**SUMMARY:**

PAXOS is a simple yet powerful and crucial consensus algorithm that has been proposed. This paper explains it in simple terms. First we see how the algorithm communicates over set of fixed roles and messages to reach consensus. The next part suggests what other issues could occur while following this alogrithm and how to overcome those.

The problem is that we need to chose a value in a safe environment where there wont be multiple values agreed upon or unproposed values chosen or processes missing out on chosen values. The agents in this system would have three roles: Proposer, acceptor, learner. An agent could act as one or all of the roles. The network is assumed to be non-byzantine, customary asynchronous model where messages can be delivered very late but not corrupted and agents fail by stopping. Choosing a value can occur in 2 phases,

Phase1 :a) Proposer selects a proposal number *n* and sends a *prepare* message to a majority of the acceptors. b) The acceptor sends a promise to ignore all requests lower than this request n and reply with the highest -numbered proposal accepted( if any).

Phase 2.a) If the proposer receives response from majority of acceptors, it now sends an *accept* request to the same acceptors with proposal number n with value v , where v would be the highest-numbered proposal among the responses, or any value if no value was accepted.

b)If acceptor receives an accept request it accepts the proposal unless it has promised an higher numbered prepare request.

There can be multiple proposal and the any proposal can be dropped in between at any time.

Learning a chosen value: Instead of each acceptor sending out a message to all learners, we can send it to a single/ a set of distinguished learners who can then propagate the message to rest of the learners.

Acceptors send this message as and when a value has been accepted for a proposal. If any agents never figures out the value accepted on, it can get a proposal initiated through a proposer to find out the value in case of message loss.

There can be scenarios where the phase one doesnot get completed due to contention between the proposers each re-initiating a proposal with higher number, before any value can be agreed upon. To avoid this we can have a distinguished proposer, who will be the only one to try issuing proposals. This can be achieved either through timeouts or other randomness or realtime to elect a distinguished proposer. The algorithm also choses a leader which plays the role of both a distinguished proposer and a distinguished learner. The Paxos consensus algorithm is precisely the one described above, where requests and responses are sent as ordinary messages. Stable storage, preserved during failures, is used to maintain the information that the acceptor must remember. An acceptor records its intended response in stable storage before actually sending the response. To guarantee that no two proposals are ever issued with the same number different proposers choose their numbers from disjoint sets of numbers, so two different proposers never issue a proposal with the same number. Each proposer remembers (in stable storage) the highest-numbered proposal it has tried to issue, and begins phase 1 with a higher proposal number than any it has already

used. Its easy implement this kind of distributed system as a state machine. Since we would know the steps that would be executed are deterministic. To guarantee that all servers execute the same sequence of state machine commands, we implement a sequence of separate instances of the Paxos

consensus algorithm, the value chosen by the *i*th instance being the *i* th state machine command in the sequence. Each server plays all the roles (proposer, acceptor, and learner) in each instance of the algorithm. Clients send commands to the

leader, who decides where in the sequence each command should appear.

If the leader decides that a certain client command should be the 135th

command, it tries to have that command chosen as the value of the 135th

instance of the consensus algorithm. Here the paper explains how the commands are chosen and how a gap could occur in between and what strategies are taken to fix those. Finally we learn that paxos is highly efficient with least messages to be sent even in case of faults by batch execution of such commands and support reconfiguration.