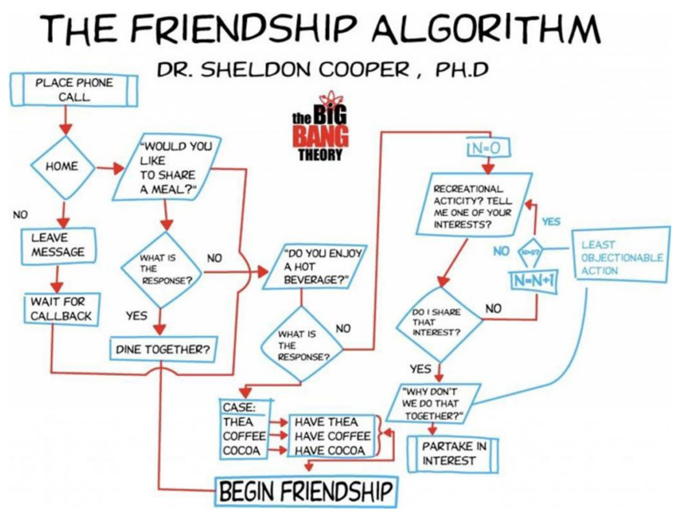
Paxos Made Simple

What is the Consensus problem?

- A group wants to make a decision
- They don't care what decision is made as long as one of them proposed it.

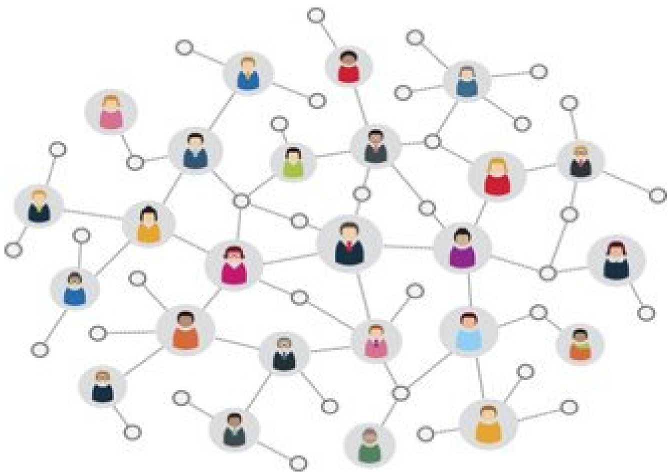
Motivation Replicated State Machine

- We can think of a distributed system as a server and a group of clients making requests from that server.
- We want to have more than one server as back up.
- We want to have all servers synchronized.



Motivation Blockchain

- In blockchain we are dealing with distributed system.
- We want a way to reach consensus on the value of the chain.
- In Bitcoin we do this by the longest chain rule.



Motivation Byzantine Generals Problem

- A group a general of the Byzantine army camped with their troops around an enemy city
- Communicating only by messenger the generals must agree upon a common battle plan.
- However, one or more of them may be traitors who try to confuse the others.

Formally

We have three roles in a consensus algorithm

- Acceptor
- Proposers
- Learner

In implementation it is common that a single process is playing all three roles but we do not need to think about it now.

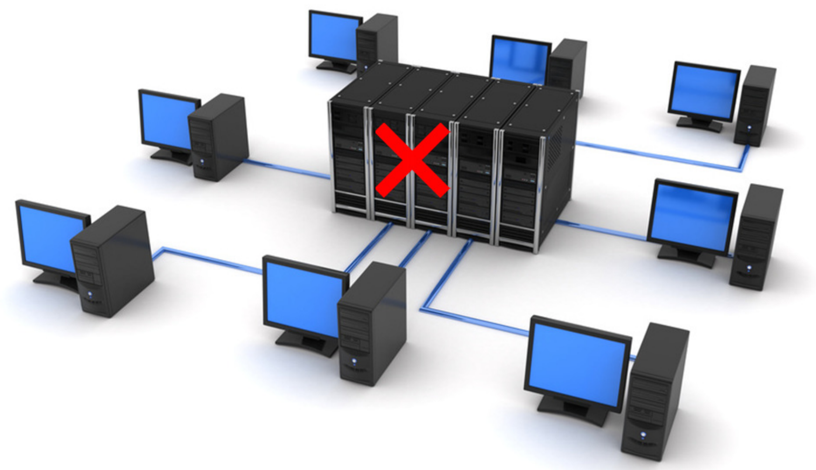
How do we solve this problem

Think of a trivial algorithm: One process is assigned acceptor, all proposers offer him their proposals he chooses the first one that reached him.

Problem: If this process fails no consensus is reached.

How do we solve this problem

Slightly better algorithm: Have a set of processes that are assigned acceptors, each one accepts the first offer it receives. To ensure a single value is chosen for the consensus we choose the value of the algorithm to be the value accepted by a majority of the acceptors.

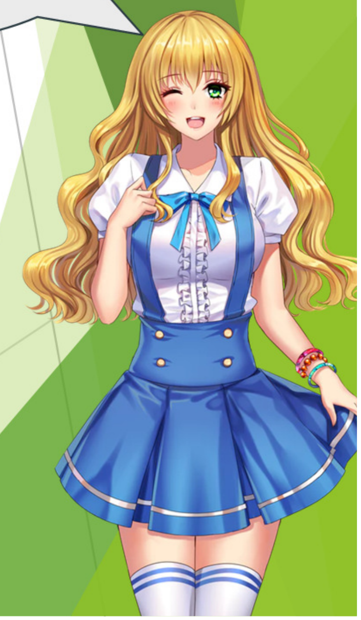


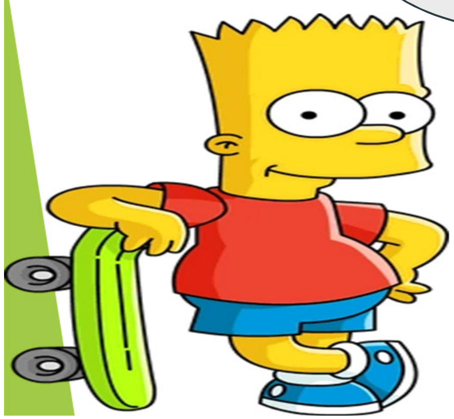
problem: a majority may never be reached, even if there is a odd number of acceptors a single failure may change that.

Idea One

In the absence of failure or message loss we want a value to be chosen even if a single value is proposed by a single proposer.

P1: An acceptor must accept the first offer it receives.





Idea Two

- **P1 and the requirement that a proposal is accepted by a majority of the acceptors implies that an acceptor must accept more than one proposal.**
- **In order to keep track of the proposals we assign each of them a natural number.**
- **We can allow multiple proposals to be chosen but we must guarantee that all chosen proposals have the same value.**

P2: if a proposal with value v is chosen then every proposal with higher numbered that is chosen has value v.

How can we maintain such invariant?

In order to satisfy P2 it is sufficient to satisfy P2a

P2a: if a proposal with value v is chosen then any proposal accepted by an acceptor with higher number has a value v.

This raises another problem: suppose a proposer wakes up and offers a higher numbered proposal with a different value, since we have P1 any acceptor that receives it must accept it causing a violation of P2.

P2b: if a proposal with value v is chosen then any higher numbered proposal issued by any proposer must have value v.

How can we maintain such invariant?

In order to satisfy P2b we propose

P2c: If a proposal with value V and number n is issued then there is a majority of acceptors such that:

- No acceptor has accepted a proposal with a number < n.
- V is the value of the highest number proposal accepted by this majority.


Prepare
Retreat!
None
None
Retreat

The Paxos Algorithm

Proposer Algorithm:

1. Choose a new proposal with a number n send it to all the acceptors asking for:

> **A commitment not to accept requests with number less than n.**
>
> **The highest number request he has accepted if any.**

We call this **prepare** request.

2. If you have received a response from a majority of the acceptors then issue a request with number n and value v where v is the value of the highest numbered proposal or any value if no such proposal exist.

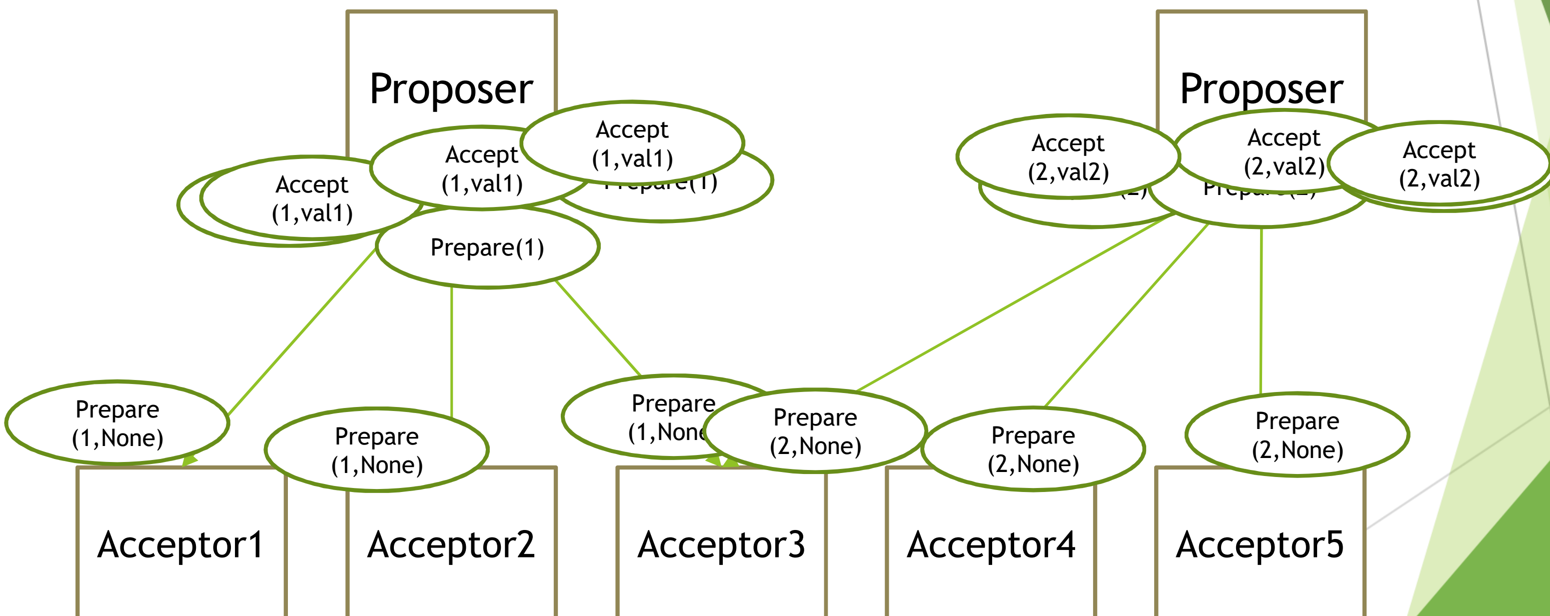
We call this **accept** request.

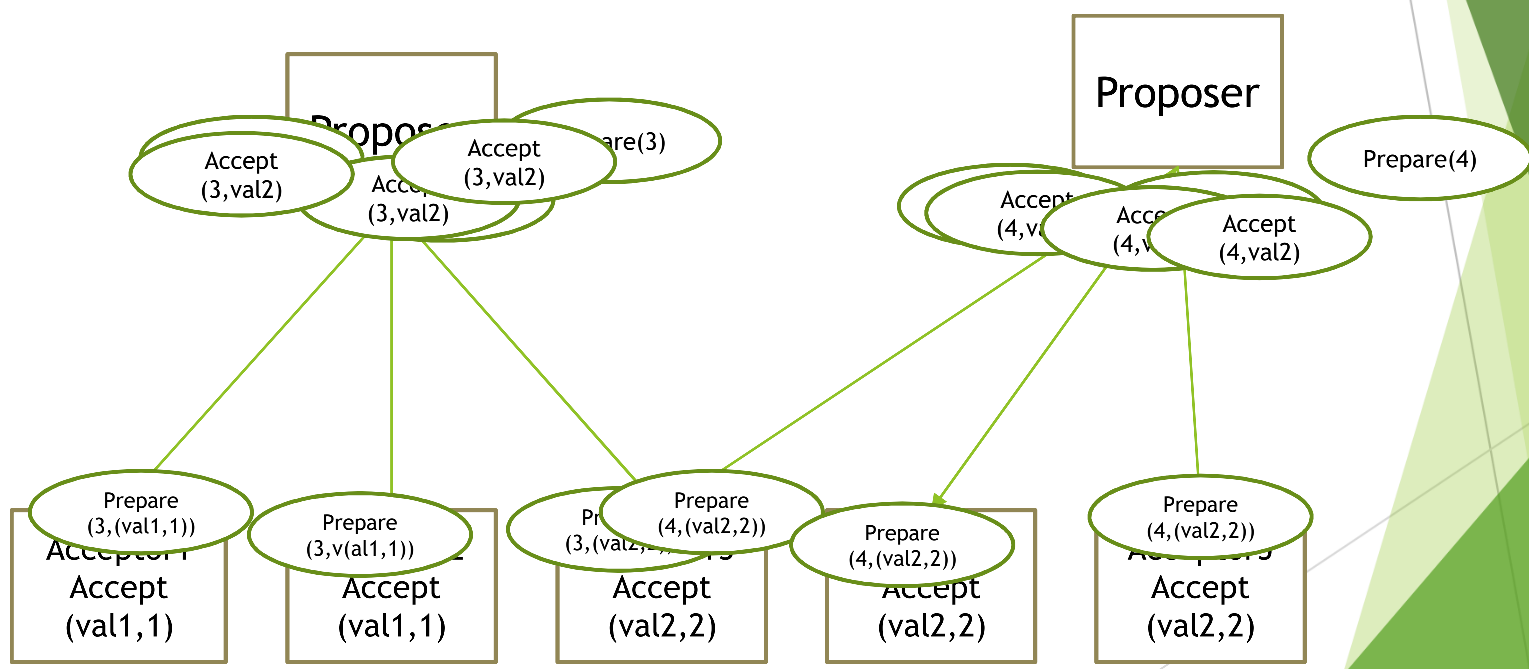
The Paxos Algorithm

- Acceptor Algorithm:
  - Always answer prepare requests.
  - Only answer accept request if you did not commit not to accept it.
- Optimization: also ignore prepare requests with number that you have commited never to accept.

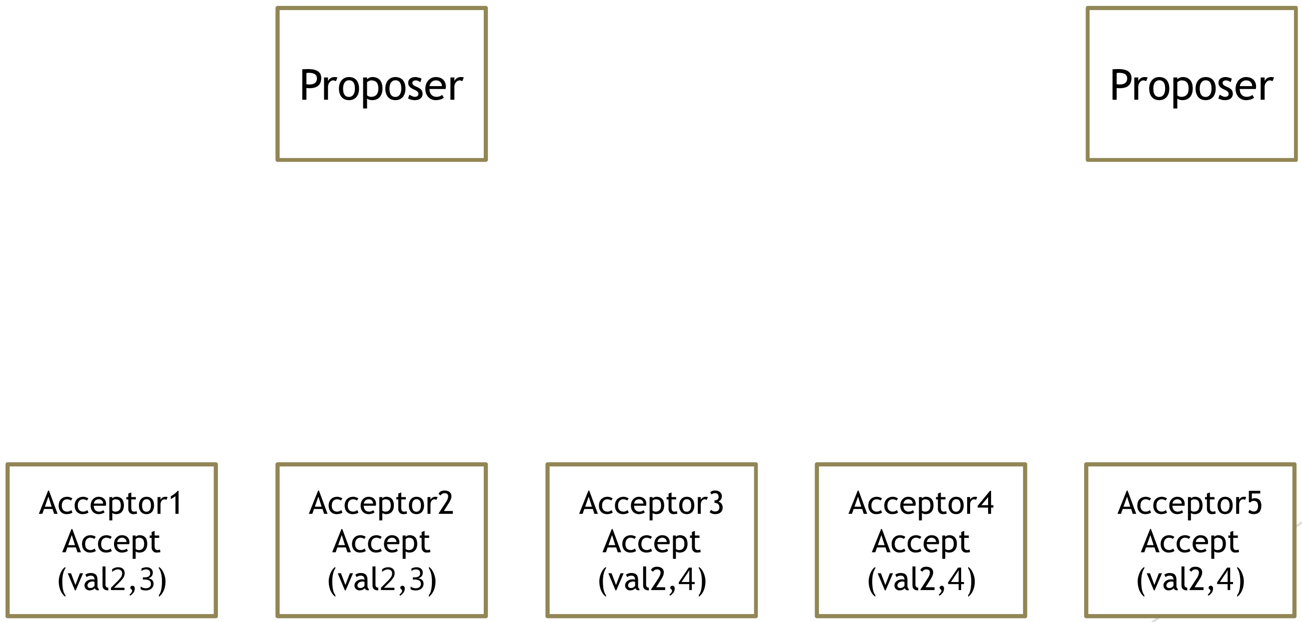
The Paxos Algorithm

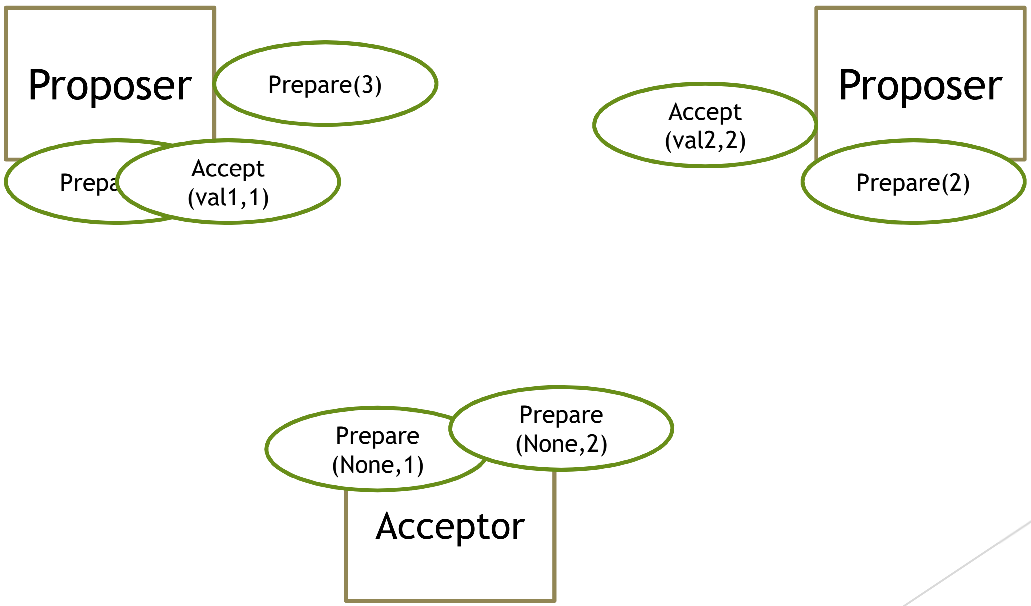
- **Learner Algorithm:**
- **We could have each acceptor send the value he accepts every time he accepts a new value to each of the learners. This is wasteful instead each acceptor has a set of distinguished learners every time he accepts a new value he informs each of those learners they are incharge of informing all the other learners of this.**

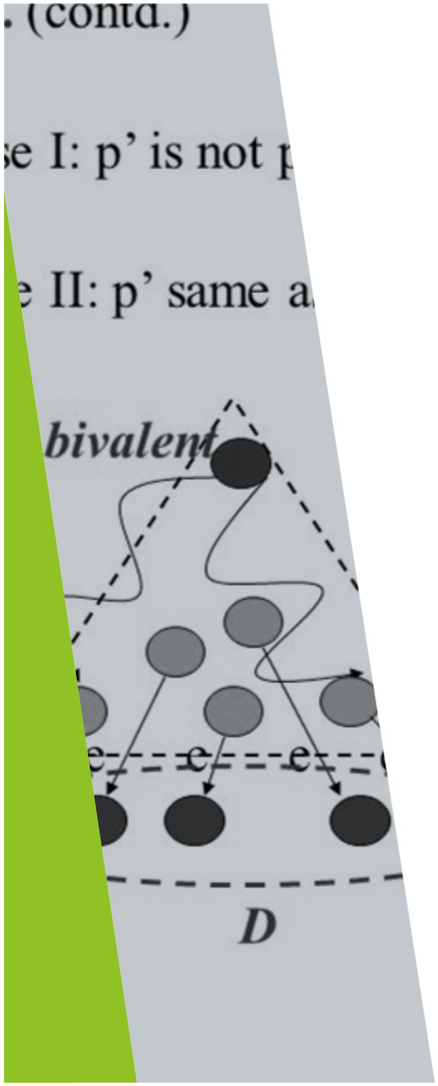

Proposer
Accept
(1,val1)
Accept
(1,val1)
Accept
(1,val1)
Prepare(1)
Prepare(1)
Proposer
Accept
(2,val2)
Accept
(2,val2)
Accept
(2,val2)
Prepare
(1,None)
Prepare
(1,None)
Prepare
(2,None)
Prepare
(2,None)
Prepare
(2,None)
Acceptor1
Acceptor2
Acceptor3
Acceptor4
Acceptor5



Proposer
Prepare(4)
Accept
(3,val2)
Accept
(3,val2)
Accept
(4,val2)
Prepare
(3,(val1,1))
Prepare
(3,v(al1,1))
Prepare
(4,(val2,2))
Prepare
(4,(val2,2))
Prepare
(4,(val2,2))
Accept
(val1,1)
Accept
(val1,1)
Accept
(val2,2)
Accept
(val2,2)
Accept
(val2,2)


Proposer
Proposer
Acceptor1 Accept (val2,3)
Acceptor2 Accept (val2,3)
Acceptor3 Accept (val2,4)
Acceptor4 Accept (val2,4)
Acceptor5 Accept (val2,4)


Proposer
Prepare(3)
Prepa
Accept
(val1,1)
Accept
(val2,2)
Proposer
Prepare(2)
Prepare
(None,1)
Prepare
(None,2)
Acceptor



FLP

- We solve this problem by selecting a leader Proposer. But how do we do that?
- Why not use an instance of Paxos?
- The famous result of Fischer, Lynch, and Patterson [1] implies that a reliable algorithm for selecting a proposer must use either randomness or real time.
- Proof idea: There is always an initial “configuration” of the system that can reach more the one value. There is also for each configuration that can reach more than one value a configuration following it that can reach more than one value.

- [1] Michael J. Fischer, Nancy Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. Journal of the ACM, 32(2):374–382, April 1985.

Implementation Details

- A process implements learner, proposer and acceptor.
- We pick requests numbers from disjoint sets (Think of finite fields).
- A server implementing the Paxos algorithm must have stable storage. Stable storage is used to store the number of the highest numbered proposal made by this server.

The part time parliament

- The name Paxos is actually a name of a island in Greece. For more details read [2]

- [2] Leslie Lamport. The part-time parliament. ACM Transactions on Computer Systems, 16(2):133–169, May 1998.

Questions