Distributed Systems
[Fall 2012]

Lec 18: Agreement in Distributed Systems: Paxos

Slide acks: Jinyang Li

Last Times: Agreement/Commitment

- 2PC: simple, safe, but blocking commitment protocol
- 3PC: unsafe, but live commitment protocol

- Paxos: safe and mostly live agreement protocol
  - Agreement is somewhat different from commitment in semantic
    - Commitment = everyone agrees
    - Agreement = sufficient agree so that I can always tell what the outcome was

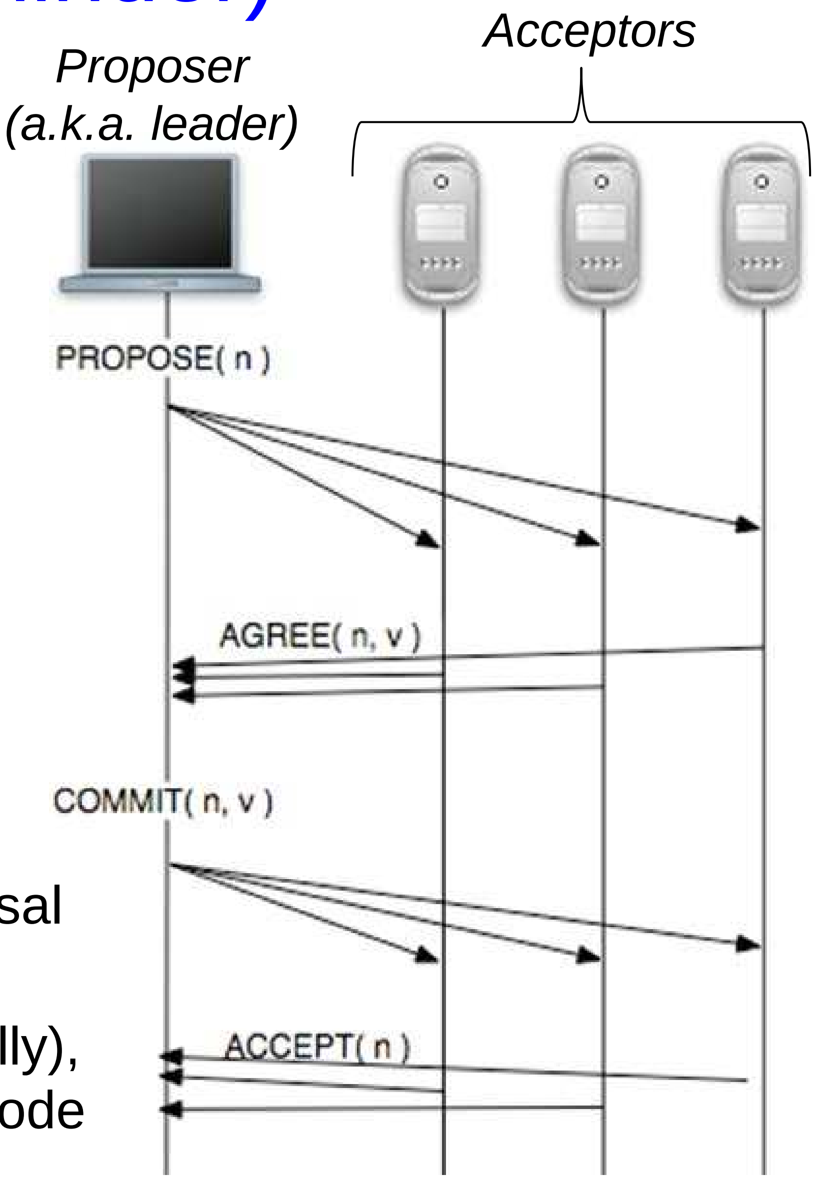
- NLP impossibility result: you can't have both liveness and safety in an asynchronous network

- Today:
  - Continue Paxos
  - One application of it in a real system: Google's Chubby

Paxos Functioning (Reminder)

- Similar to 2PC, but also different…
- Two differentiating mechanisms:

1. Proposal ordering and acceptance protocol
   - Acceptors accept only proposals of monotonic sequence numbers
   - Once a value is accepted, acceptors ask for that value to be preserved
2. Majorities
   - (half+1) need to agree to accept proposal
   - Guarantees that if two proposals are accepted (simultaneously or sequentially), there will be at least one overlapping node to arbitrate them and make sure their values are the same

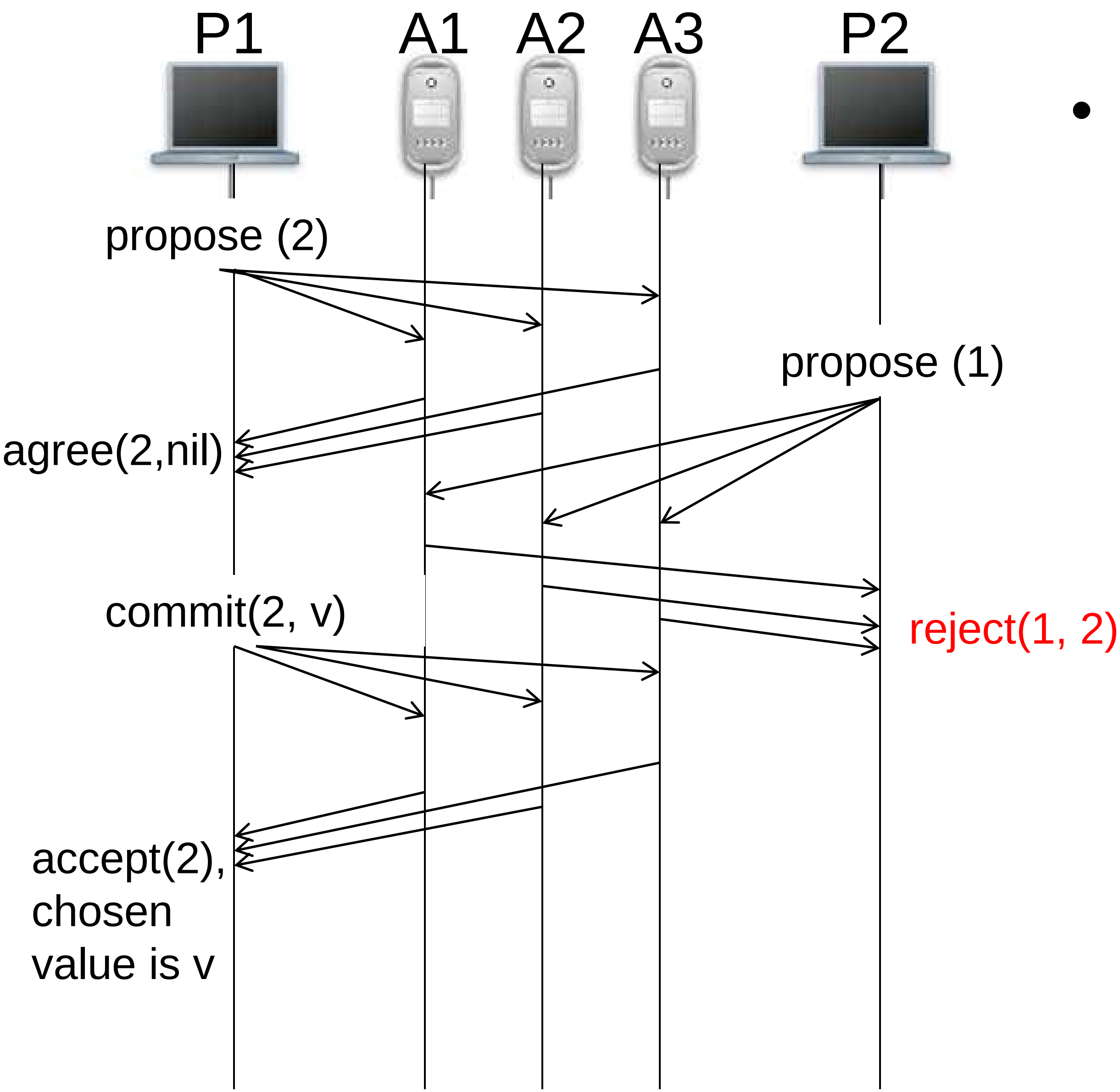


One successful round (trial) of Paxos

Outline of Paxos Presentation

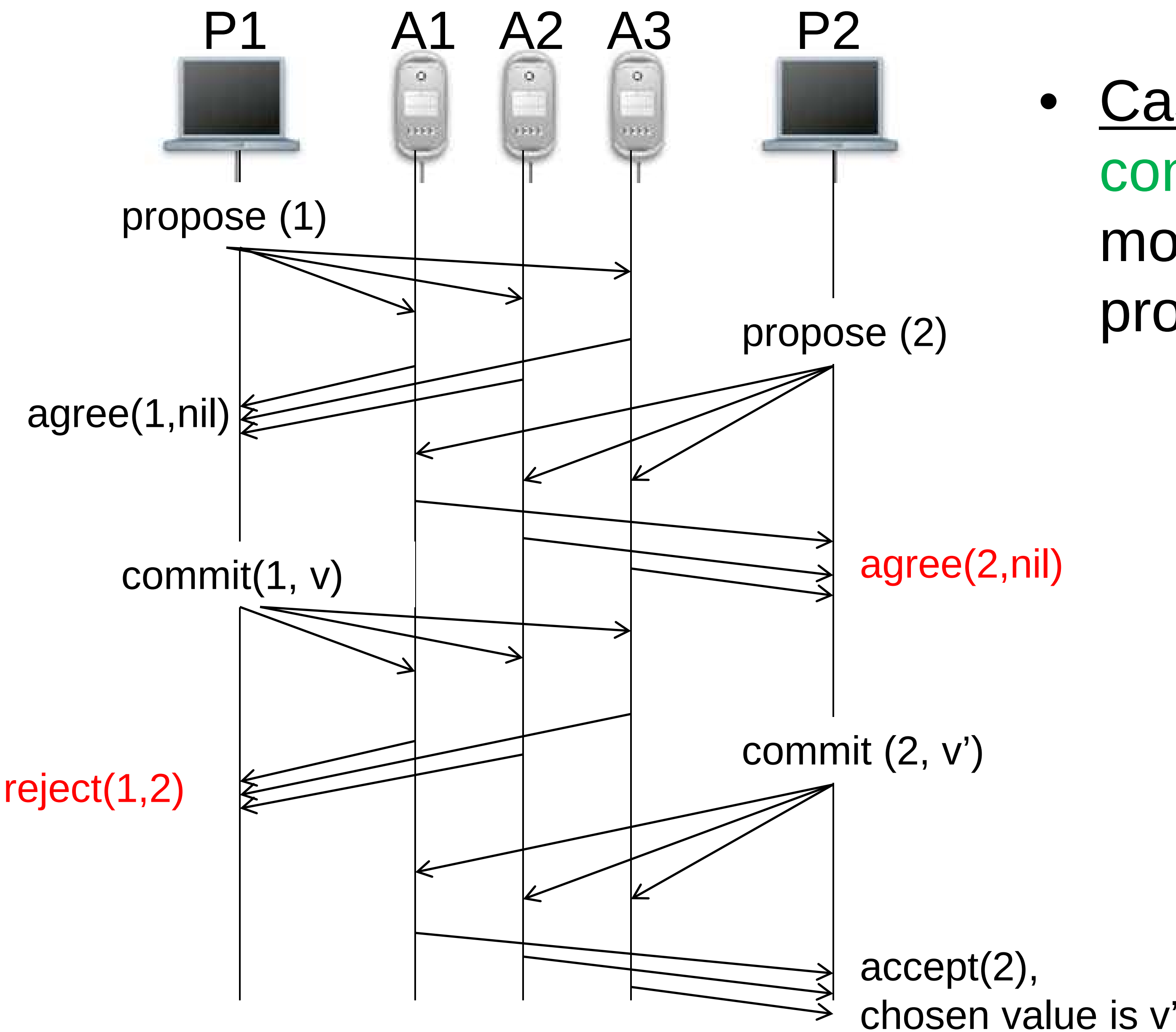
- High-level overview (last time)
- Detailed operation (today)

Paxos with Multiple Proposers



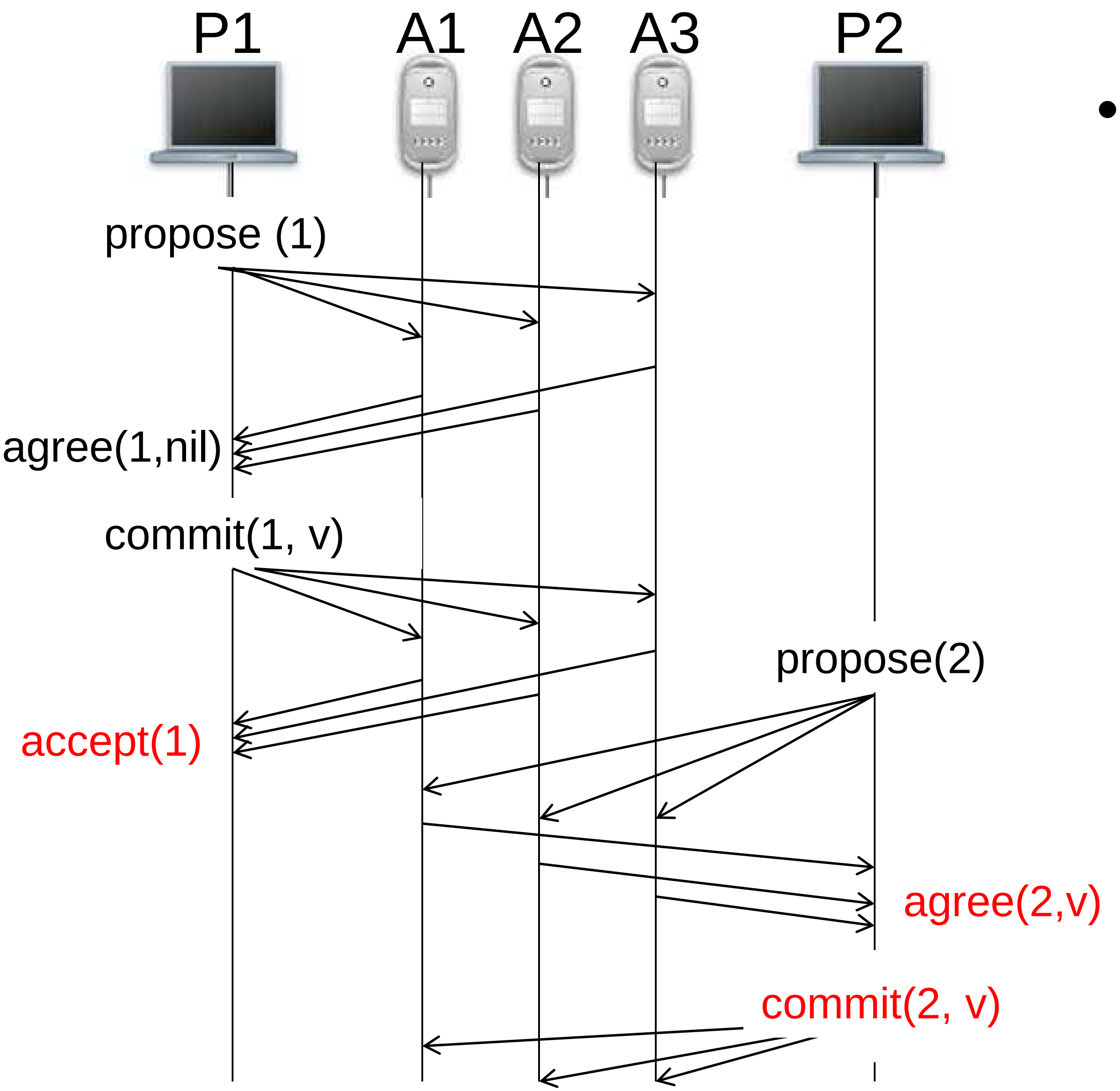
- Case 1: proposals with lower sequence numbers

Paxos with Multiple Proposers



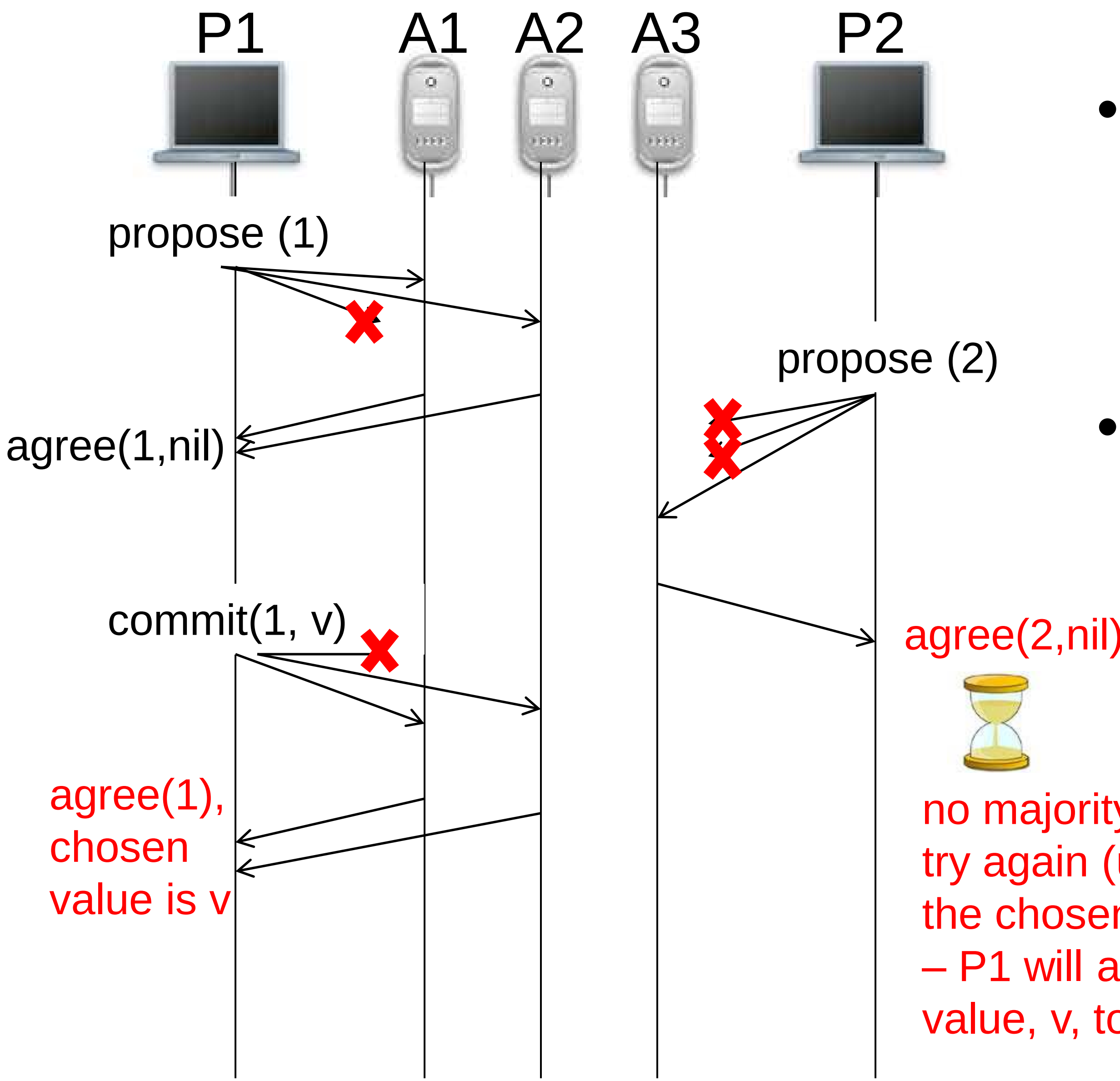
- Case 2: concurrent monotonic proposals

Paxos with Multiple Proposers



- Case 3: sequential monotonic proposals

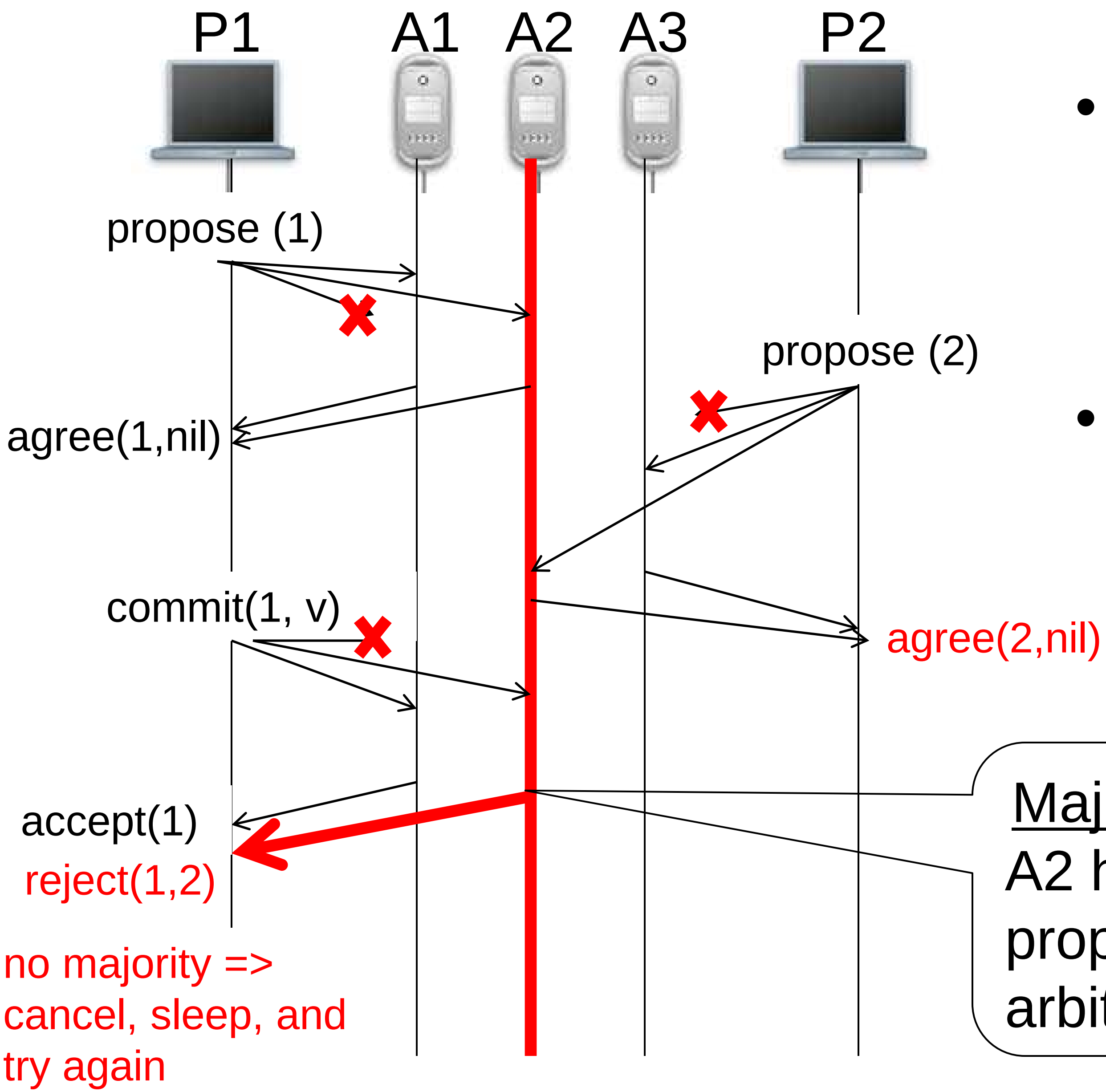
Paxos with Failures – Why It works?



- Back to Case 2: concurrent monotonic proposals
- Add network partitions

no majority => cancel, sleep, and try again (unless you hear about the chosen value in the meantime – P1 will announce the chosen value, v, to everyone)

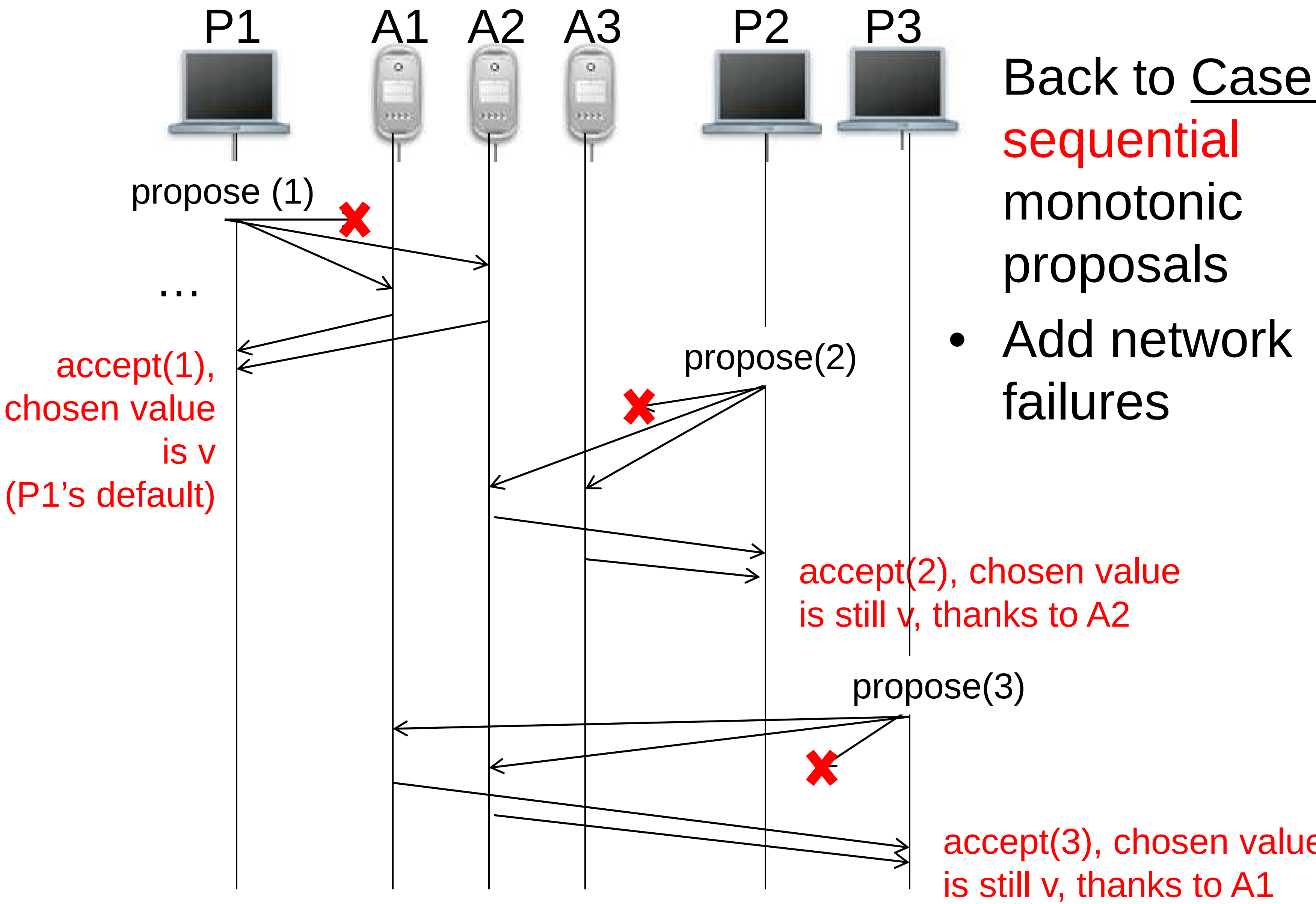
Paxos with Failures – Why It works?



- Back to Case 2: concurrent monotonic proposals
- Add network partitions

Majorities overlap:
A2 has seen both proposals and can arbitrate them

Paxos with Failures – Why It Works?



Back to Case 3: sequential monotonic proposals

- Add network failures

Paxos Safety

- Stems from:
  - At most one value can be chosen by simultaneous proposals
  - After a value is chosen, any subsequent proposal will preserve it
  - Any chosen value is one of the proposed values

- Nodes learn about chosen values:
  - From leader, which announces the chosen value after a successful round
  - Or by becoming leaders themselves, making a new proposal, and learning about what the chosen value was in this way

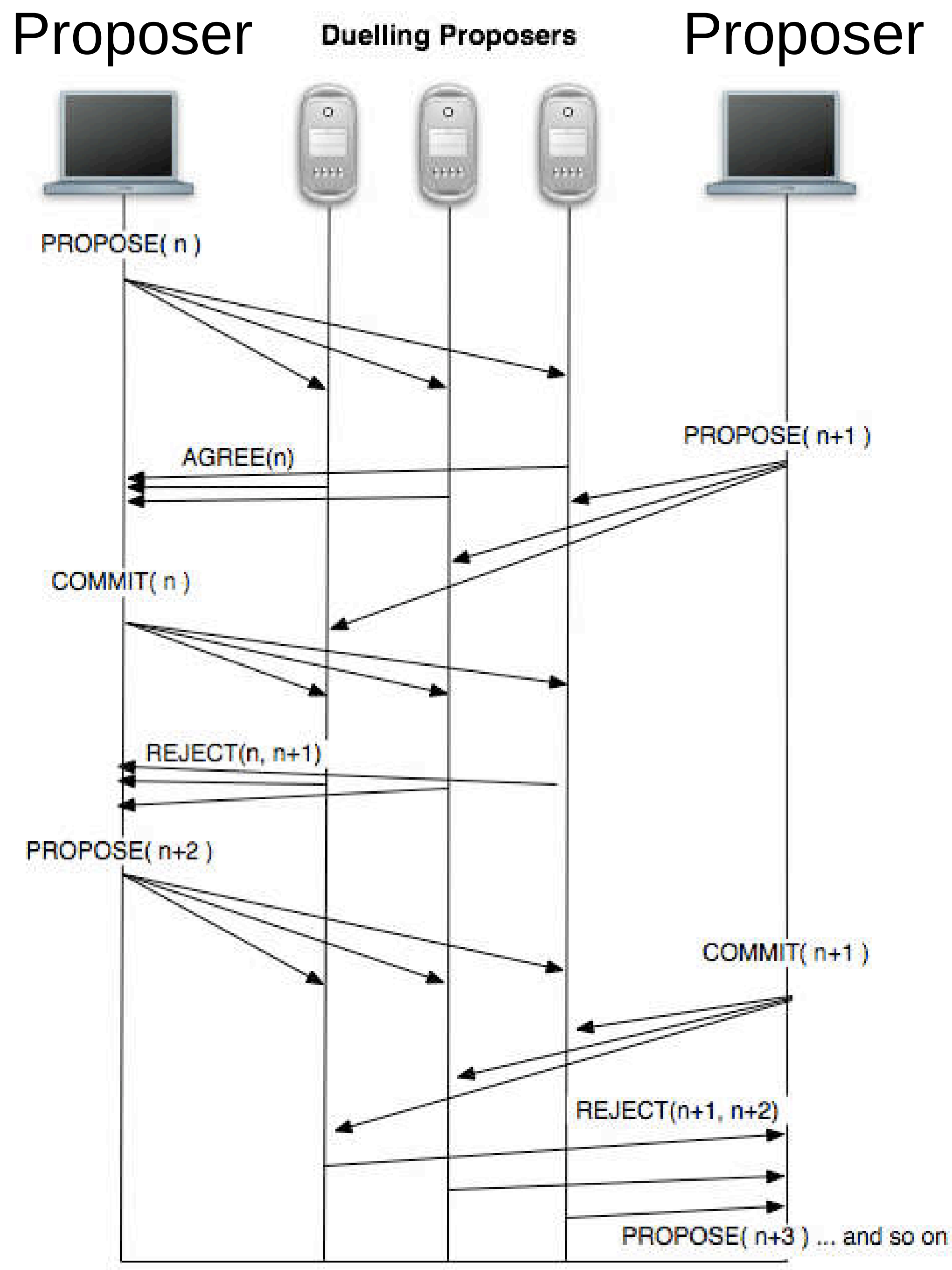
Paxos Liveness

- A Paxos run consists of one or more trials (rounds) run by different nodes
- If one leader dies, another one times out and offers be a leader
- A Paxos run is successful if at least a majority of the nodes is up and accepts the proposal

- But, there are degenerate cases where Paxos just doesn’t finish

Paxos May Not Terminate

- For example, if two or more proposers race to propose new values, they might step on each other toes all the time
  - This is a liveness exception
- With randomness, this occurs exceedingly rarely



Algorithm: Node State

- Each node maintains:
  - n_a, v_a: highest proposal # accepted and its corresponding accepted value
  - n_h: highest proposal # seen
  - my_n: my proposal # in the current Paxos round

Algorithm: Phase 1

- Phase 1 (Propose)
  - A node decides to be proposer (a.k.a. leader)
  - Leader chooses $my_n > n_h$
  - Leader sends <propose, my_n> to all nodes
  - Upon receiving <propose, n>

    If $n < n_h$

      reply <reject, n_h>

    Else

      $n_h = n$

      reply <agree, n_a,v_a>

This node will never accept any proposal lower than n in the future

Algorithm: Phase 2

- Phase 2 (Commit):
  - If proposer gets “agree” from a majority

    V = value corresponding to the highest n_a received

    If V == null, then proposer can pick any V

    Send <commit, my$_n$, V> to all nodes
  - If leader fails to get majority prepare-ok
    - Delay and restart Paxos
  - Upon receiving <accept, n, V>

    If n < n_h

    reply with <reject, n_h>

    else

    n_a = n; v_a = V; n_h = n

    reply with <accept>

Algorithm: Phase 3

- Phase 3 (Decide)
  - If leader gets accept-ok from a majority
    - Send <decide, va> to all nodes
  - If leader fails to get accept-ok from a majority
    - Delay and restart Paxos

Exam FAQ

- When is the value V chosen?
  1. When leader receives a majority prepare-ok and proposes V
  2. When a majority nodes accept V
  3. When the leader receives a majority accept-ok for value V

Exam FAQ

- 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?

Exam FAQ

- What if leader fails while sending accept?
- What if a node fails after receiving accept?
  - If it doesn’t restart …
  - If it reboots …
- What if a node fails after sending prepare-ok?
  - If it reboots …

Using Paxos

- As we said before, Paxos can be used for lots of things
  - Distributed lock service
  - Choose master/primary in a master-slave/primary-secondary system
  - Choose which operation to perform next
  - …
- It can used in conjunction with 2PC to implement ACID transactions
  - For exam: think about how it would be used, what problems it would solve, etc.
- Next time: Chubby – Google's distributed lock service