What can we do about FLP?

• Consider easier problems

• Randomization

• Consider easier models (partial synchrony)

• Agreement, total order bcast, and replication possible in psync or async with randomization
  – Single-value broadcast still impossible
Partial Synchrony

• (Intuitively) The network is sometimes asynchronous and sometimes synchronous
  – Maintain safety during asynchronous periods
  – Achieve liveness during synchronous periods
Partial Synchrony

• (Formally) There exists an **unknown** Global Standardization Time (GST) after which the network becomes synchronous
  – Forever synchronous after GST???
    • Hope to capture “sufficiently long sync periods”
  – Unknown to whom?
    • Can be viewed as a game between protocol designer and the adversary
Psync Agreement Fault Bound

• Crash: $f < \frac{n}{2}$
  
  – Proof: Two groups $|P| \leq f$ and $|Q| \leq f$
  
  – Scenario I:

  – Scenario II:

  – Scenario III:
Psync Agreement Fault Bound

• Crash: $f < \frac{n}{2}$
  
  – Proof: Two groups $|P| \leq f$ and $|Q| \leq f$
  
  – Scenario I: P non-faulty & receive $v$, Q crash
    • P eventually commit $v$ due to validity
  
  – Scenario II: Q non-faulty & receive $v'$, P crash
    • Q eventually commit $v'$ due to validity
  
  – Scenario III: Both non-faulty, P receive $v$, Q receive $v'$
    GST sufficiently large $\rightarrow$ Both think the other crashed
    • P commit $v$, Q commit $v'$
Paxos


- Partial synchronous

- Tolerate $f < \frac{n}{2}$ crash faults (best possible)

- First practical consensus protocol, likely the most widely known/used (before Bitcoin)
Paxos

• A (state machine) replication protocol
  – Agree on a sequence of values
    • We will again start with a single value
  – Values come from clients, validity is “external”

• Partial synchrony with alternating periods
  – Delay bound $\Delta$ holds during synchronous periods
  – Maintain safety during async, live during sync
    • We will use the unknown GST model
Paxos Protocol

• Leader sends \((\text{propose, } x, k)\)
  
  – \(x\) is the proposed value
  
  – \(k\) is a rank/ballot/view/iteration number
Paxos Protocol

• Leader sends \((\text{propose, x, k})\)

• Upon receiving the leader’s proposal, others send \((\text{vote, x, k})\) back to the leader
Paxos Protocol

- Leader \((\text{propose, } x, k)\); Others \((\text{vote, } x, k)\)

- Leader waits for \(n-f\) votes, sends \((\text{success, } x, k)\)

- Upon receiving \((\text{success, } x, k)\), others commit \(x\)
Paxos Protocol

• Leader \((\text{propose, } x, \ k)\); Others \((\text{vote, } x, \ k)\)

• Leader: \((\text{success, } x, \ k)\); Others: commit \(x\)

• After a time-out, repeat under the next leader with \(k\) incremented
Paxos Liveness

• Rotating leaders tolerate faulty leaders
• Non-faulty leader after GST gives liveness
Paxos Safety

• Safety under one leader is obvious
  – Because leader is benign

• Safety across leaders is the challenge
Safety Across Leaders

• New leader must find out what happened
• If one replica commits $x$, we want many replicas to “recommend” $x$ to new leaders
  – Naturally, recommend the value one has voted
Paxos Protocol

• Leader (replica k % n) sends (new-view, k)
• Others reply with (status, k, x_{lck}, k_{lck})
• Leader (propose, x, k)
• Others (vote, x, k) and lock (x, k)
• Leader (success, x, k); Others commit x
Safety Across Views

• One replica commits \( x \)
  \( \rightarrow \) n-f replicas voted and locked \( x \)
  \( \rightarrow \) Each future leader collects locks from n-f replicas, at least one is locked on \( x \)
  \( \rightarrow \) Due to quorum intersection
  \( \rightarrow \) Each future leader re-proposes \( x \)
  \( \rightarrow \) No other value can ever be proposed, voted or committed

Any issues in this proof?
Safety Across Views

• One replica commits \( x \)

\( \rightarrow \) n-f replicas voted and locked \( x \)

\( \rightarrow \) Each future leader collects locks from n-f replicas, at least one is locked on \( x \)

\( \rightarrow \) Due to quorum intersection

\( \rightarrow \) Each future leader re-proposes \( x \)

What if some other replica reports a different locked value?
Paxos Locks

• Can replicas lock on different values?
  – and one of the value is committed?

• Need a tie-breaking mechanism on locks that favors the committed value (if any)
Paxos Protocol

- Leader (replica k % n) sends *(new-view, k)*
- Others reply with *(status, k, x_{lck}, k_{lck})*
- Leader *(propose, x, k)*

- Others *(vote, x, k)* and lock *(x, k)*
- Leader *(success, x, k)*; Others commit x
Paxos Protocol

- Leader (replica k % n) sends (new-view, k)
- Others reply with (status, k, x_{lck}, k_{lck})
- Leader (propose, x, k) where x is the highest locked value among the f+1 status
- Others (vote, x, k) and lock (x, k)
- Leader (success, x, k); Others commit x
Single-slot Paxos Full Protocol

• Upon detecting a lack of progress, replica (k % n) sends (new-view, k)

• Upon receiving (new-view, k), a replica enters view k and replies with (status, k, x_{\text{lck}}, k_{\text{lck}})

• Upon receiving n-f status, leader sends (propose, x, k) where x is the highest locked value. If none has locked, the leader can choose x freely.

• Upon receiving (propose, x, k), a replica sends (vote, k) and locks (x, k) if it has not entered a higher view

• Upon receiving n-f (vote, k), leader sends (success, x)

• Upon receiving (success, x), a replica commits x
Safety Across Views

• One replica commits x in view k
  → n-f replicas voted and locked (x, k)
  → Leader k+1 collects locks from n-f replicas, at least one (x, k), which is the highest
  → Leader k+1 re-proposes x. No other value can be voted or locked in view k+1
  → Leader k+2 collects locks from n-f replicas, at least one (x, k), still the highest
  → Leader k+2 re-proposes x. No other value can be voted or locked in view k+2
  → ……
Paxos Locks

- Tie-breaking favors lock from the latest view
- Why?

- Lock protects a potential commit
- Value x committed $\rightarrow$ no other higher lock ever in all subsequent views
- Hence, favoring a higher lock is always safe
  - Safe to “unlock” x if there is a higher lock on x’
Quiz

• What will go wrong if … ?
  – vote for leader k even after quitting view k
  – leader waits for only f status
  – leader does not repropose highest lock
  – the network is async

• When does Paxos become univalent?

• If $n > 2f+1$, can we wait for less than $n-f$ msgs?
Multi-slot Paxos

• All messages are tagged with a slot number $s$ (position in the ledger)
  – (propose/vote/success, $s$, $x$, $k$)

• Steady state vs. view-change
  – Repeat propose + vote + success for each slot in steady state
  – Upon lack of progress, do view-change using new-view + status
Multi-slot Paxos

- During view-change, exchange information on what slots have been committed
  - New leader sends \((\text{new-view, k, s^*})\) where \(s^*\) is its last committed slot (or any format to convey this)
  - For slots committed by the follower but not the leader, send success msg to the leader
  - For slots committed by the leader but not the follower, request success msg from the leader
  - For slots committed by neither but locked by the follower, send \((\text{status, k, x_{lck}, k_{lck}, s})\) for all such \(s\)
- Leader updates its ledger, send requested success msgs, re-propose for locked slots, and propose new values for “fresh” slots
Multi-slot Paxos Efficiency

• During steady state (non-faulty leader and synchrony), 3 rounds and $3n$ msgs per decision
  – Isn’t there a $f+1$ round lower bound?

• View-change: 2 rounds and possibly many msgs
Paxos Summary

• Most widely known/used and first practical crash fault tolerant protocol
  – Replication, partial synchrony, f < n/2 crash
  – Leader-based, quorum intersection, lock ranking

• Original notation FYI:
  – new-view = prepare
  – status = promise
  – propose = accept
  – vote = accepted