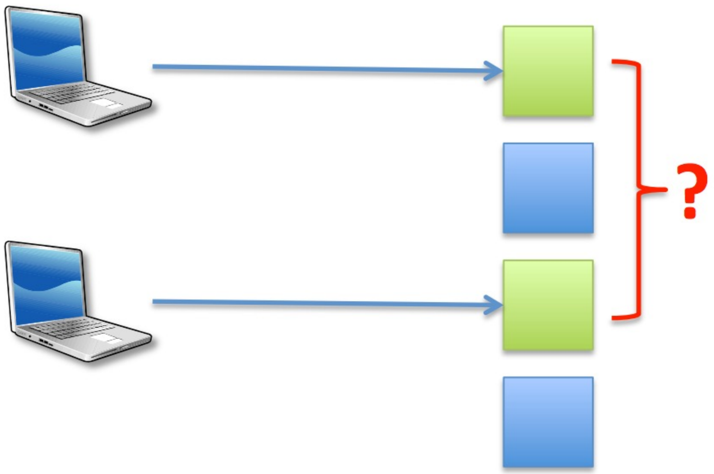
Implementing Consistency -- Paxos

Some slides from Michael Freedman

CAP Conjecture

- System can have two of:
  - C: Strong consistency
  - A: Availability
  - P: Tolerance to network partition
- 2PC: CA
- Consensus (Paxos/Raft): CP
- Eventual consistency: AP
- ACID/BASE

What do clients see?



- Distributed stores use replication
  - Fault tolerance and scalability
  - Does replication necessitate inconsistency?
    - Harder to program, confusing for clients

Problem

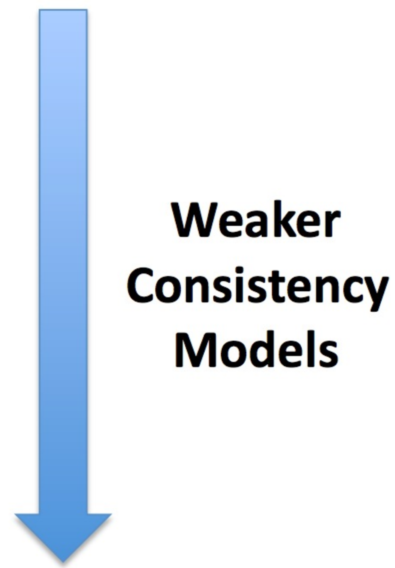
- How to reach consensus/data consistency in distributed system that can tolerate non-malicious failures?
- We saw some consistency models – how to implement them?

Another perspective

- Lock is the easiest way to manage concurrency
  - Mutex and semaphore.
  - Read and write locks.
- In distributed system:
  - No shared state
  - Failures
  - What can we do?

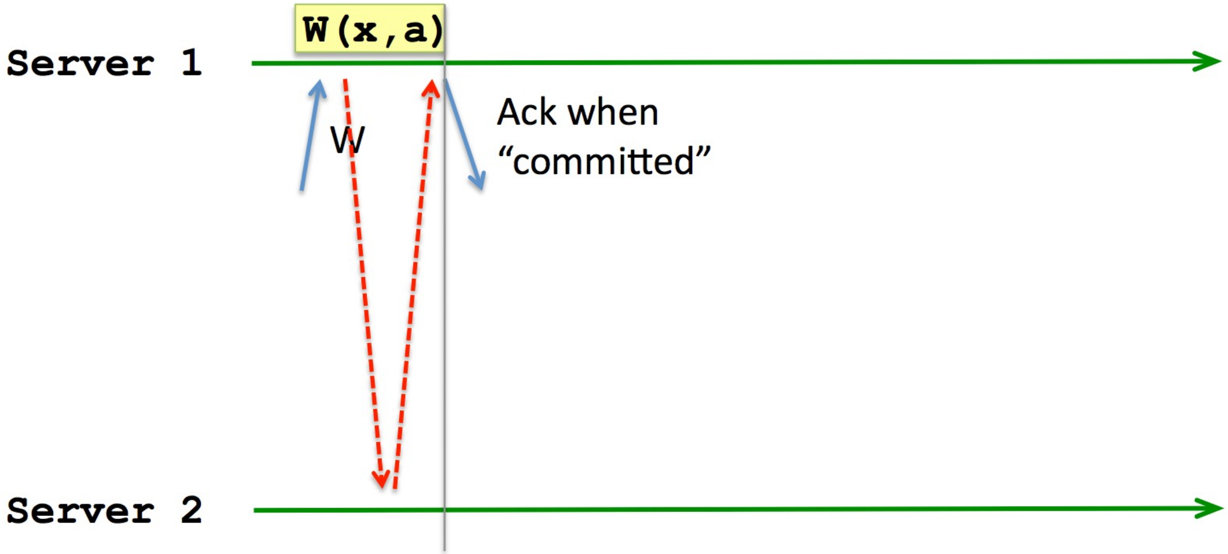
Recall, consistency models

- Strict
- Strong (Linearizability)
- Sequential
- Causal
- Eventual



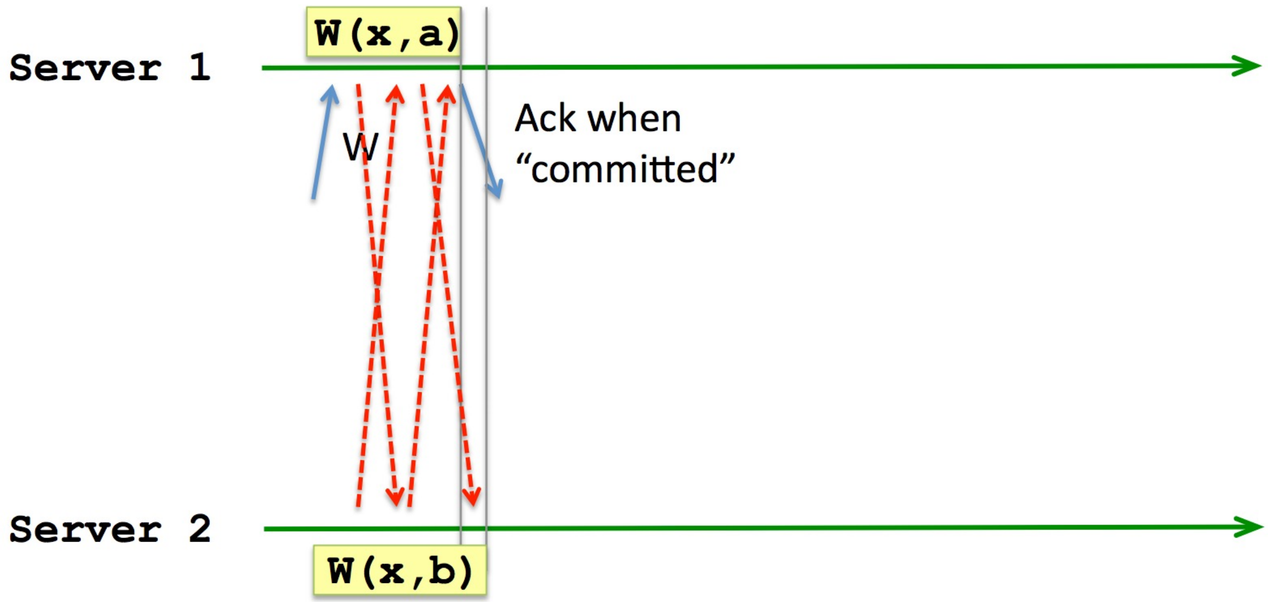
These models describes when and how different nodes in a distributed system / network view the order of messages / operations

Implementing Linearizability



- If OP must appear everywhere after some time (the conceptual "timestamp" requirement) ⇒ "all" locations must locally commit op before server acknowledges op as committed
- Implication: Linearizability and "low" latency mutually exclusive
  - e.g., might involve wide-area writes

Implementing Linearizability



- Algorithm not quite as simple as just copying to other server before replying with ACK: Recall that all must agree on ordering
  - Both see either a → b or b → a , but not mixed
  - Both a and b appear everywhere as soon as committed

Ok, what to do?

- We want consistency and availability
- Two options

1. Master Replica Model
   - All operations and ordering happen on a single master
   - Replicates to secondary copies
2. Multi-master model
   - Read/write anywhere
   - Replicas order and replicate content before returning

Coordination protocols

• Marriage ceremony	Do you? I do. Do you? I do.	Prepare
	I now pronounce…	Commit
• Theater	Ready on the set? Ready!	
	Action!	
• Contract law	Offer Signature	
	Deal / lawsuit	

Two phase commit (2PC)

What about failures?

- If one or more acceptors fail:
  - Still ensure linearizability if |R|+|W|>N+F
    - Read and write quoroms of acceptors overlap in at least one non-failed node
- Leader fails?
  - Bye bye ☺: system no longer live
- Pick a new leader?
  - How do we agree?
  - Need to make sure that group is know

Consensus protocol: Requirements

- Safety
  - One value accepted
  - If a server learns a value has been chosen, it has
- Liveness (some timeliness requirements)
  - Some proposed value is eventually chosen
  - Each node eventually learns it
- Fault tolerance
  - If <= F faults in a window, consensus reached eventually
  - Liveness not guaranteed: if >F no consensus

Given desired F, what is N?

- Crash faults need 2F+1 processes
- Byzantine faults (malicious) need 3F+1 processes
  - i.e., some replicas are trying to intentionally lie to prevent consensus or change the value

Why is agreement hard?

- What if more than one node is leader?
- What if network is partitioned?
- What if leader crashes in middle?
- What if new leader proposes different values than those committed?
- Network is unpredictable, delays are unbounded

Paxos players

- Proposers
  - Active: put forth values to be chosen
  - Handle client requests
- Acceptors
  - Passive: respond to messages
    - Responses are basically votes to reach consensus
  - Store chosen value, need to know which
- Each Paxos server can be both

Strawman solution I

- One node X designated as acceptor
  - Each proposer sends its value to X
  - X decides one value and announces it
  - Problem?
    - Failure of acceptor halts decision
    - Breaks fault-tolerance requirement!

Strawman II

- Each proposer (leader) proposes to all acceptors (replicas)
  - Acceptor accepts first proposal, rejects rest
  - Acks proposer
  - If leader receives acks from majority, picks that value and sends it to replicas
  - Problems?
    - Multiple proposals – may not get a majority
    - What if leader dies before chosing value?

Paxos!

- Widely used family of algorithms for asynchronous consensus
- Due to Leslie Lamport
- Basic approach
  - One or more nodes decide to act like a leader
  - Proposes a value, tries to get it accepted
  - Announces value if accepted

Paxos has three phases

Phase 1 (Prepare)

- Node decides to become leader
  - Chooses $t_{my} > t_{max}$
  - Sends *<prepare,* t_{my}> to all nodes
- Acceptor upon receiving *<prep, t>*

  If $t < t_{max}$

    reply <prep-reject>

  Else

    $t_{max} = t$

    reply <prep-ok, t_a, v_a>

Phase 2 (Accept)

- If leader gets *<prep-ok, t, v>* from majority
  - If v == null, leader picks v_{my}. Else v_{my} = v.
  - Send *<accept,* t_{my}, v_{my}> to all nodes
- If leader fails to get majority, delay, restart
- Upon *<accept, t, v>*

  If $t < t_{max}$

    reply with <accept-reject>

  Else

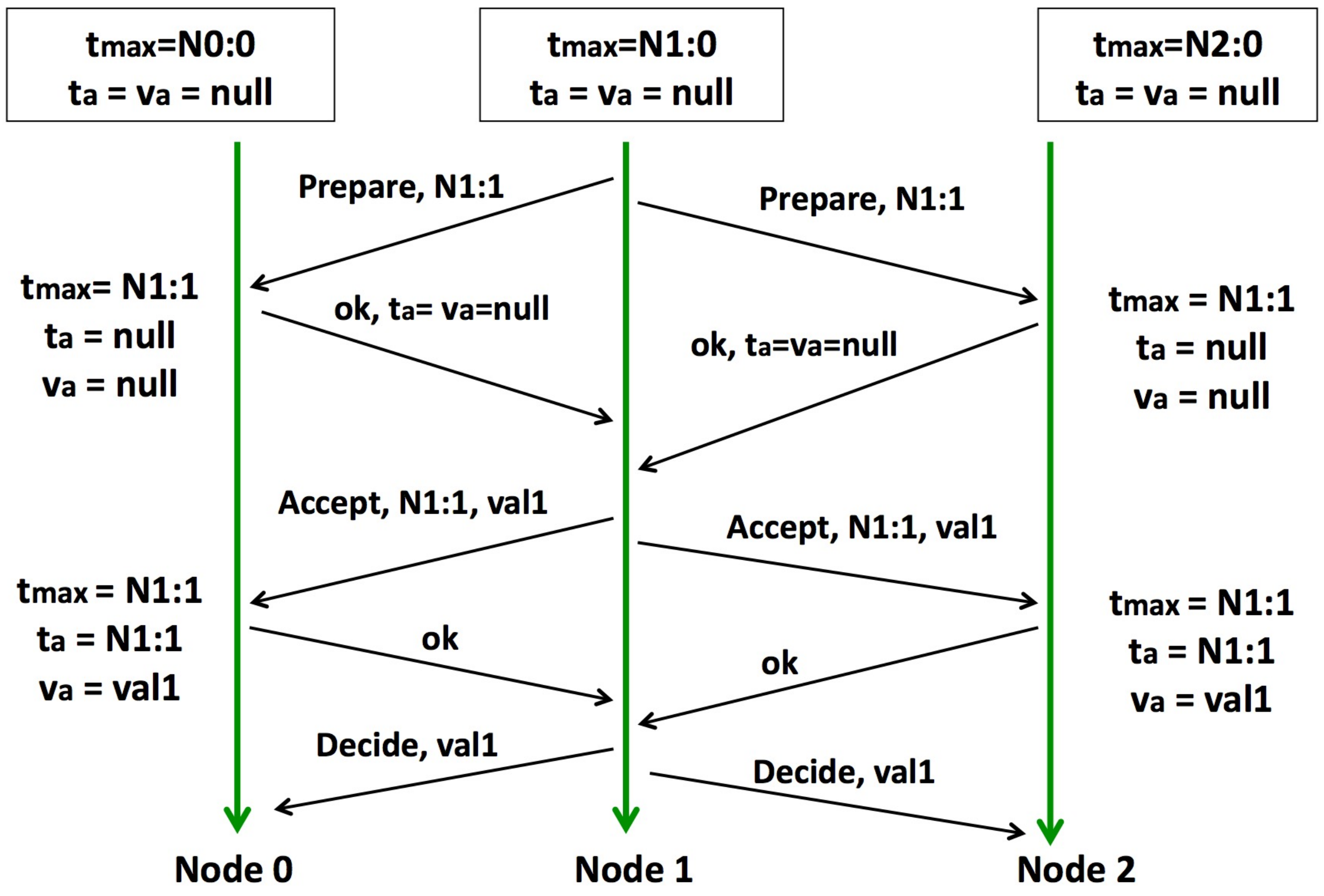
    $t_a = t$; $v_a = v$; $t_{max} = t$

    reply with <accept-ok>

Phase 3 (Decide)

- If leader gets *acc-ok* from majority
  - Send *<decide,* v_a> to all nodes
- If leader fails to get accept-ok from majority
  - Delay and restart

Example


tmax=N0:0
ta = va = null
tmax=N1:0
ta = va = null
tmax=N2:0
ta = va = null
Prepare, N1:1
Prepare, N1:1
tmax= N1:1
ta = null
va = null
ok, ta= va=null
ok, ta=va=null
tmax = N1:1
ta = null
va = null
Accept, N1:1, val1
Accept, N1:1, val1
tmax = N1:1
ta = N1:1
va = val1
ok
ok
tmax = N1:1
ta = N1:1
va = val1
Decide, val1
Decide, val1
Node 0
Node 1
Node 2

Paxos Properties

- Paxos is guaranteed safe.
  - Consensus is a stable property: once reached it is never violated; the agreed value is not changed.

Paxos Properties

- Paxos is not guaranteed live.
  - Consensus is reached if “a large enough subnetwork...is non-faulty for a long enough time.”
  - Otherwise Paxos might never terminate.

Combining Paxos and 2pc

- Use paxos for view-change
  - If anybody notices current master or one or more replicas unavailable
  - Propose view change to paxos to establish new group
  - Forms the new group for 2pc
- Use 2PC for actual data
  - Writes go to master for 2pc
  - Reads from any replica or master
- No liveness if majority of nodes from previous view unreachable
- What if a node comes back/joins?

Example system

- Apache zookeeper
- Used by a large number of Internet scale projects
  - Locking/barriers
  - Leader election
  - Consistency
  - …