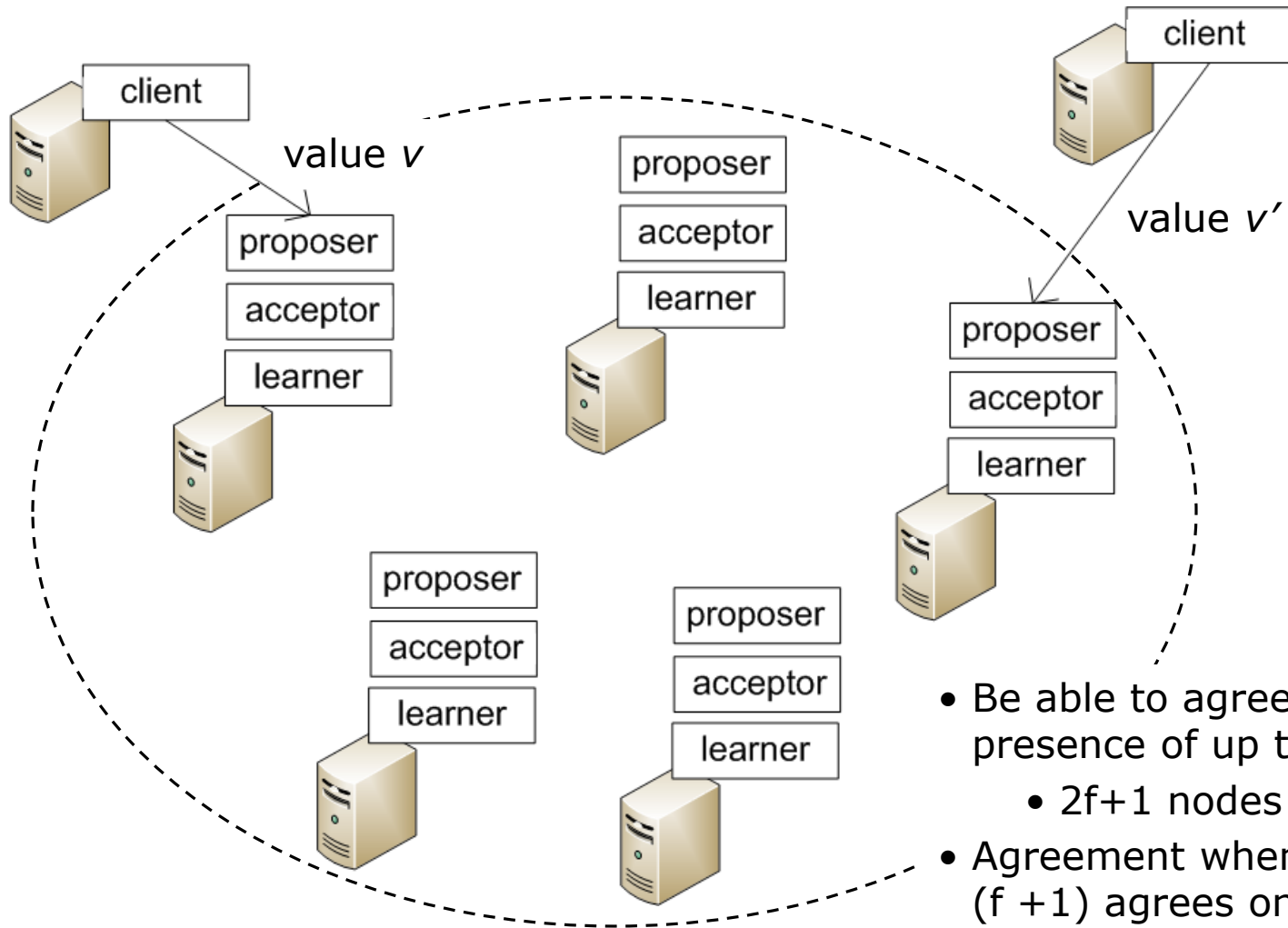
# Informally

- Three roles: Proposer, acceptor, learner
- Simplest, but fault-intolerant solution: single acceptor
- With  $>1$  acceptors, agreement by a majority required
- If single value proposed, that value should be chosen
  - Thus, an acceptor must accept the first value proposed to it
- However, this may lead to fragmented electorate
  - Multiple proposals by each proposer should be possible
  - Identify each proposal by a unique integer  $N$

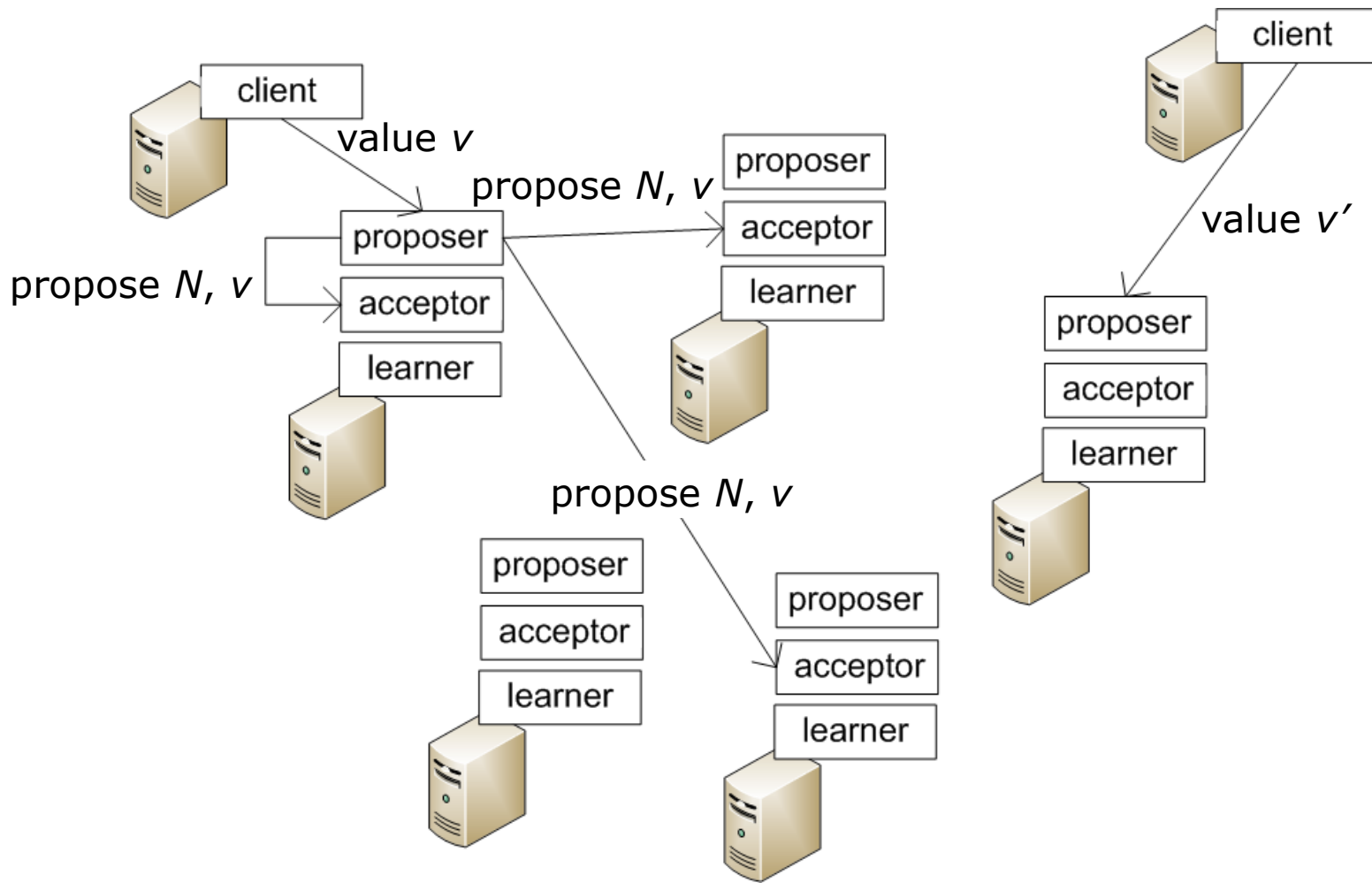
# Informally

- After consensus, an acceptor cannot change its mind
  - A value is chosen when single proposal with that value accepted by a majority of the acceptors
- Allow multiple proposals to be chosen, but guarantee that all chosen proposals have the same value

# Paxos setup



# Need to try to get a majority to accept



# Informally

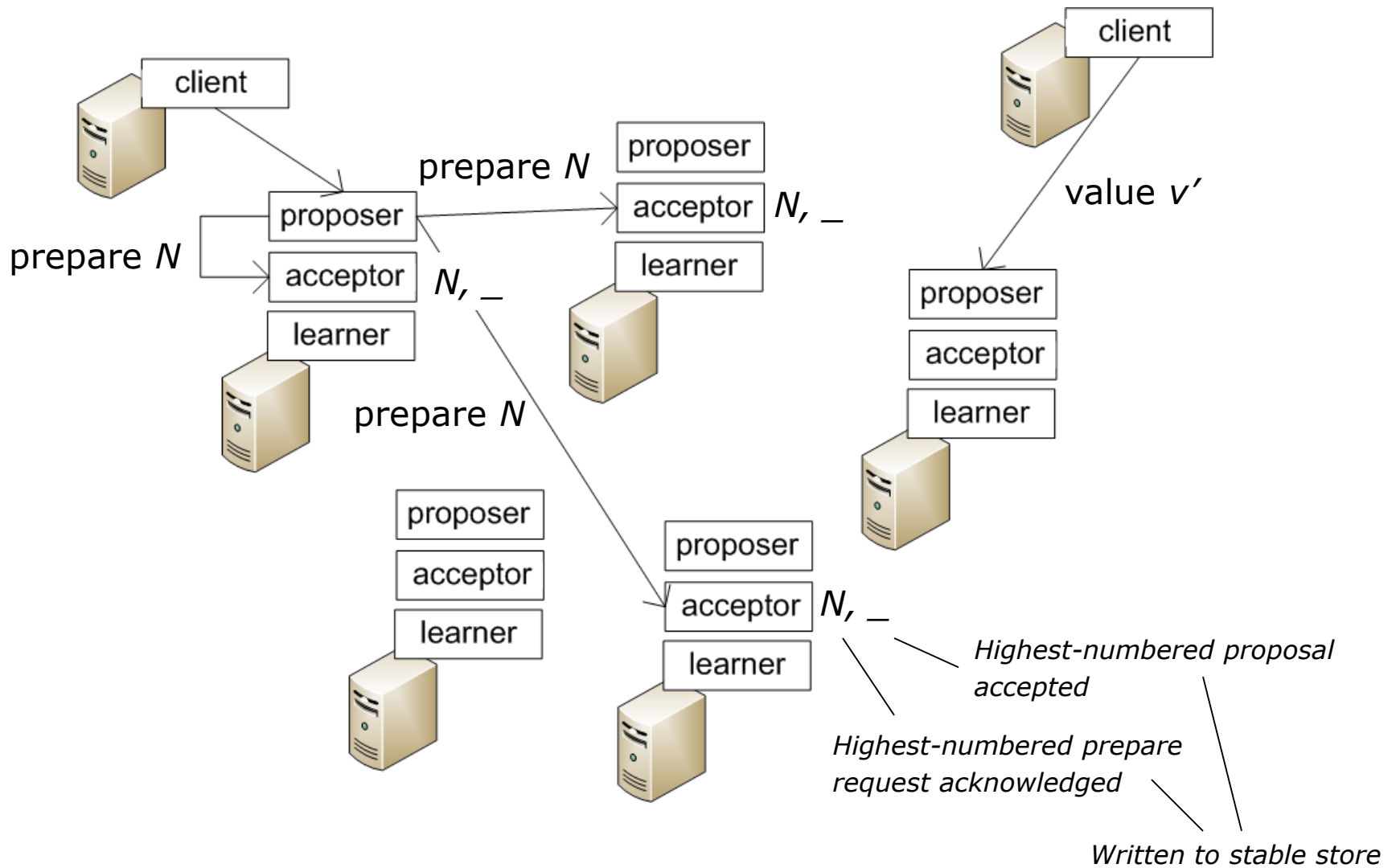
- Allow multiple proposals to be chosen, but guarantee that all chosen proposals have the same value
- If proposal  $N$  with value  $v$  is chosen, every higher numbered proposal issued by any proposer should have value  $v$
- A proposer wanting to issue a proposal numbered  $N$  must learn the highest-numbered proposal  $< N$  (if any) that has been or will be accepted by a majority



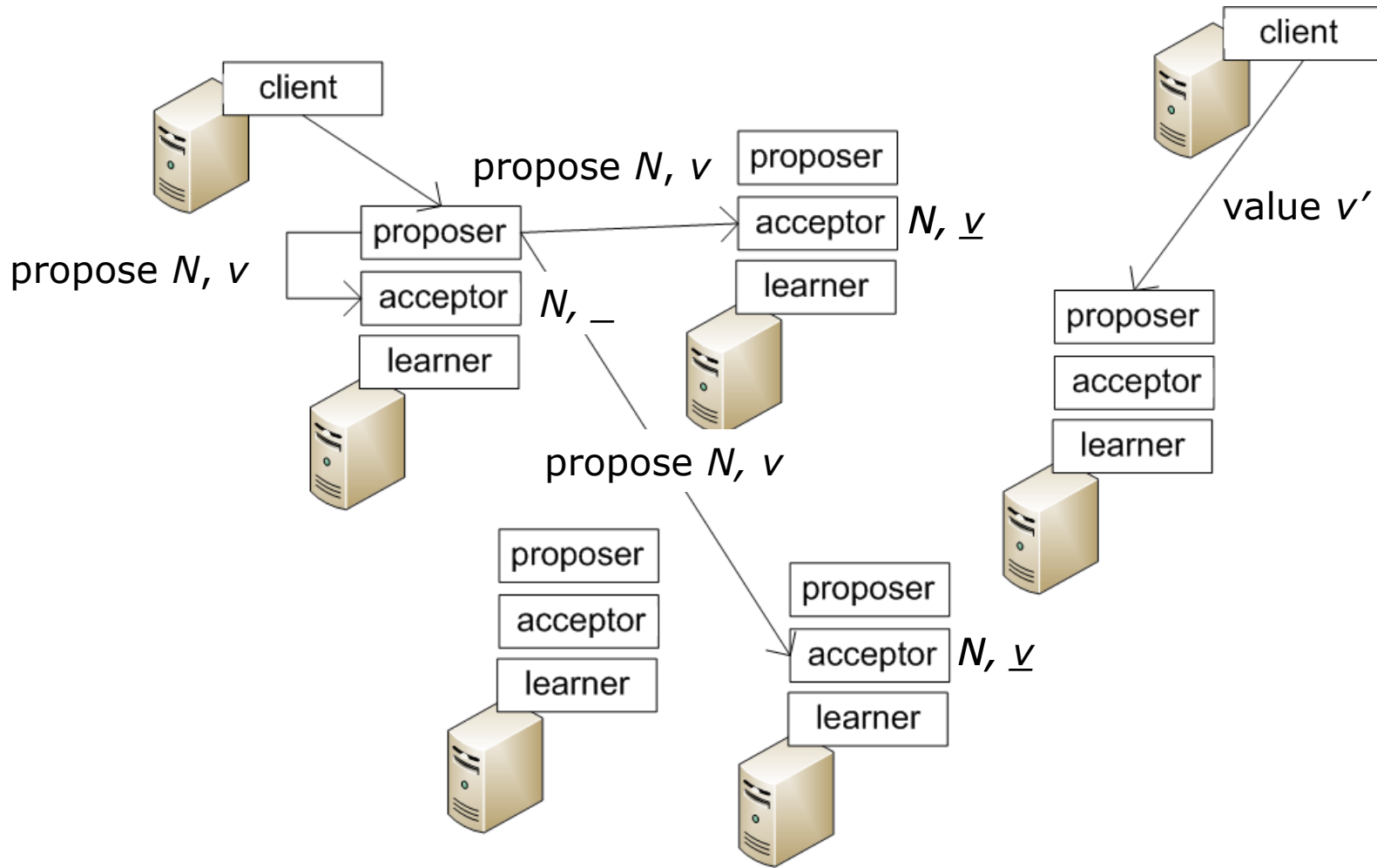
# Informally

- A proposer wanting to issue a proposal numbered  $N$  must learn the highest-numbered proposal  $< N$  (if any) that has been or will be accepted by a majority
  - Easy to learn about values already accepted
  - Hard to predict the future
- Control the future by extracting a promise that there will not be any acceptances of proposals  $< N$

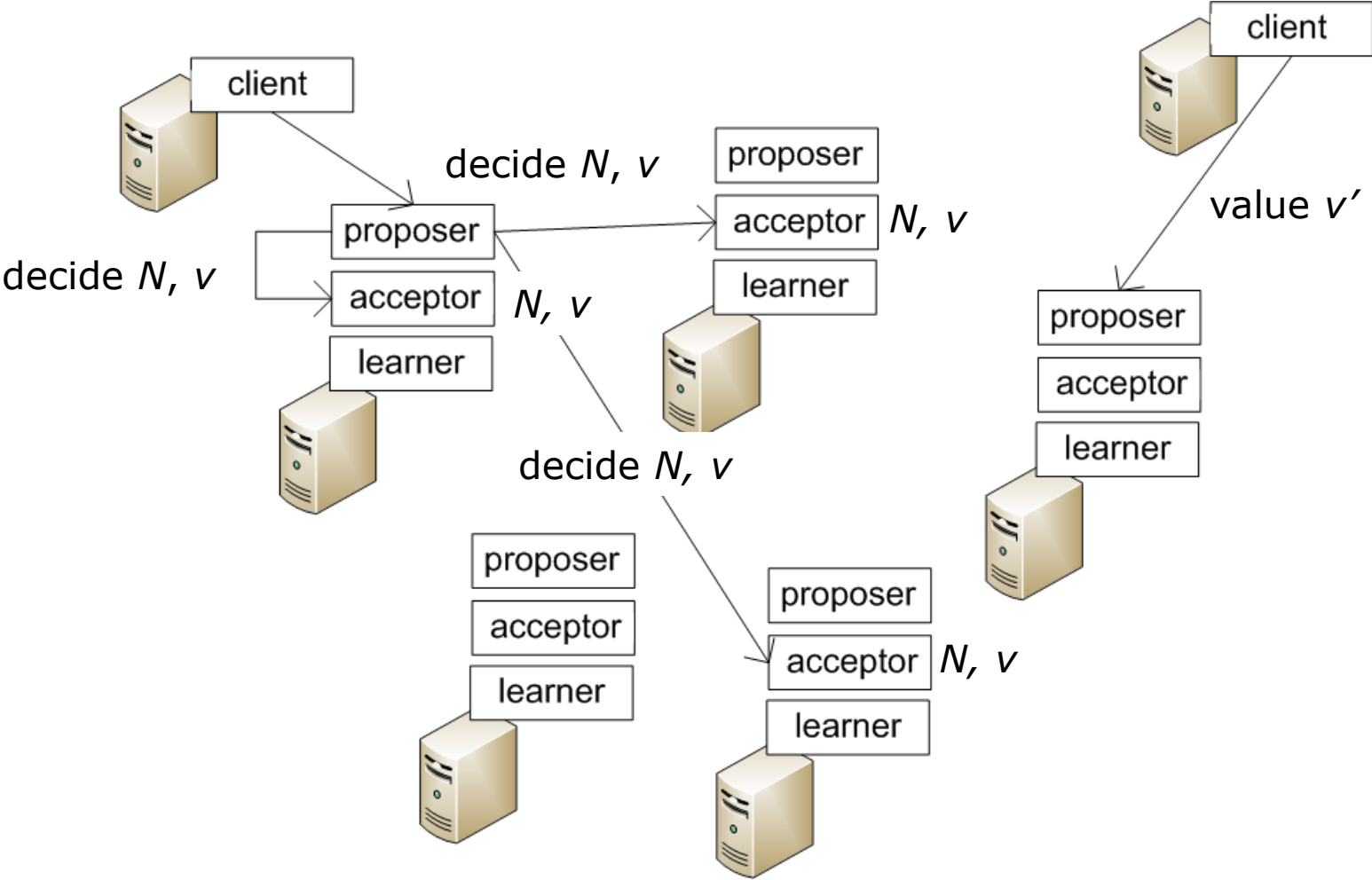
# Paxos – phase 1



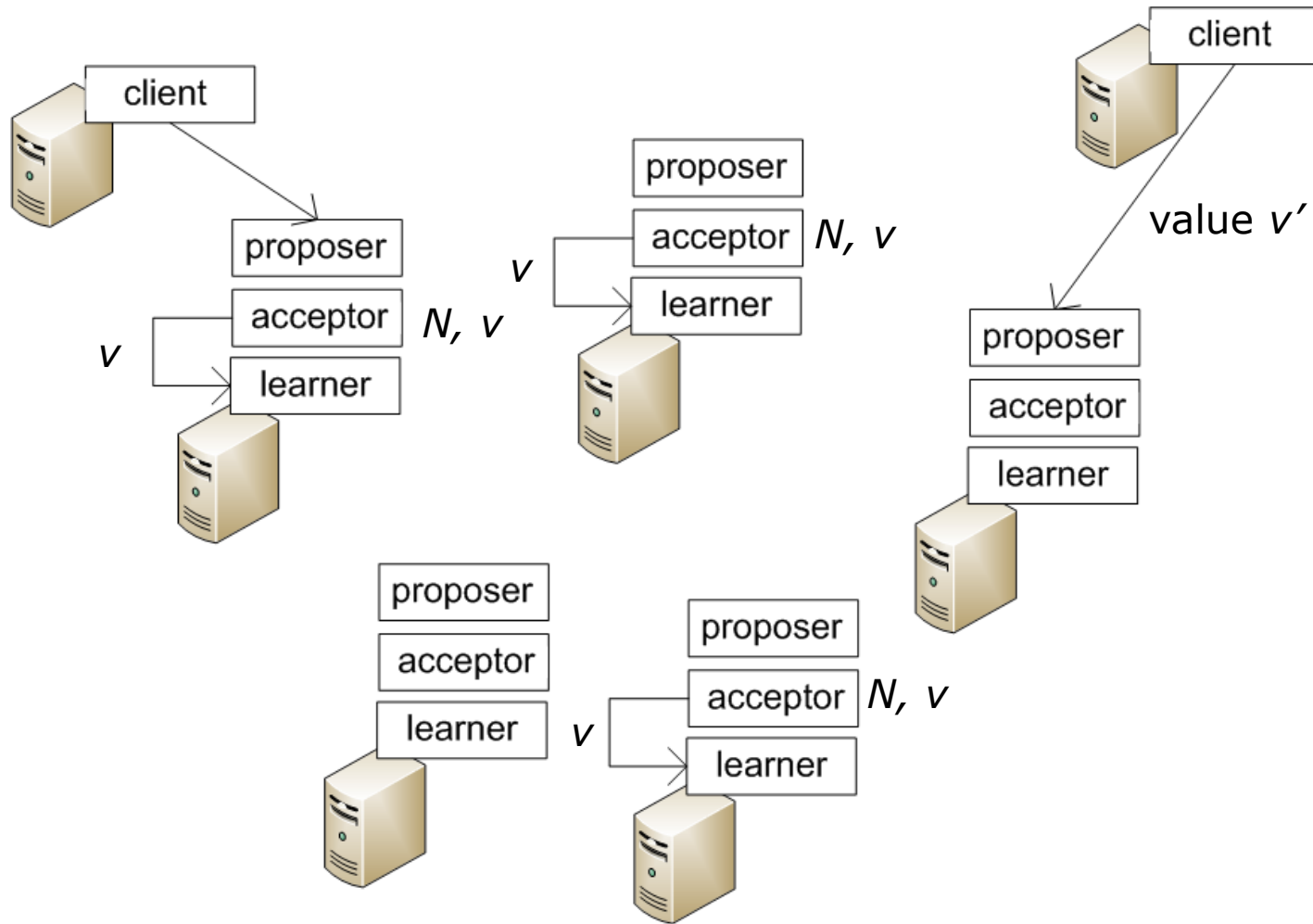
# Paxos – phase 2



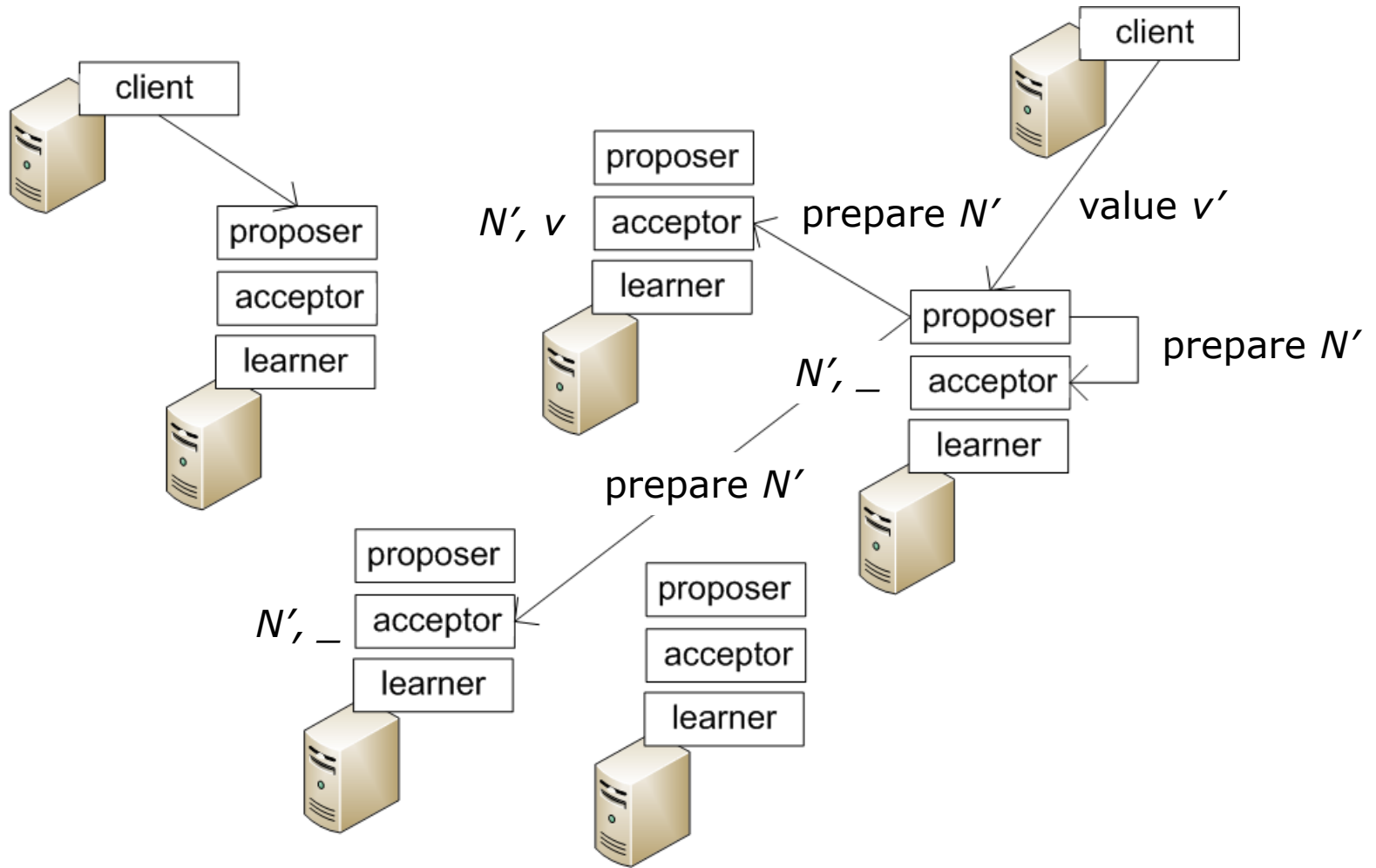
# Paxos – communicate agreement



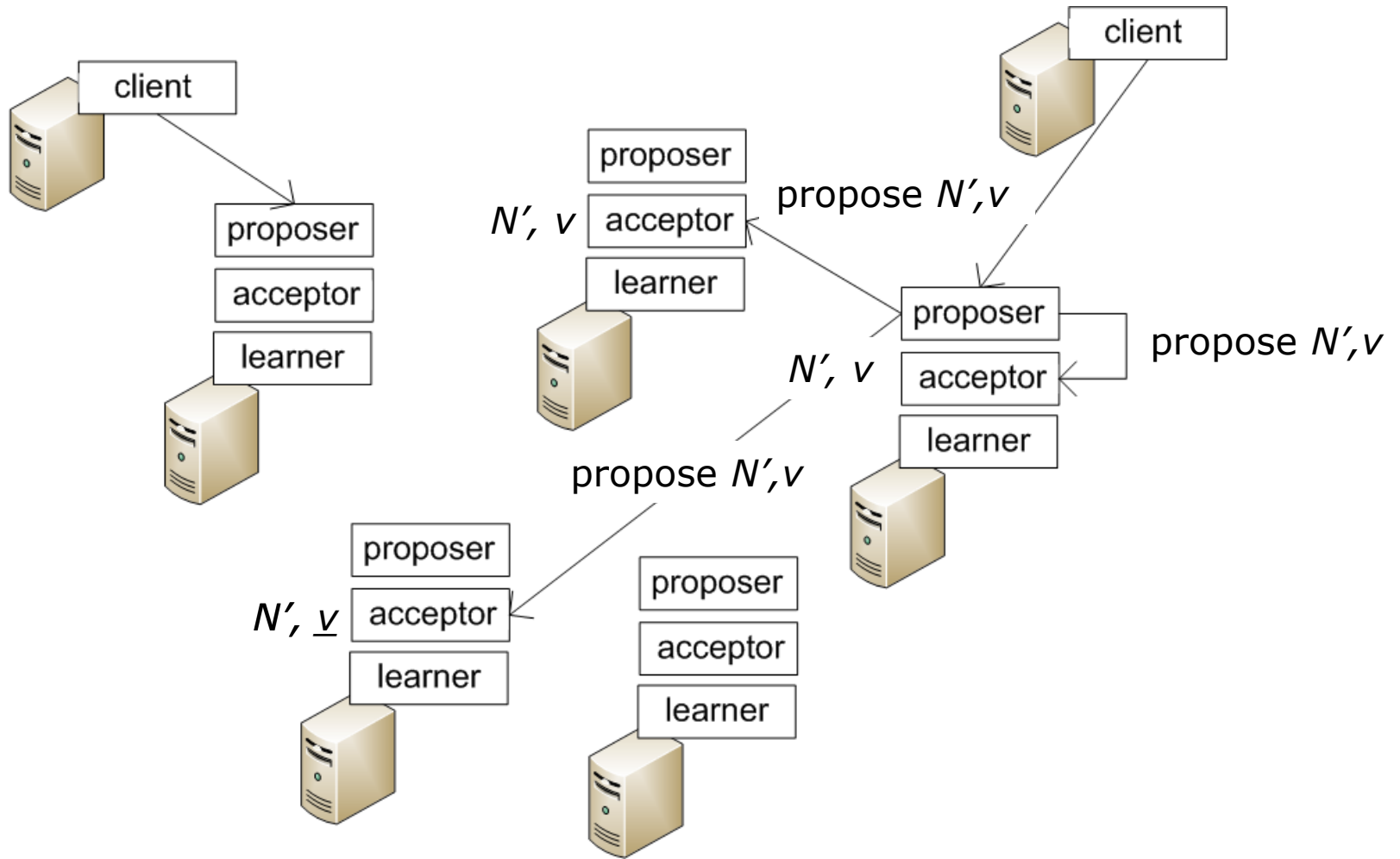
# Paxos – majority learns outcome



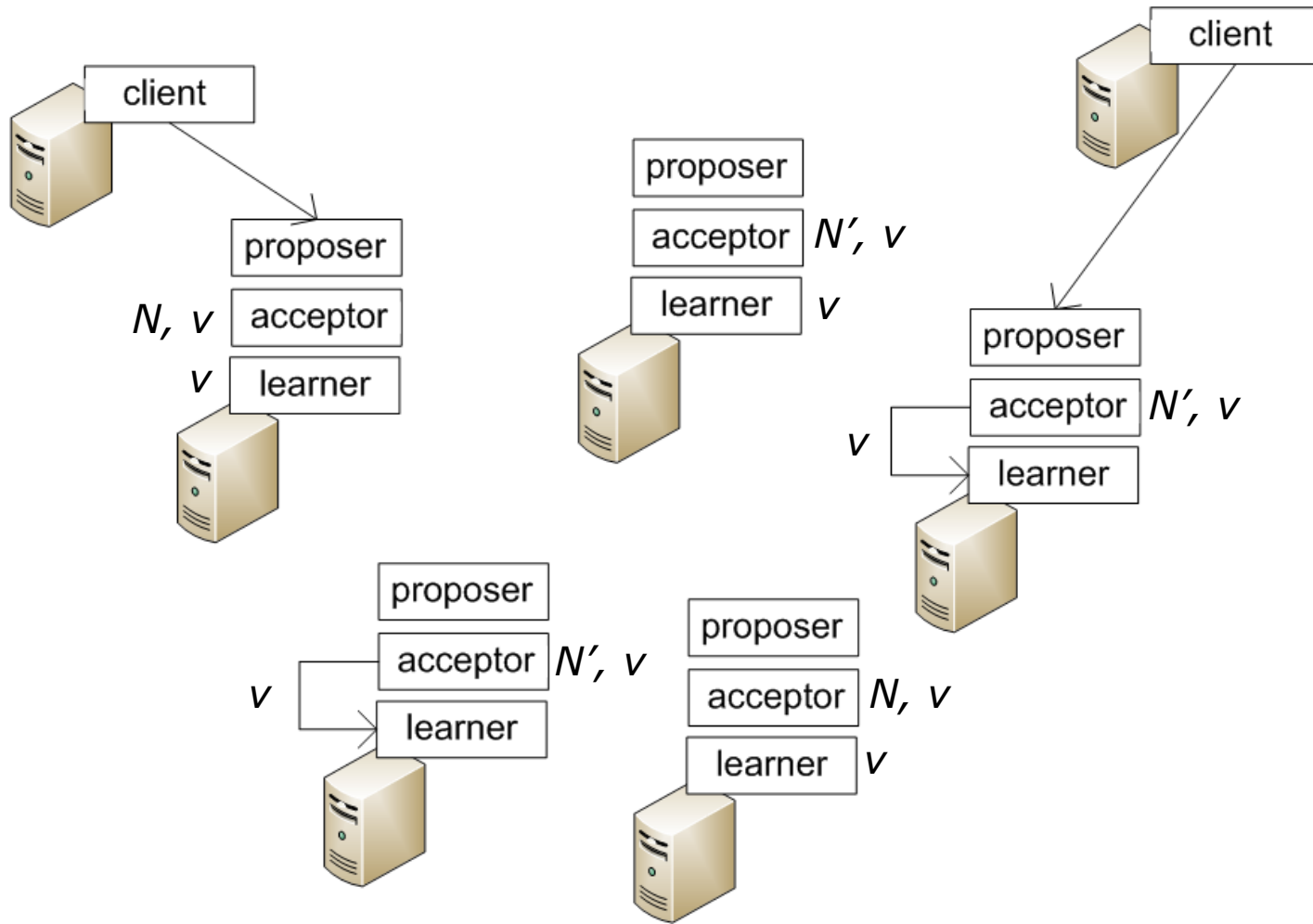
# Paxos – learning chosen value



# Paxos – propagate chosen value

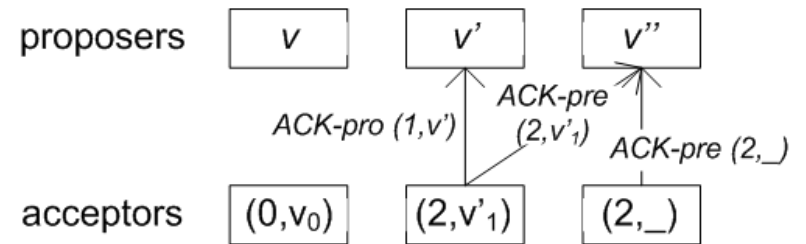
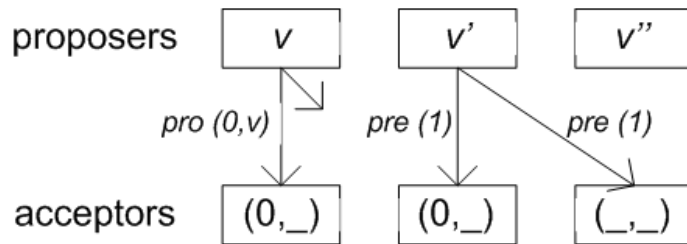
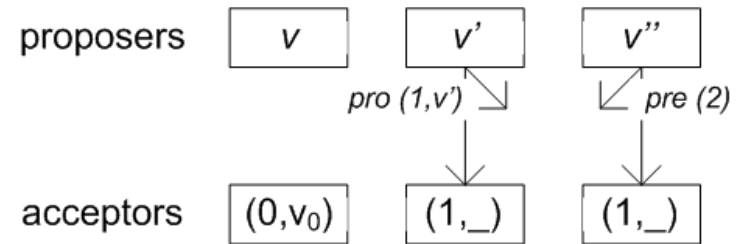
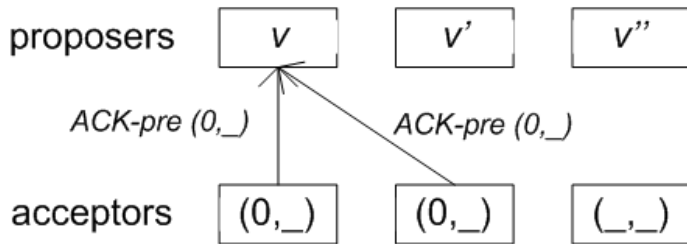
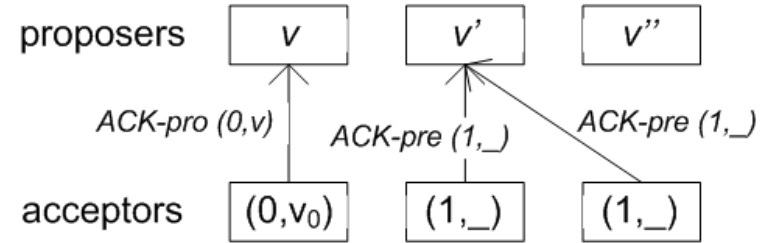
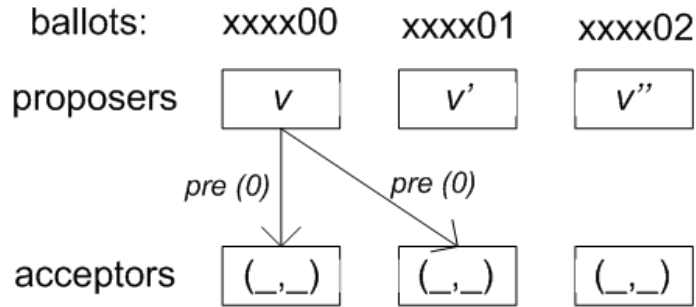


# Paxos – everyone learns outcome

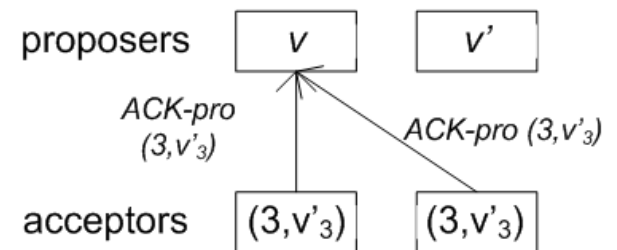
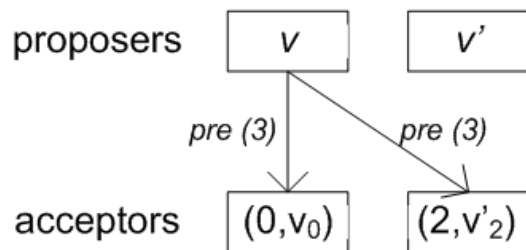
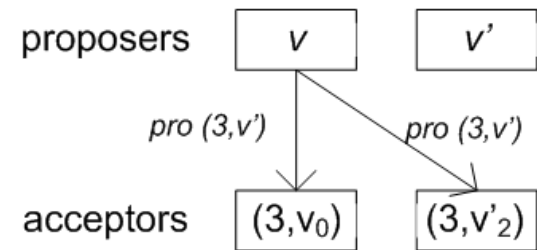
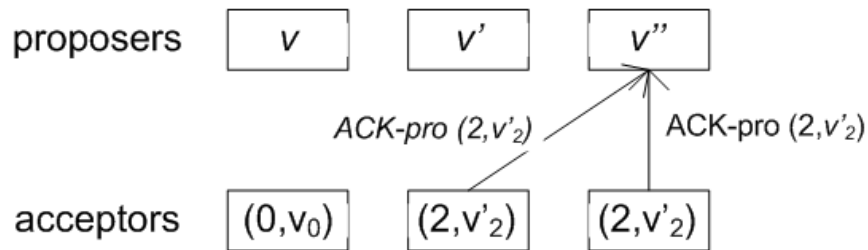
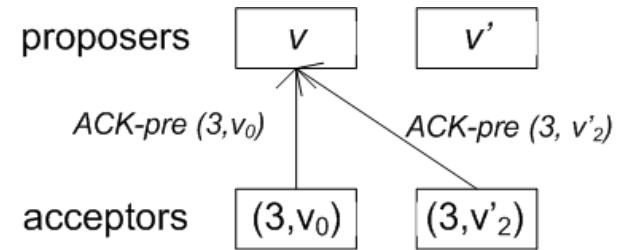
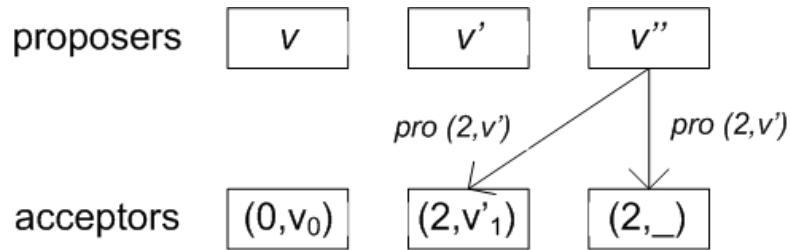




# Example



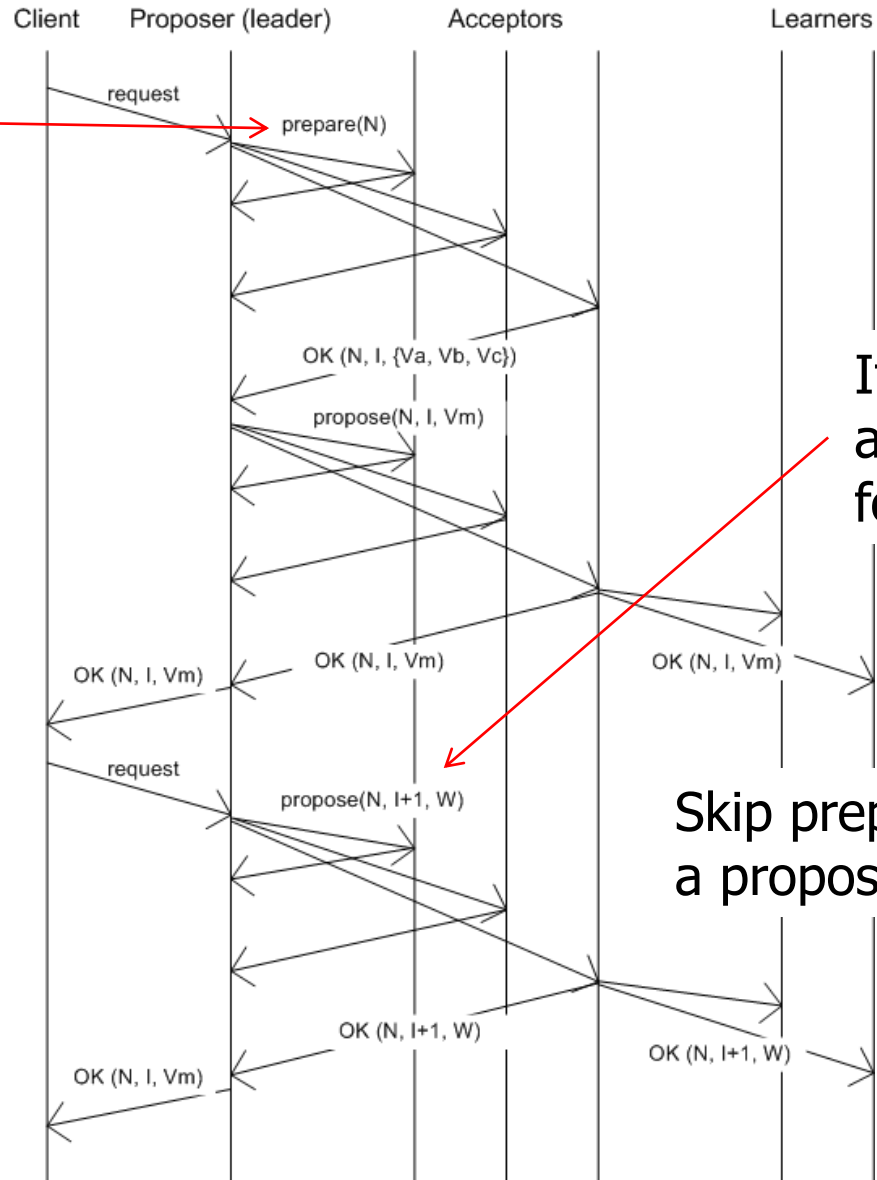
# Example (*contd.*)



# Lamport: implementing a state machine

- How to run multiple instances of Paxos
  - A new leader may know the outcome of only some instances
  - There may be gaps in the chosen instances (1-134, 138, ..)
  - A new leader will try to fill in those slots or propose *no-op*
  - As soon as gap fills, commands can be executed
- Multi-Paxos
  - New leader: execute phase 1 for infinitely many instances
  - Acceptors can respond with reasonably short messages
  - Cost of Paxos effectively the cost of executing phase 2

# Multi-Paxos

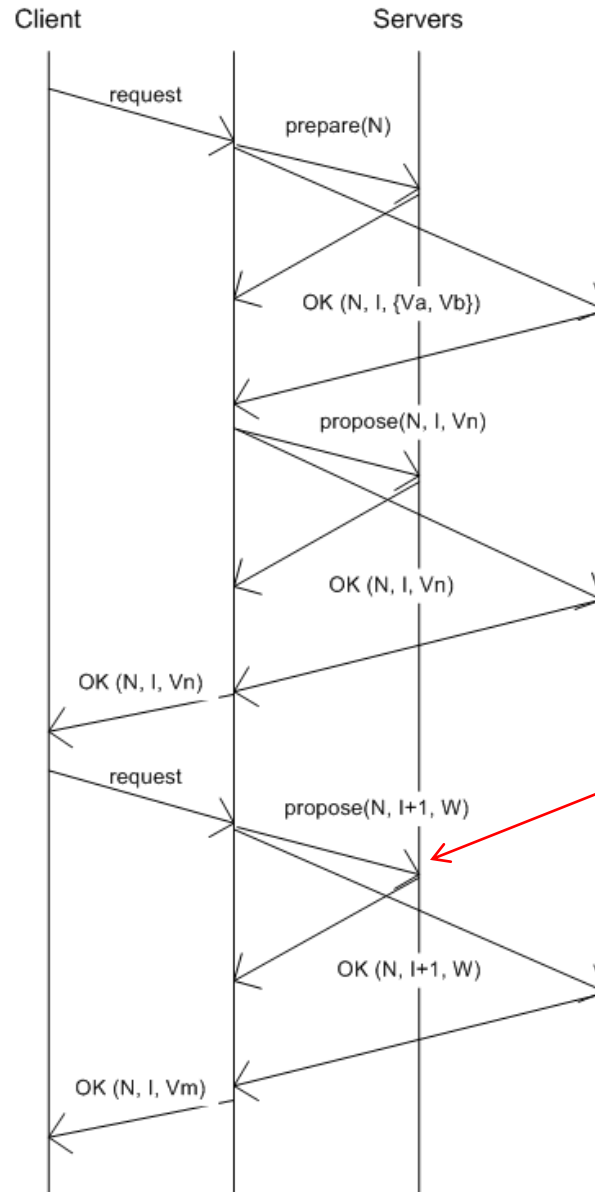


Block acceptance of proposal # < N & learn accepted values

If a majority has not accepted anything for instances > I

Skip prepare phase until a propose is rejected!

# Multi-Paxos



Servers play all roles

Replicas write to disk prior to sending ACK