Paxos: How it works

Jaideep Khandelwal

- Introduction
- Logical Clocks
- Paxos overview
- Multi Paxos
- Summary

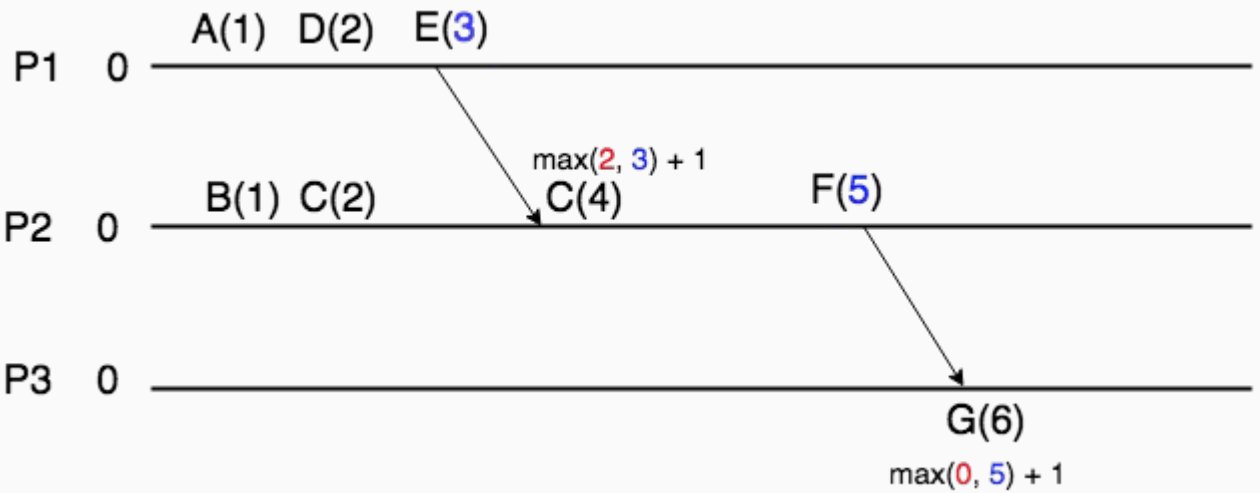
Before we get into it

- Invented by Leslie Lamport in 1989.
- Google(Big Data, Chubby), Apache Zookeeper and many more.
- When a group of processes try to attain consensus. Need to satisfy :
  - Termination - eventually processes should decide on a value.
  - Validity - If all process proposes a value, then all processes should decide same value.
  - Integrity - if one process decide one value, then it should be proposed by another process.
  - Agreement - All processes must agree on the same value.

Logical Clocks

- Logical ordering of events.
- Relation between events Happens-Before (->) . Also called as Causality.
- For a process P, events - A, B and C s.t A->B and B->C.
  - Thus, time(A) < time(B) and time(B) < time(C).
  - ∴ A -> B -> C (by Transitivity)
- Each process maintains a local counter initialized to 0.
- Increment local counter -
  - by 1 in case of send message/instruction execution
  - for receive message max(local counter, timestamp of send message) + 1 .

Logical Clocks



Paxos overview

Two phase Protocol:

In Phase 1:

Proposer - Sends a proposal number to Acceptors (Propose).
Acceptor - Sends a promise to Proposers (Promise).

In Phase 2:

Proposer - Sends a proposal number with value to be accepted to Acceptors(Accept).
Acceptor - Sends OK or reject value to Proposers(Accepted/Rejected).
Learners - Learns iff value is accepted.

A process can be any of the agents in an implementation.

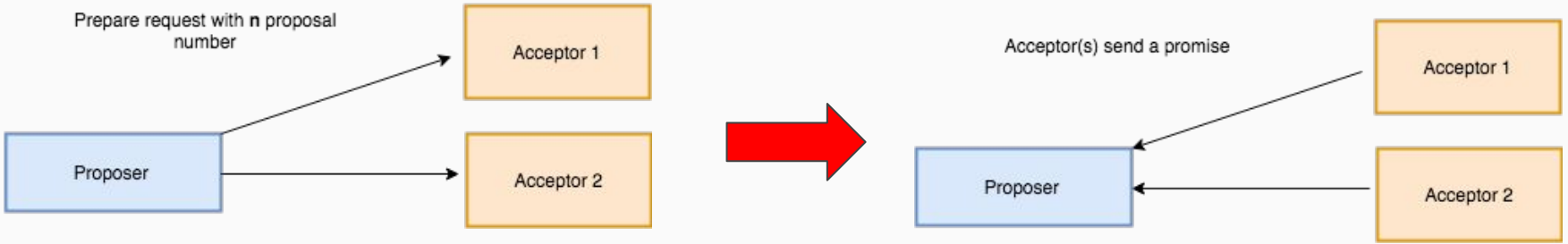
Assumptions:

- Asynchronous message delivery.
- Messages can be lost, duplicated, though correctness is maintained.
- Processes are non-Byzantine.

Phase 1

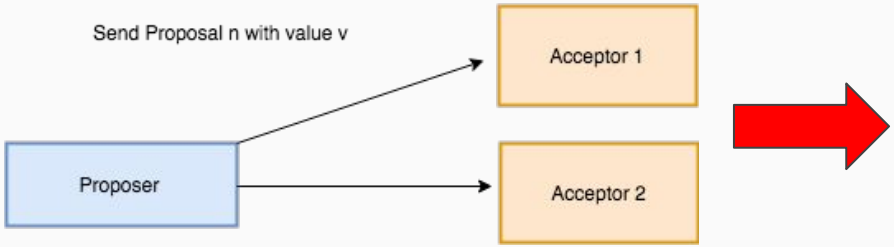
1a) Prepare(n) request to Acceptors

1b) Acceptors send a promise, i.e. not to accept proposal number < n

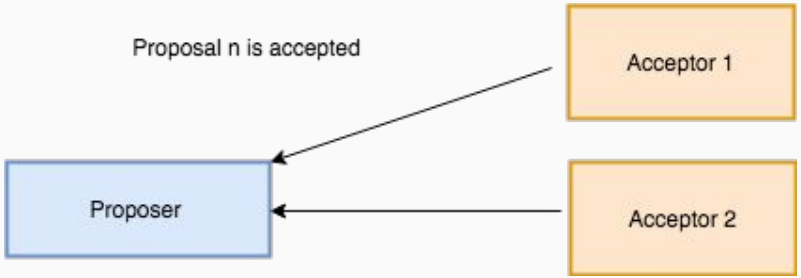


Phase 2

2a) On receiving Quorum for n, Proposer sends Accept(n,v) with value v.



2b) Proposal number n is highest, with Quorum, thus accepted

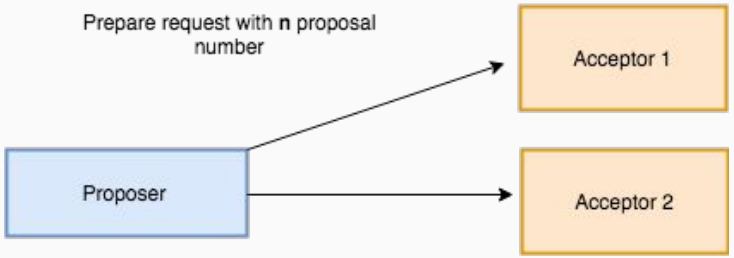


What if ?

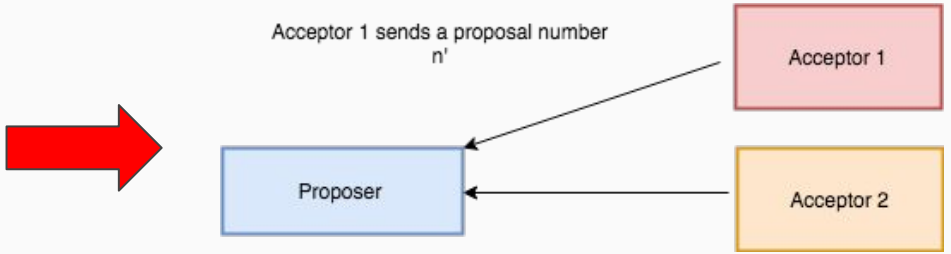
Any of the Acceptor(s) returns accepted value(s) instead of promise in Phase 1(b) ?

Phase 1 (Return with accepted value)

1a) Prepare(n) request to Acceptors

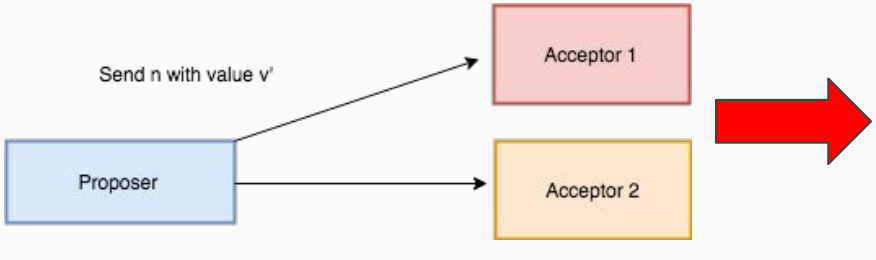


1b) Acceptor 1 sends n' with value v', which is already accepted.

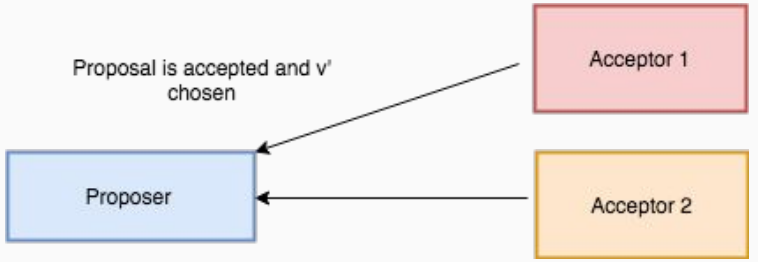


Phase 2 (Return with accepted value)

2a) Replace the value v with v’



2b) Proposer needs Quorum of acceptors to choose v.



What if ?

Any of the Acceptor(s) returns higher proposal number **n’** instead of promise in Phase 2(b) ?

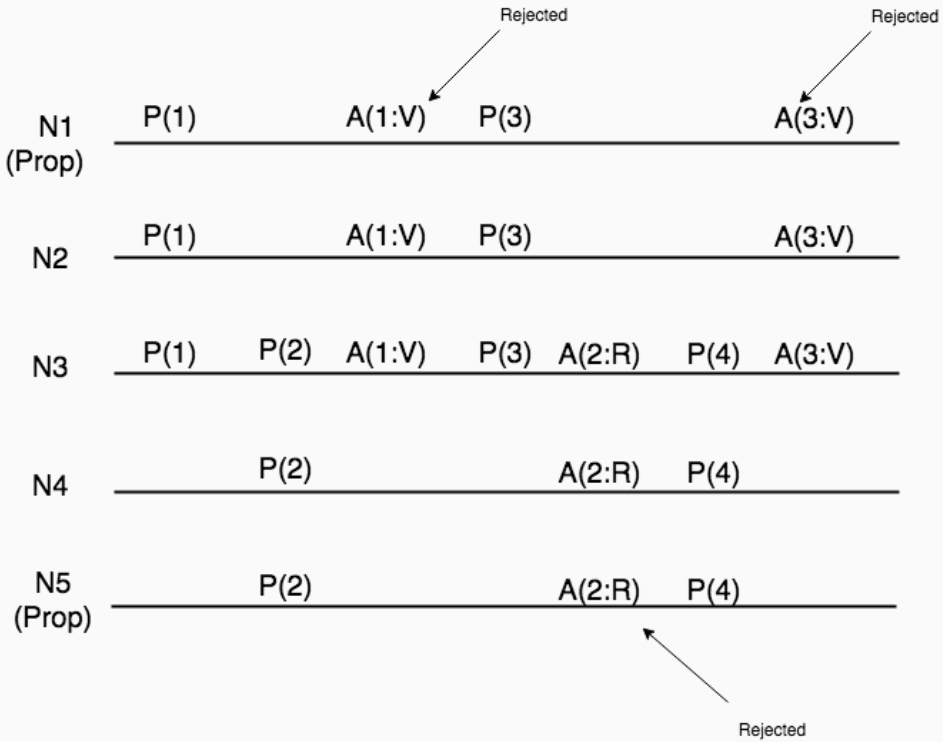
What if ?

Any of the Acceptor(s) returns higher proposal number **n'** instead of promise in Phase 2(b) ?

The Proposal is rejected and the Proposer re-starts with Prepare(**n''**) **(n'' > n').**

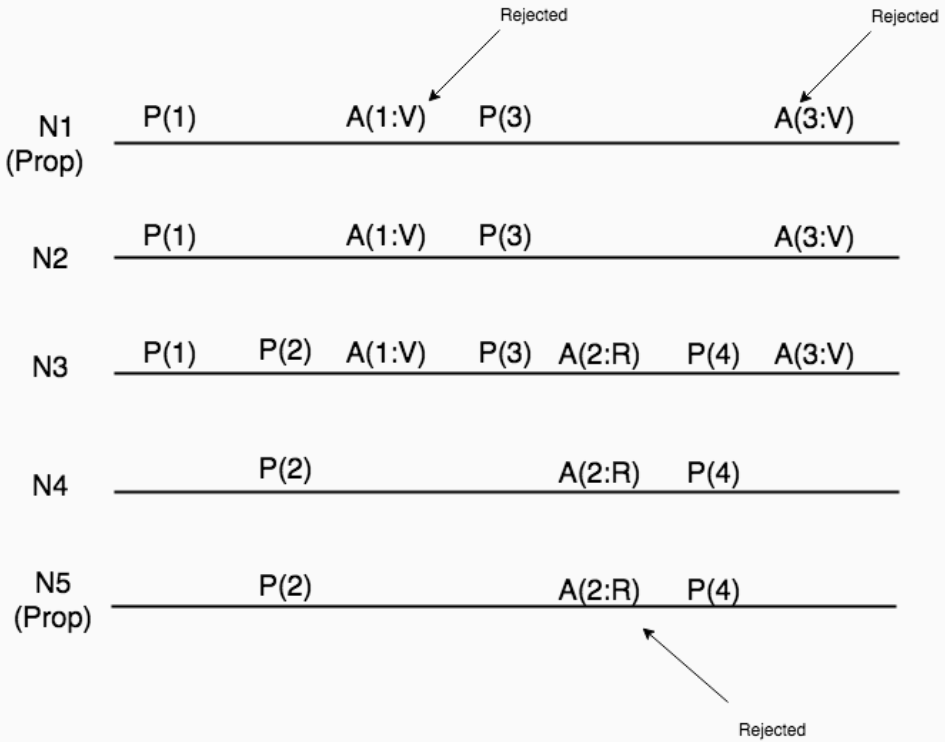
Multiple Proposers conflict (1)

Multiple Proposers can conflict for choosing a value



Multiple Proposers conflict (2)

Multiple Proposers can conflict for choosing a value



Have randomized delay before re-starting.

Basic Paxos (drawbacks)

- There is no single Proposer(or leader), so more conflicts.
- Proposer only knows about the chosen value.
- Acceptors have no knowledge about chosen value. They need to start a round.
- There is communication overhead(2 phases).

Multi Paxos

- A leader to avoid conflicts.
- Instead of Basic Paxos for each command, Multi Paxos for many commands.
- Get rid of Prepare phase for every round, have one phase instead.

Single leader

- Group of processes from (P_1, P_2 … P_n).
- Process with highest id could be leader, say Pn.
- Every process in the group sends heartbeat message to other server, after every *t* ms.
- If any process from P_1 to P_{n-1}, did not receive heartbeat from higher id process.
  - It acts as a leader i.e Proposer and Acceptor
- If a process is not a leader, acts as Acceptor

Eliminate Prepare phase

- Have one proposed number to all the instances.
- Get highest possible accepted value.
- Every Acceptor can return a state, which defines no more accepted value beyond highest possible accepted value.
- On receiving such state from Quorum of Acceptors, it can eliminate Prepare phase.

How Paxos can help ?

- Election of a leader among group of processes.
- Information about group membership (process failures/joining).
- Maintain the order of messages (Log Replication)

Something More ..

Different Versions

- Fast Paxos - Optimization to reduce number of delays.
- Byzantine Paxos - Paxos involving lying processes/ crash failures.
- Cheap Paxos - Using an auxiliary Acceptor in case an actual Acceptor fails.

Summary

- Logical Clocks to maintain local counter for each process.
- Three user agents: Proposer, Acceptor and Listener
- Basic Paxos a two phase protocol.
- Issues such as leader election.
- Get rid of Prepare-Promise phase for every command.
- Different version of Paxos

Resources

Time, clocks, and the ordering of events in a distributed system - Leslie Lamport
https://www.microsoft.com/en-us/research/publication/time-clocks-ordering-events-distributed-system/

Paxos made Simple - Leslie Lamport
https://www.microsoft.com/en-us/research/publication/paxos-made-simple/

Implementing Paxos for Replicated Logs - John Ousterhout and Diego Ongaro
https://www.youtube.com/watch?v=JEpsBg0AO6o

About me

Twitter - https://twitter.com/jdk2588

I am currently working as Backend Engineer for Partywithalocal, a Dutch mobile app.

Email - jdk2588@gmail.com