CS 525 Advanced Distributed Systems Spring 2015

Indranil Gupta (Indy)

Lecture 8

Paxos

February 12, 2015

Consensus Problem

- Every process contributes a value
- Each process *decides* a value
  - Decision once made can't be changed
- *Goal is to have all processes decide same value*
- If everyone votes V, decision is V

- Consensus impossible to solve in asynchronous systems (FLP result)
- But important since it maps to many important distributed computing problems
- Um, can't we just solve consensus?

Yes we can!

- Paxos algorithm
  - Most popular “consensus-solving” algorithm
  - Does not solve consensus problem (which would be impossible, because we already proved that)
  - But provides <u>safety</u> and <u>eventual liveness</u>
  - A lot of systems use it
    - Zookeeper (Yahoo!), Google Chubby, and many other companies
- Paxos invented by? (take a guess)

Yes we can!

- Paxos invented by Leslie Lamport

- Paxos provides safety and eventual liveness
  - Safety: Consensus is not violated
  - Eventual Liveness: If things go well sometime in the future (messages, failures, etc.), there is a good chance consensus will be reached. But there is no guarantee.

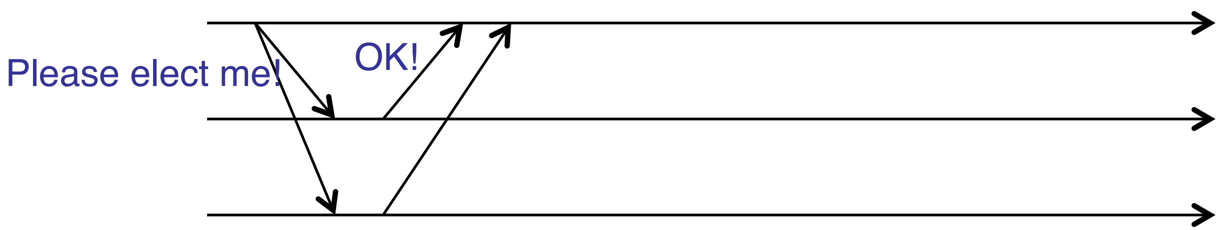
Political Science 101, i.e., Paxos Groked

- Paxos has rounds; each round has a unique ballot id
- Rounds are asynchronous
  - Time synchronization not required
  - If you're in round *j* and hear a message from round *j+1*, abort everything and move over to round *j+1*
  - Use timeouts; may be pessimistic
- Each round itself broken into phases (which are also asynchronous)
  - Phase 1: A leader is elected (Election)
  - Phase 2: Leader proposes a value, processes ack (Bill)
  - Phase 3: Leader multicasts final value (Law)

Slide ideas borrow from Jeff Chase's material (Duke U.)

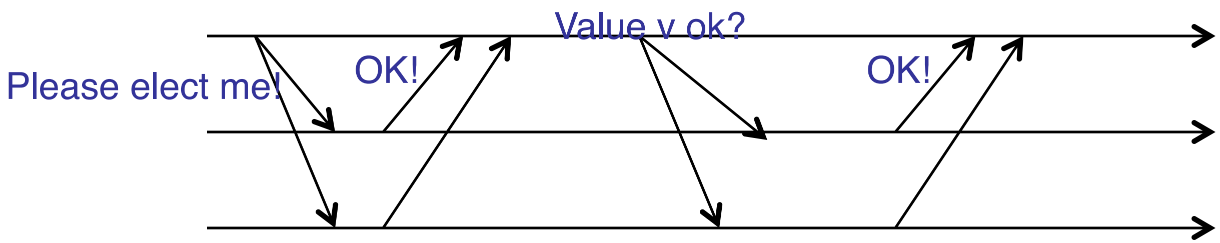
Phase 1 – Election

- Potential leader chooses a unique ballot id, higher than seen anything so far
- Sends to all processes
- Processes wait, respond once to highest ballot id
  – If potential leader sees a higher ballot id, it can't be a leader
  – Paxos tolerant to multiple leaders, but we'll only discuss 1 leader case
  – Processes also log received ballot ID on disk
- If a process has in a previous round decided on a value v', it includes value v' in its response
- If majority (i.e., quorum) respond OK then you are the leader
  – If no one has majority, start new round
- (If things go right) A round cannot have two leaders (why?)



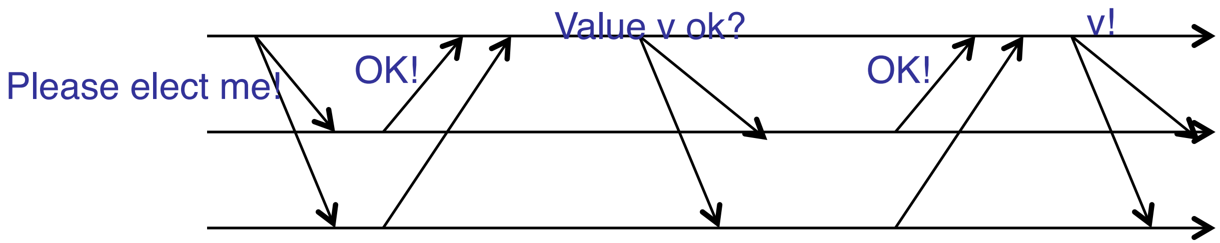
Phase 2 – Proposal (Bill)

- Leader sends proposed value v to all
  - use v=v' if some process already decided in a previous round and sent you its decided value v'
- Recipient logs on disk; responds OK



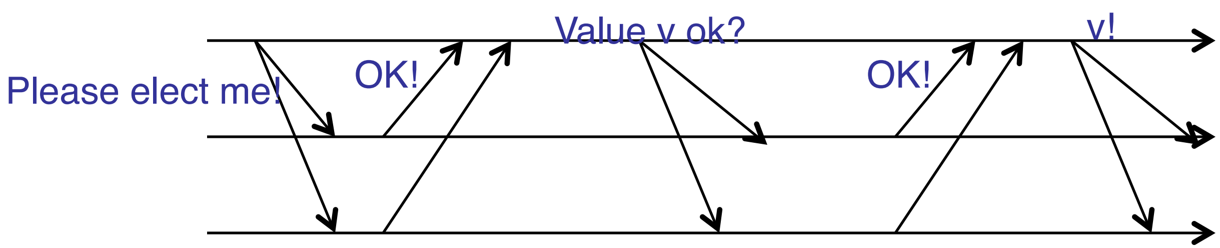
Phase 3 – Decision (Law)

- If leader hears a majority of OKs, it lets everyone know of the decision
- Recipients receive decision, log it on disk



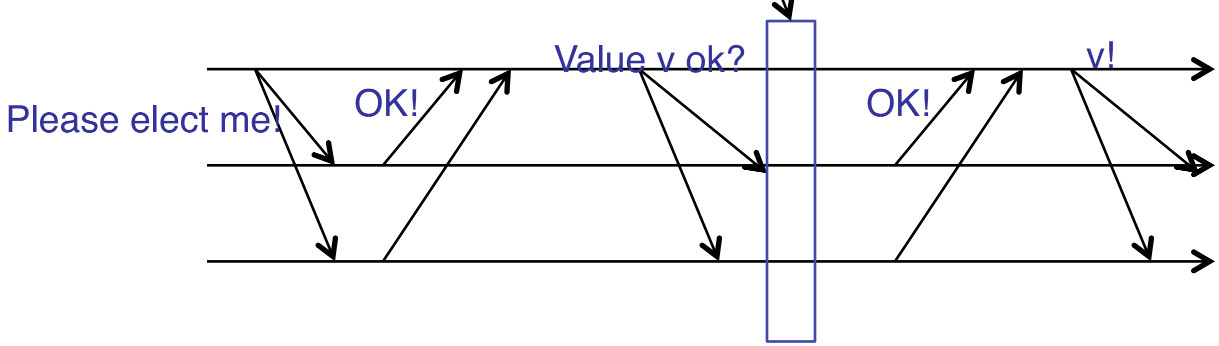
Which is the point of no-return?

- That is, when is consensus reached in the system



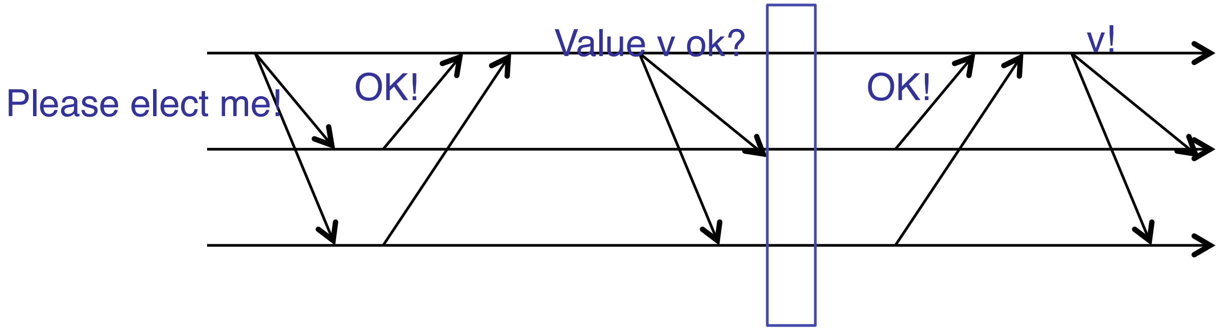
Which is the point of no-return?

- If/when a majority of processes hear proposed value and accept it (i.e., are about to/have respond(ed) with an OK!)
- Processes *may not know it yet*, but a decision has been made for the group
  - Even leader does not know it yet
- What if leader fails after that?
  - Keep having rounds until some round completes



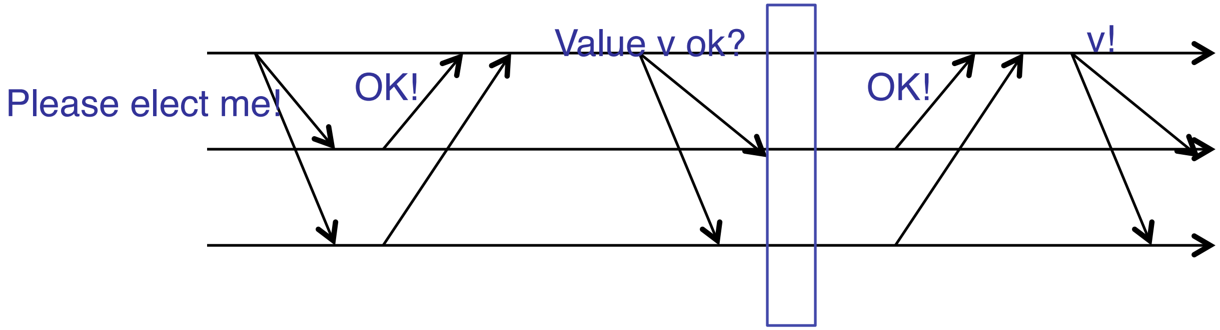
Safety

- If some round has a majority (i.e., quorum) hearing proposed value v' and accepting it, then subsequently at each round either: 1) the round chooses v' as decision or 2) the round fails
- Proof:
  - Potential leader waits for majority of OKs in Phase 1
  - At least one will contain v' (because two majorities or quorums always intersect)
  - It will choose to send out v' in Phase 2
- Success requires a majority, and any two majority sets intersect



What could go wrong?

- Process fails
  - Majority does not include it
  - When process restarts, it uses log to retrieve a past decision (if any) and past-seen ballot ids. Tries to know of past decisions.
- Leader fails
  - Start another round
- Messages dropped
  - If too flaky, just start another round
- Note that anyone can start a round any time
- Protocol may never end – tough luck, buddy!
  - Impossibility result not violated
  - If things go well sometime in the future, consensus reached



What could go wrong?

- A lot more!

- This is a highly simplified view of Paxos.
- See Lamport's original paper: http://research.microsoft.com/en-us/um/people/lamport/pubs/paxos-simple.pdf