

Chapter 5a

Paxos

The consensus problem

How can a Distributed system come to a decision in the presence of:

failing machines: *messages may never be sent*

unreliable links: *Messages may arrive arbitrarily late.*

But processes do not *collude, lie or divert the protocol* – referred to as Byzantine failures and need a more sophisticated algorithm

While respecting

Safety

Non-trivial– only proposed values are learned

Consistency – one value at most is learned

Liveness

If some value has been learned, then some learner will eventually learn some value

Agreement in the presence of failure

Distributed systems need to come to a decision.
A leader; commit or rollback a transaction.

The decision must be **unique** – only one value can be chosen by the system. One value must be chosen.

This is hard because messages may be lost or arrive late by an unbounded amount

This solutions to this sort of problem are described as *safe* if they provide an answer required by the problem and are said to *progress* if they also guarantee to finish in a finite time

An algorithm (due to Lamport) called *Paxos* solves the consensus problem in a way which is *safe* but is not guaranteed to *progress*.

It will progress if the system is stable for some length of time.

Client requests information from a distributed system and awaits a response

Distributed Node roles

A node can take on any or all of the roles

Proposer

Acceptor

Learner

Leader

Leader is a special proposer needed to make progress.
chosen by any suitable election

A proposer proposes a value which it sends to a set of acceptors.

Each acceptor independently decides on whether to accept a value – it may receive multiple values from different proposers.

Acceptors send their values to a set of learners.

For a value to be chosen by *Paxos* a majority of the acceptors must choose the same value.

Prepare and Accept

A proposer first sends a **prepare request** to each acceptor

(A proposer does not know if a given acceptor is running – or if the prepare request will actually reach them)

The prepare request contains a proposed value v and a proposal number n .

The proposal number is a unique natural number, which increases monotonically (never decreases). Choosing the number in a distributed system is a problem which *Paxos* does not consider.

Each proposer can for instance have a disjoint set.

1,6,11, ...	4, 9, 14, ...
2,7,12, ...	5, 10, 15,
3,8,13, ...	

Prepare Response

An acceptor which has not seen a proposal replies with a prepare response promising not to accept a proposal with a smaller proposal number.

If it has seen a proposal with a higher proposal number it ignores the request.

If it has accepted a proposal with a smaller proposal number it sends back the highest proposal number it has received along with its value.

Receives Prepare response

An proposer waits until it has received prepare responses from the majority of acceptors.

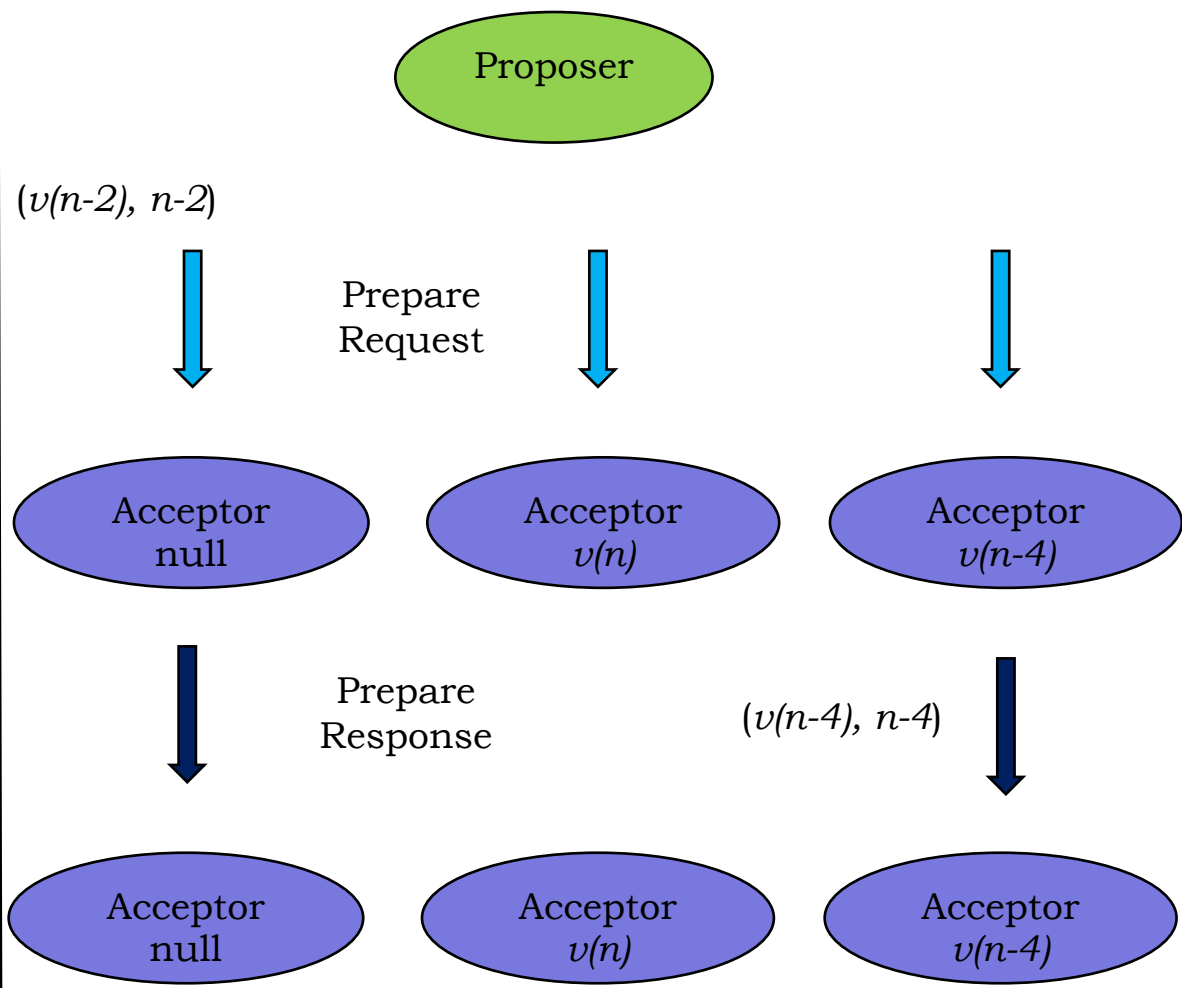
They may all be promises to accept the value:
The proposer then sends an accept request to all of the acceptors with the same value and proposal number.

Some of the responses may be for values which the acceptor has already agreed to accept. The proposer finds the value associated with the highest proposal number which has already been accepted by any of the learners. It sends an accept request with the original proposal number and the value of this highest previously accepted request.

Receives Accept Request

When the acceptor receives an accept request with a proposal number equal to or higher than the proposal it has agreed to honour it sends a notification to every learner.

When any learner is informed that a majority of the acceptors have accepted. Then the value is agreed upon.



Widely used

Google Chubby

Yahoo Zookeeper

Microsoft autopilot