CONSENSUS

Murat Demirbas, SUNY Buffalo

several slides borrowed/modified from Jeff Chase, Duke
http://www.cs.duke.edu/~chase/cps212/consensus.pdf

Consensus specification

N nodes; each with value v_i, decision d_i, completion t_i

Agreement: No two process can commit different decisions $\forall i,j : t_i \wedge t_j : d_i = d_j$

Validity: If all initial values are equal, nodes must decide on that value $\exists k :: (\forall i :: v_i = k) \Rightarrow (\forall i : t_i : d_i = v_i)$

Termination: Nodes decide eventually $true \rightarrow (\forall i :: t_i)$

Paxos, the best so far...

A 3PC protocol that works, with correctness proof

Preserves safety against asynchrony & message loss

Achieves progress when the conditions improve

How does Paxos work?

There are N nodes, some (ideally one) act as a leader

Leader presents a consensus value to the acceptors, counts the ballots for acceptance of the majority, and notifies acceptors of success

Paxos can mask failure of a minority of N nodes

Rounds and ballots

Paxos proceeds in rounds. Each round has a uniquely numbered ballot. Each round has 3 phases.

If no failures, then consensus is reached in one round.

Any would-be leader can start a new round on any (apparent) failure.

Consensus is reached when some leader successfully completes a round.

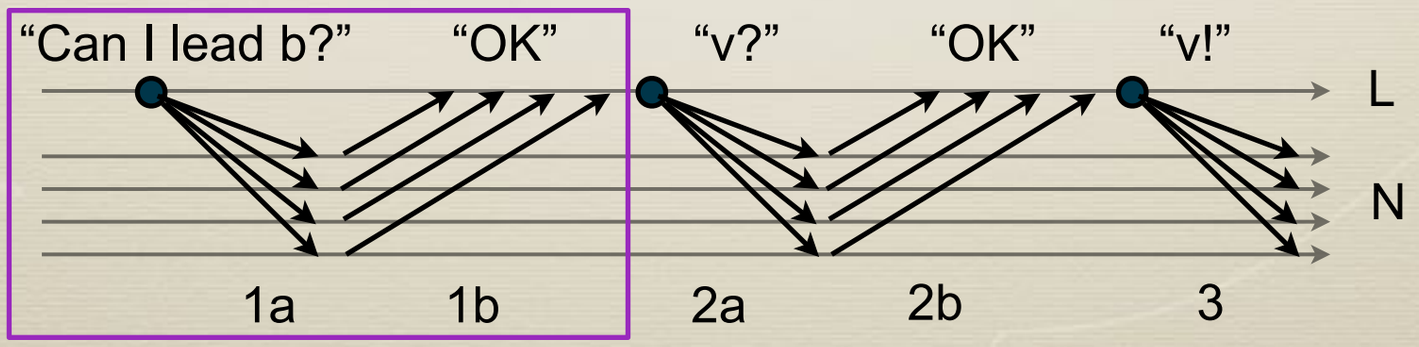
Phase 1: Proposing a round

Would-be leader chooses a unique ballot ID (round#)

Propose to the acceptors/agents (1a). Will you consider this ballot with me as leader?

Agents return the highest ballot ID seen so far (1b). Seen one higher than yours? That's a rejection.

If a majority respond and no one knows of a higher ballot number, then you are their leader (for this round)



Phase 2-3: Leading a round

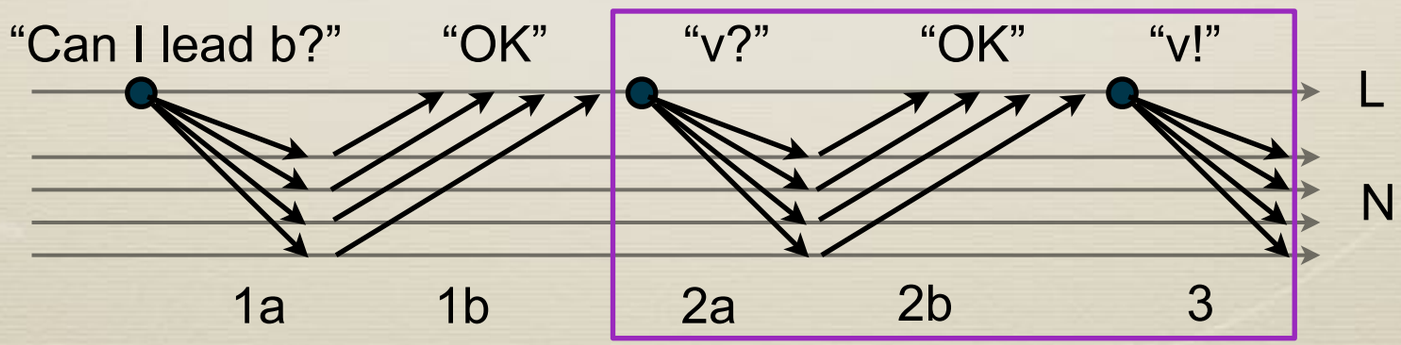
Choose a "suitable value"™ for this ballot.

Command the agents to accept the value (2a).

Did a majority respond (2b) and assent?

Yes: tell everyone the round succeeded (3).

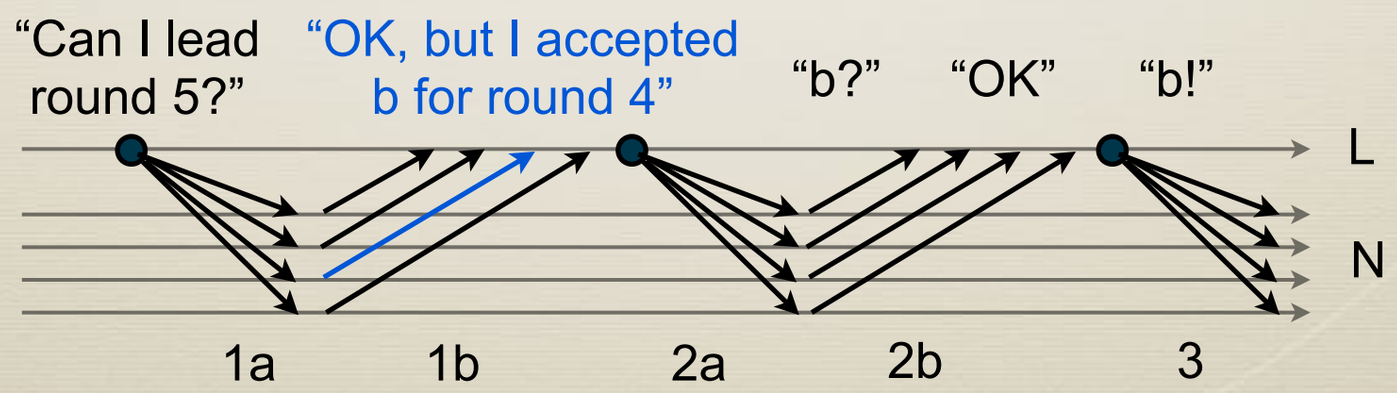
No: move on, e.g., ask for another round.



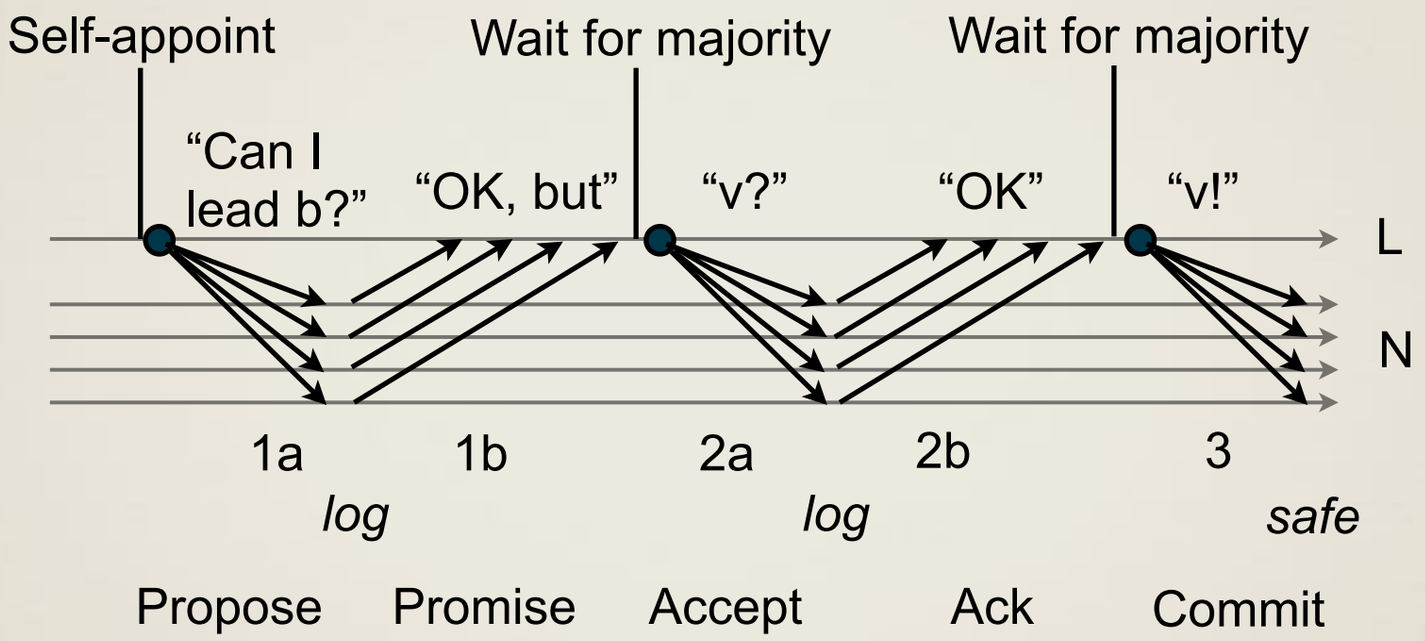
Choosing a suitable value

A majority of agents responded (1b). Did any accept a value for some previous ballot (in a 2b message)? No: choose any value you want.

Yes: they tell you the ballot ID and value. Find the most recent value that any responding agent accepted, and choose it for this ballot too.



Where is the anchor point?



Anchoring a value

A round anchors if a majority of agents hear the command (2a) and obey. (that value is chosen/anchored, even though no nodes may individually know of this!)

The round may then fail if many agents fail, many command messages are lost, or another leader usurps

But: safety requires that once some round anchors, no subsequent round can change it.

The system may have another round, possibly with a different leader, until all nodes learn of the success.

Class reenactment

5 students come to the board. Each student draws a ledger for round# and value.

scenario 1: fault-free, the leader completes in a round

scenario 2: leader dies after making majority accept, one node becomes leader, re-learns and commits value

scenario 3: leader dies before making majority accept, one node becomes leader, commits another value, old leader wakes up, re-learns and commits value.

Class reenactment ...

scenario 4: two leaders duel, progress violated, as the leaders keep undoing what the other tries to do. Finally, one leader drops the race, the remaining leader commits

scenario 5: the network is partitioning. If a majority partition exists it makes progress. Other partitions cannot make progress. Progress is made when network is connected again.

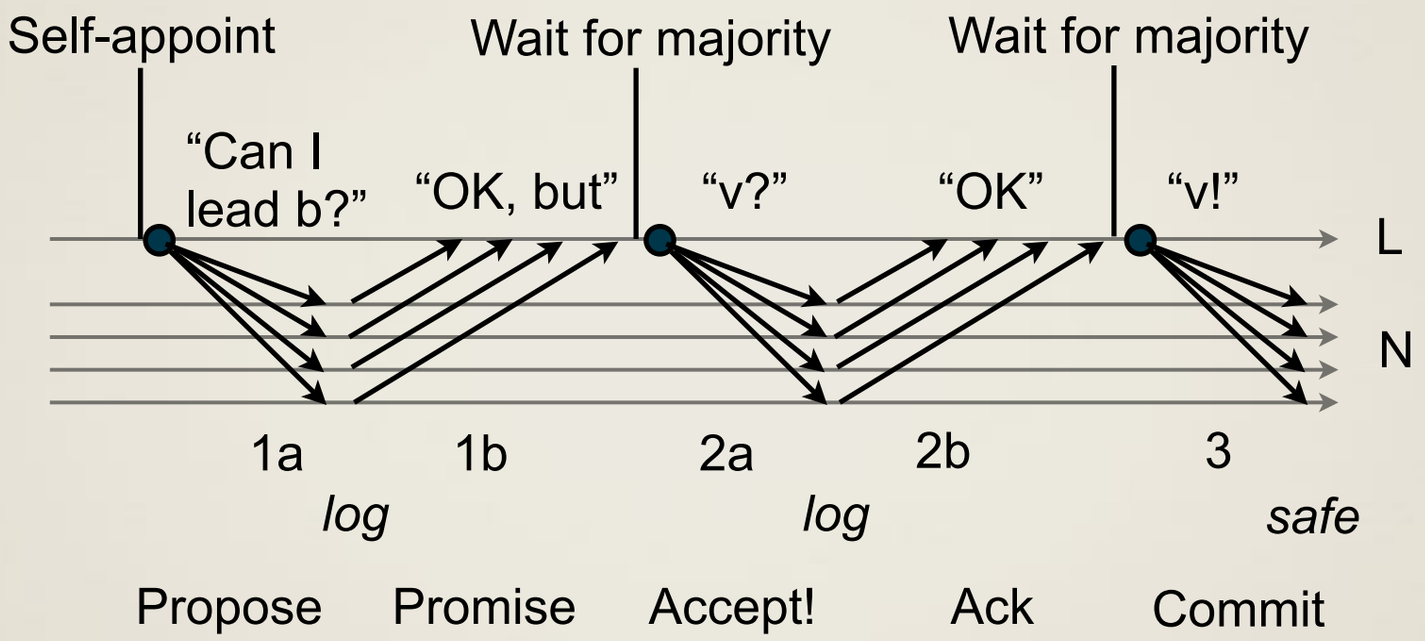
Why does Paxos work?

Key invariant: If some round commits, then any subsequent round chooses the same value, or it fails.

Consider leader L of a round R that follows a successful round P with value v, then either L learns of (P, v), or else R fails. Why? P got responses from a majority: if R does too, then some agent responds to both.

If L does learn of (P, v), then by the rules of Paxos L chooses v as a "suitable value"™ for R.

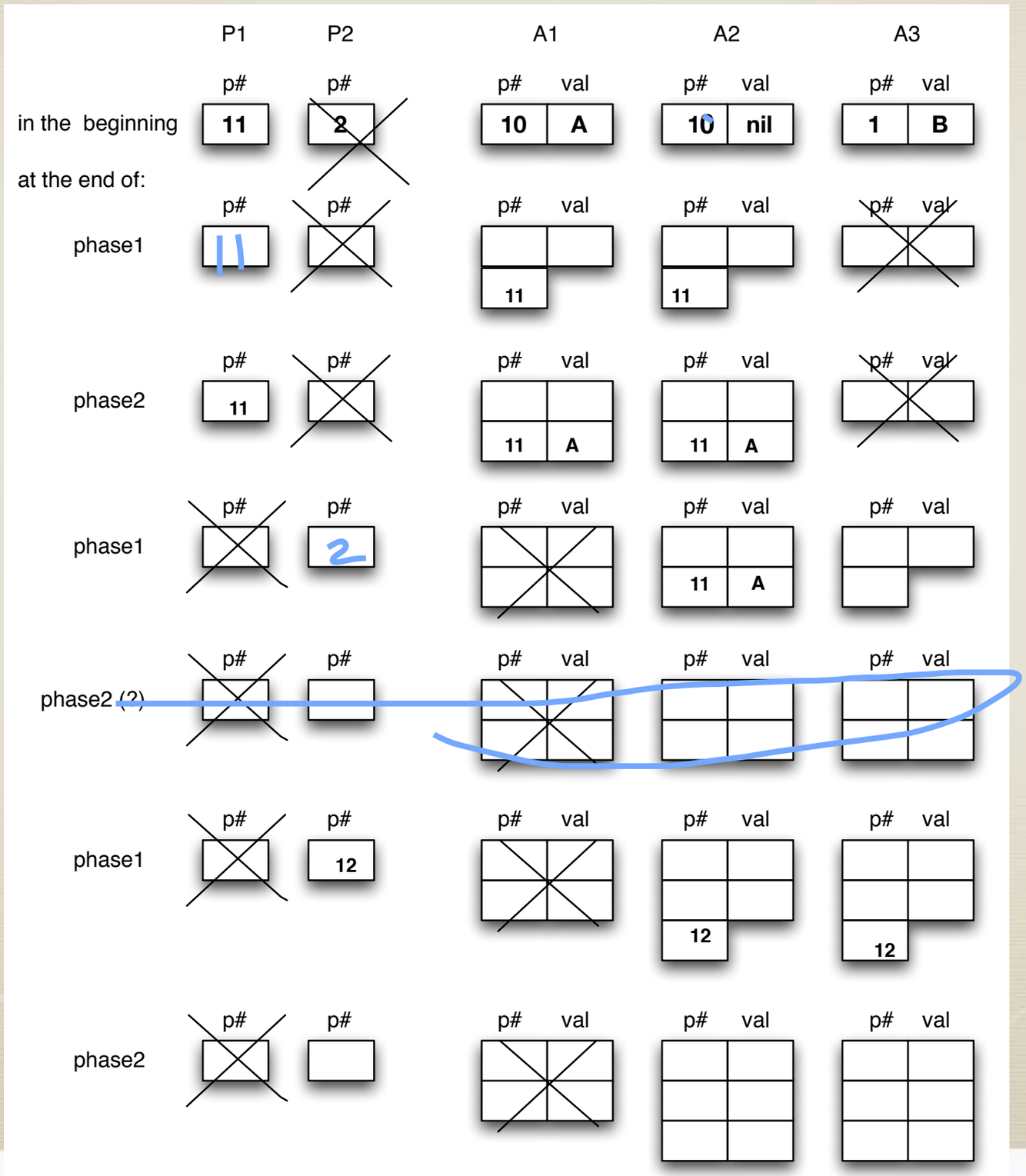
Paxos summary



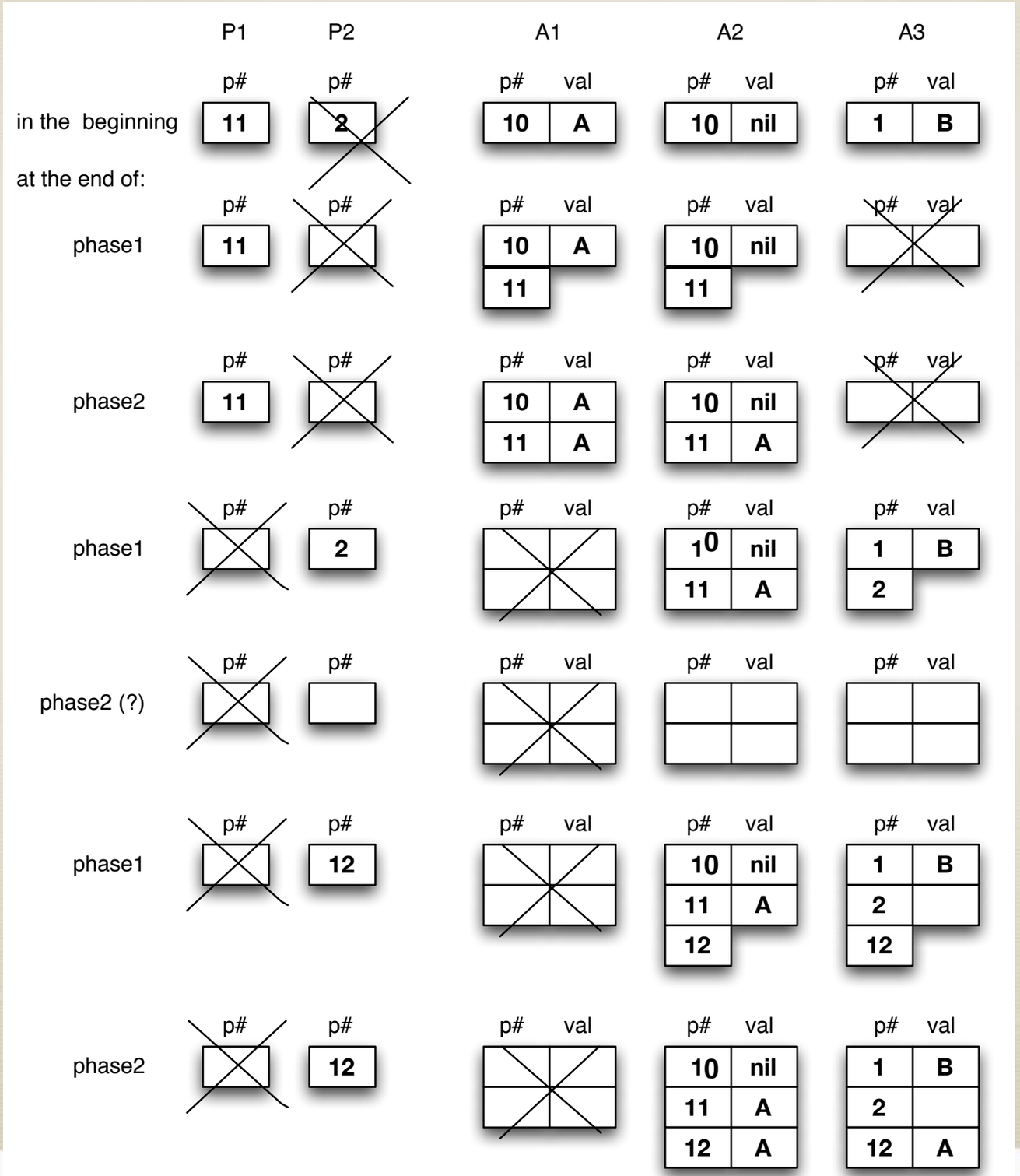
Paxos can be made efficient, and serve as the building block of highly consistent and partition-resilient systems with pretty good best-effort availability

Paxos exercise (again)

7. Paxos (10 points) Consider a system running the Paxos consensus algorithm with two proposers, $P1$, $P2$, and three acceptors, $A1$, $A2$, and $A3$. In the system the channels are reliable, so there are no message losses. However, nodes can crash, which we denote as putting a cross on the node. No learning takes place among acceptors. With these in mind, fill out the empty boxes in the following execution. $p\#$ denotes the proposal number and the val denotes the value of the proposal.



7. Paxos (10 points) Consider a system running the Paxos consensus algorithm with two proposers, $P1$, $P2$, and three acceptors, $A1$, $A2$, and $A3$. In the system the channels are reliable, so there are no message losses. However, nodes can crash, which we denote as putting a cross on the node. No learning takes place among acceptors. With these in mind, fill out the empty boxes in the following execution. $p\#$ denotes the proposal number and the val denotes the value of the proposal.



PAXOS MADE LIVE

Chandra, Griesemer, Redstone
Google 2007

Chubby

Chubby is a fault-tolerant system at Google. Typically there is one Chubby instance ("cell") per data center

Several Google systems (Google-Filesystem, Bigtable,..) use Chubby for distributed coordination/locking and to store a small amount of metadata.

Chubby ...

Chubby achieves fault-tolerance through replication. Chubby cell consists of 5 replicas. One replica is master

Every Chubby object (lock or file) is stored as an entry in a database. It is this database that is replicated.

Clients contact cell for service. Master replica serves all requests. If client contacts a replica, replica replies with master's network address. Client then contacts master.

How does Paxos fit here?

Each replica maintains a local copy of the request log

Paxos is used for ensuring all replicas have identical sequences of entries in their local logs despite faults

This is a standard replicated state machine approach to fault-tolerance

Multi-Paxos

This is an optimization to reduce the number of phases involved by chaining together multiple Paxos instances
Propose messages can be omitted if the leader identity does not change between instances.

This does not interfere with the properties of Paxos because any replica at any time can still try to become a leader by broadcasting a propose message with a higher round/ballot number

Master leases

Reads of the data structure require executing Paxos

Read operations cannot be served out of the master's copy of the data structure because other replicas may have elected another master and modified the data structure without notifying the old master

Since read operations comprise a large fraction of all operations, serializing reads through Paxos is expensive

Master leases mechanism of Paxos solves this problem

Leader is chosen to serve until lease expires. Replicas refuse to process messages from another master while lease holds

Master leases ...

If the master has the lease, it is guaranteed that other replicas cannot successfully submit values to Paxos

Thus a master with the lease has up-to-date information in its local data structure which can be used to serve a read operation purely locally

By making the master attempt to renew its lease before it expires we can ensure that a master has a lease most of the time

If the master fails or gets stuck in a minority partition, when the lease expires another replica can become a master and continue execution as per Paxos rules