This document provides a terse summary of the Basic Paxos (single-decree) consensus protocol as well as Multi-Paxos. It is intended as an accompaniment to a one-hour video lecture introducing Paxos, which was developed as part of a user study comparing Paxos with the Raft consensus algorithm. Multi-Paxos is not specified precisely in the literature; our goal here is to provide a fairly complete specification that stays close to Leslie Lamport’s original description of Paxos in “The Part-Time Parliament.” The version of Multi-Paxos described here has not been implemented or proven correct.

1 Basics

- proposal number \((n) = \text{(round number, server ID)}\)
- \(T\): a fixed timeout value used in the leader election algorithm
- \(\alpha\): concurrency limit in Multi-Paxos

1.1 Leader election algorithm

- Every \(T\) milliseconds, send an empty heartbeat message to every other server.
- A server acts as leader if it has not received a heartbeat message in the last \(2T\) milliseconds from a server with higher ID.

2 Basic Paxos (Single-decree)

2.1 Persistent state per server

- \(\text{minProposal}\): the number of the smallest proposal this server will accept, or 0 if it has never received a Prepare request
- \(\text{acceptedProposal}\): the number of the last proposal the server has accepted, or 0 if it never accepted any
- \(\text{acceptedValue}\): the value from the most recent proposal the server has accepted, or null if it has never accepted a proposal
- \(\text{maxRound}\): the largest round number the server has seen

2.2 Messages

2.2.1 Prepare (Phase 1)

Request fields:

- \(n\): a new proposal number

Upon receiving a Prepare request, if \(n \geq \text{minProposal}\), the acceptor sets \(\text{minProposal}\) to \(n\). The response constitutes a promise to reject Accept messages with proposal numbers less than \(n\) in the future.

Response fields:

- \(\text{acceptedProposal}\): the acceptor’s \(\text{acceptedProposal}\)
- \(\text{acceptedValue}\): the acceptor’s \(\text{acceptedValue}\)
2.2.2 Accept (Phase 2)

Request fields:
- $n$: the same proposal number used in Prepare
- $v$: a value, either the highest numbered one from Prepare responses, or if none, then one from a client request

Upon receiving an Accept request, if $n \geq \text{minProposal}$, then:
- Set $\text{acceptedProposal} = n$
- Set $\text{acceptedValue} = v$
- Set $\text{minProposal} = n$

Response fields:
- $n$: the acceptor’s $\text{minProposal}$

2.3 Proposer Algorithm: $\text{write}(\text{inputValue}) \rightarrow \text{chosenValue}$

1. Let $n$ be a new proposal number (increment and persist $\text{maxRound}$).
2. Broadcast $\text{Prepare}(n)$ requests to all acceptors.
3. Upon receiving $\text{Prepare}$ responses ($\text{reply}.\text{acceptedProposal}$, $\text{reply}.\text{acceptedValue}$) from a majority of acceptors:
   - Let $v$ be set as follows: if the maximum $\text{reply}.\text{acceptedProposal}$ in the replies isn’t 0, use its corresponding $\text{reply}.\text{acceptedValue}$. Otherwise, use $\text{inputValue}$.
4. Broadcast $\text{Accept}(n, v)$ requests.
5. Upon receiving an $\text{Accept}$ response with ($\text{reply}.n$):
   - If $\text{reply}.n > n$, set $\text{maxRound}$ from $n$, and start over at step 1.
6. Wait until receiving Accept responses for $n$ from a majority of acceptors.
7. Return $v$.

3 Multi-Paxos

3.1 Persistent state per acceptor

Each acceptor stores:
- $\text{lastLogIndex}$: the largest entry for which this server has accepted a proposal
- $\text{minProposal}$: the number of the smallest proposal this server will accept for any log entry, or 0 if it has never received a Prepare request. This applies globally to all entries.

Each acceptor also stores a log, where each log entry $i \in [1, \text{lastLogIndex}]$ has the following fields:
- $\text{acceptedProposal}[i]$: the number of the last proposal the server has accepted for this entry, or 0 if it never accepted any, or $\infty$ if $\text{acceptedValue}[i]$ is known to be chosen
- $\text{acceptedValue}[i]$: the value in the last proposal the server accepted for this entry, or null if it never accepted any

Define $\text{firstUnchosenIndex}$ as the smallest log index $i > 0$ for which $\text{acceptedProposal}[i] < \infty$

3.2 Persistent state per proposer
- $\text{maxRound}$: the largest round number the proposer has seen

3.3 Soft (volatile) state per proposer

(I’m not doing a very strong separation here between the proposer and the acceptor. I allow proposers to both read and write into acceptor state sometimes.)
- $\text{nextIndex}$: the index of the next entry to use for a client request
- $\text{prepared}$: True means there is no need to issue Prepare requests (a majority of acceptors has responded to Prepare requests with $\text{noMoreAccepted}$ true); initially false
3.4 Messages

3.4.1 Prepare (Phase 1)

Request fields:
- \( n \): a new proposal number
- \( index \): the log entry that the proposer is requesting information about

Upon receiving a Prepare request, if \( request.n \geq minProposal \), the acceptor sets \( minProposal \) to \( request.n \).

The response constitutes a promise to reject Accept requests (for any log entry) with proposals numbered less than \( request.n \).

Response fields:
- \( acceptedProposal \): the acceptor’s \( acceptedProposal[index] \)
- \( acceptedValue \): the acceptor’s \( acceptedValue[index] \)
- \( noMoreAccepted \): set to true if this acceptor has never accepted a value for a log entry with index greater than \( index \)

3.4.2 Accept (Phase 2)

Request fields:
- \( n \): the same proposal number used in the most recent Prepare
- \( index \): identifies a log entry
- \( v \): a value, either the highest numbered one from a Prepare response, or if none, then one from a client request
- \( firstUnchosenIndex \): the sender’s \( firstUnchosenIndex \)

Upon receiving an Accept request: if \( n \geq minProposal \), then:
- Set \( acceptedProposal[index] = n \)
- Set \( acceptedValue[index] = v \)
- Set \( minProposal = n \)

For every \( index < request.firstUnchosenIndex \), if \( acceptedProposal[index] = n \), set \( acceptedProposal[index] \) to \( \infty \).

Response fields:
- \( n \): the acceptor’s \( minProposal \)
- \( firstUnchosenIndex \): the acceptor’s \( firstUnchosenIndex \)

3.4.3 Success (Phase 3)

Request fields:
- \( index \): identifies a log entry
- \( v \): the chosen value for entry \( index \)

Upon receiving a Success request, set \( acceptedValue[index] \) to \( v \) and \( acceptedProposal[index] = \infty \).

Response fields:
- \( firstUnchosenIndex \): the acceptor’s \( firstUnchosenIndex \)

When the sender receives the response, if \( reply.firstUnchosenIndex < firstUnchosenIndex \) then the sender sends \( Success(index = reply.firstUnchosenIndex, value = acceptedValue[reply.firstUnchosenIndex]) \).

3.5 Proposer Algorithm: \( write(inputValue) \rightarrow bool \)

1. If not leader or not done with leader initialization, return false.
2. If \( prepared \) is true:
   (a) Let \( index = nextIndex \), increment \( nextIndex \).
   (b) Go to step 6.
3. Let \( index = firstUnchosenIndex \) and \( nextIndex = index + 1 \).
4. Let \( n \) be a new proposal number (increment and persist \( maxRound \))
5. Broadcast \( Prepare(n, index) \) requests to all acceptors.
6. Upon receiving \( Prepare \) responses (\( reply.acceptedProposal \), \( reply.acceptedValue \), \( reply.noMoreAccepted \)) from a majority of acceptors:
• Let $v$ be set as follows: if the maximum $\text{reply.acceptedProposal}$ in the replies isn’t 0, use its corresponding $\text{reply.acceptedValue}$. Otherwise, use $\text{inputValue}$.
• If all acceptors in the majority responded with $\text{reply.noMoreAccepted}$, set $\text{prepared} = \text{true}$.

7. Broadcast $\text{Accept}(\text{index}, n, v)$ requests to all acceptors.
8. Upon receiving an $\text{Accept}$ response with $(\text{reply.n}, \text{reply.firstUnchosenIndex})$:
   • If $\text{reply.n} > n$, set $\text{maxRound}$ from $\text{reply.n}$. Set $\text{prepared} = \text{false}$. Go to step 1.
   • If $\text{reply.firstUnchosenIndex} \leq \text{lastLogIndex}$ and $\text{acceptedProposal}[\text{reply.firstUnchosenIndex}] = \infty$, then send $\text{Success}(\text{index} = \text{reply.firstUnchosenIndex}, \text{value} = \text{acceptedValue}[\text{reply.firstUnchosenIndex}])$.

9. Upon receiving Accept responses for $n$ from a majority of acceptors:
   • Set $\text{acceptedProposal}[\text{index}] = \infty$ and $\text{acceptedValue}[\text{index}] = v$.
10. If $v == \text{inputValue}$, return true.
11. Go to step 2.

4 Reconfiguration

• Configuration is a list of ids and addresses of servers, stored as special log entries
• Configuration for choosing entry $i$ determined by latest configuration in log at entry $i - \alpha$ or below.
• $\alpha$ limits concurrency: can’t choose entry $i + \alpha$ until entry $i$ is chosen