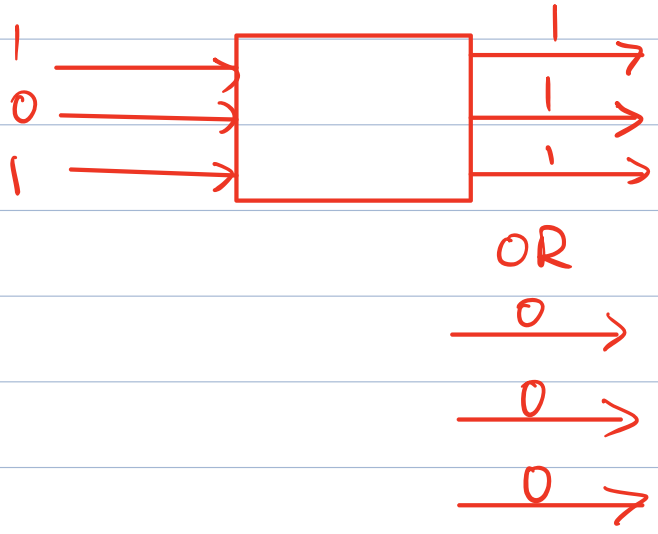
Agenda:

- definition of consensus
- FLP result
- Paxos: the easy parts
- Paxos: the interesting/hard parts

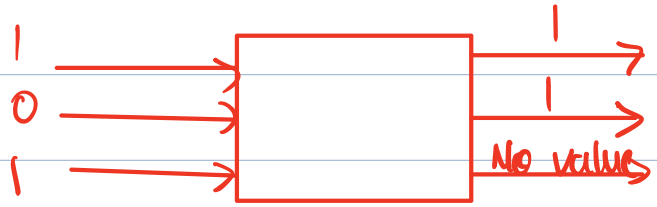
Consensus



Can be usually done with 'majority'.

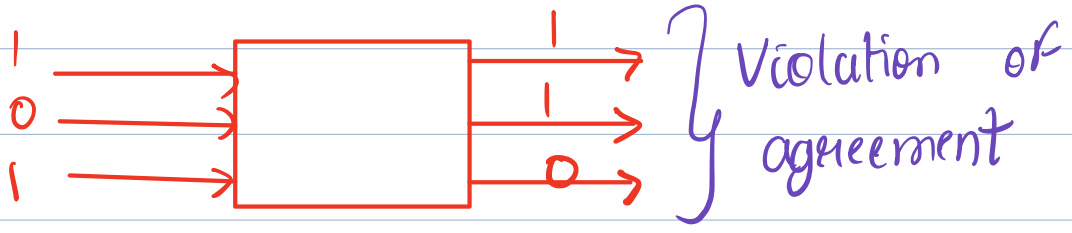
Properties that consensus algos TRY to satisfy:

Termination: Each correct process eventually decides on a value.



} Violation of termination.

Agreement: All correct processes decide on the same value.

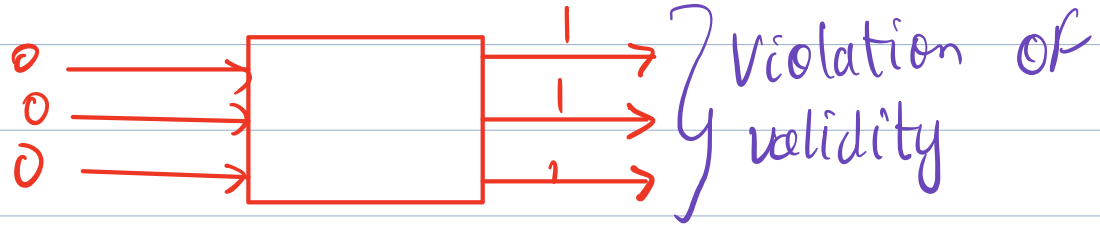


If only Termination & Agreement are the reqd conditions, how can we implement consensus easily?

Just have everyone output a constant value! i.e. Ignore the input.

Validity(Integrity; non triviality):

The agreed upon value must be one of the proposed values



No consensus algo satisfies all 3 conditions in a async network mode & a crash fault model. Why?

Impossible. Proved in 1983 by

Michael Fischer

Nancy Lynch
Michael Paterson

} FLP Result!

In some cases, we can compromise on TERMINATION

Paxos compromises on TERMINATION

If a process doesn't terminate, does it satisfy agreement?
YES!

PAXOS

Consensus Algorithm by Leslie Lamport (1998)
2001 - Paxos made simple

Roles:-

Proposer: Propose values

Acceptor: Contribute to choosing from among the proposed value

Learners - Learns the agreed upon value

One process COULD TAKE on multiple roles

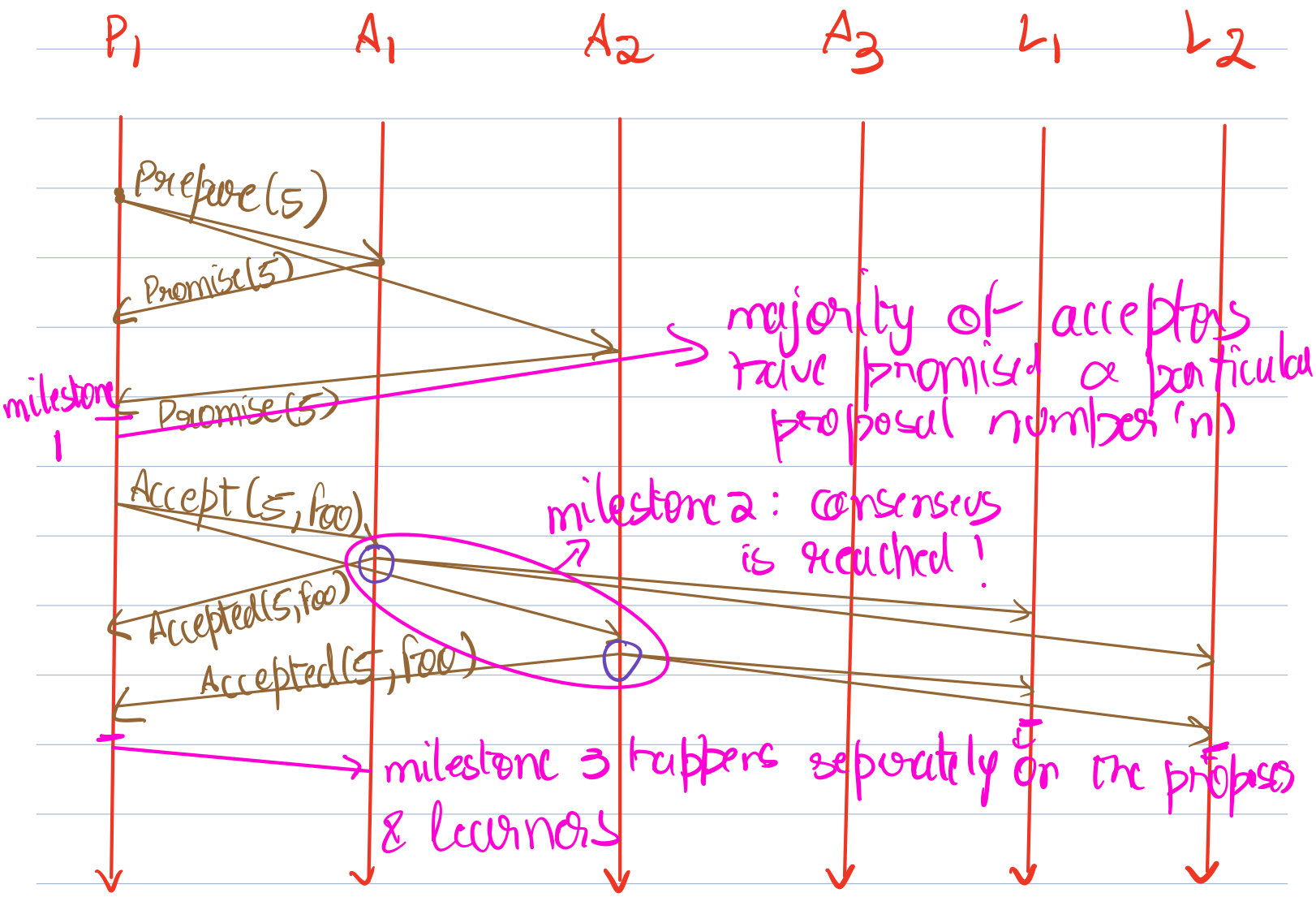
Paxos Node: any node that plays any role

Prerequisites

Paxos nodes must:

- Persist Data
- Know how many nodes is a majority of acceptors

Demo



Paxos Phase 1 (up until milestone 1)

- Majority Acceptors: 2

- P_1 sends prepare msg to atleast majority of acceptors with a proposal number (n)
  - 'n' has to be unique (globally) & higher than any number used by the current proposer has used before.
  - Store proposal number so as to not reuse them. Also the reason why nodes need to persist data

- When acceptor gets a proposal number, it will check to see if it previously promised to ignore this proposal number.
  - If yes, ignore
  - If no: It now promises to ignore any proposal with number lower than 'n'. And reply to the proposer with a Promise(n) message *

(* Addressed later)

↓

I will ignore any request with proposal

number lower than 'n'.

Paxos Phase 2 :-

- Proposer has received promise(n) from a majority of acceptors for some 'n'.
- Send Accept(n, val) message to at least a majority of acceptors where

  n - proposal number that was promised

  val - Actual value it wants to propose (Addressed later)

- When acceptor gets the accept message: i.e Accept(n, val); it will check if it previously promised to ignore requests with this proposal number.

  - If yes; ignore message!

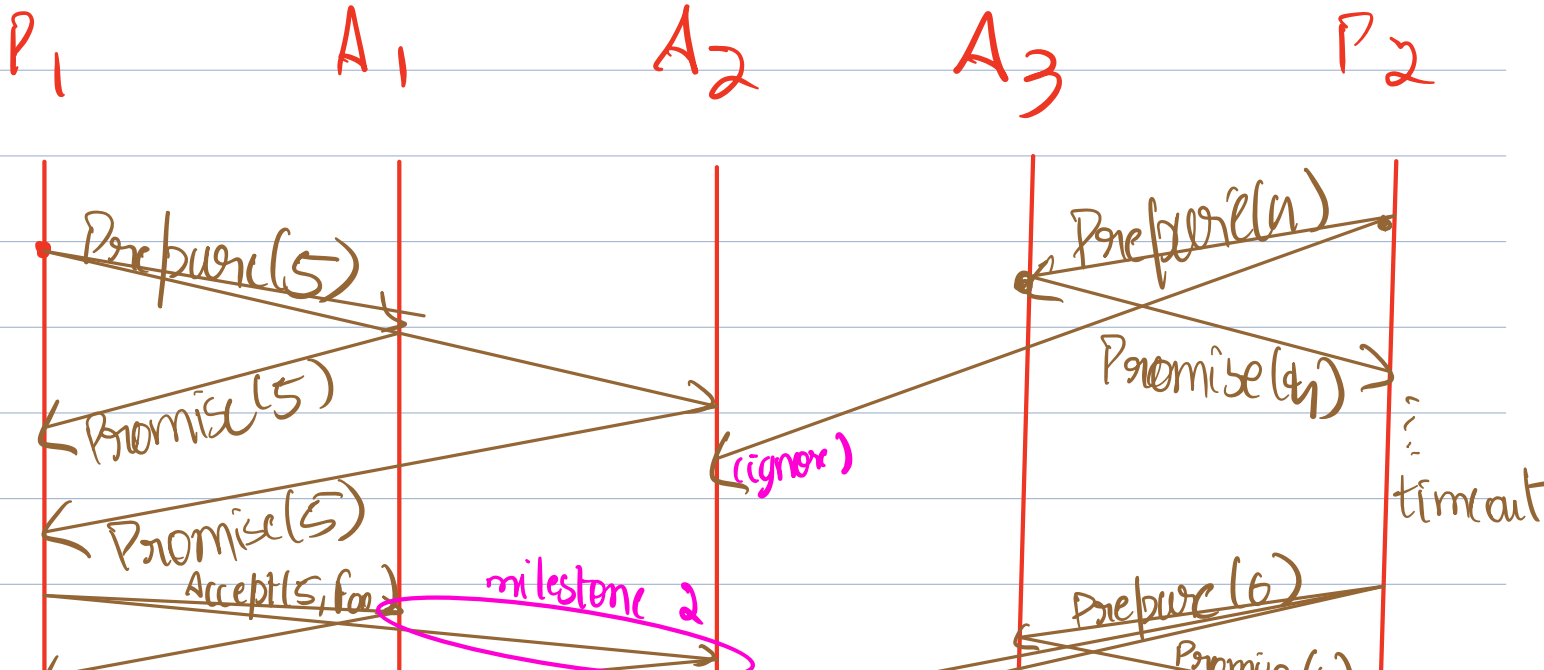
  - If no: Reply with Accepted(n, val), & also sends that message to all the Learners (not just majority)

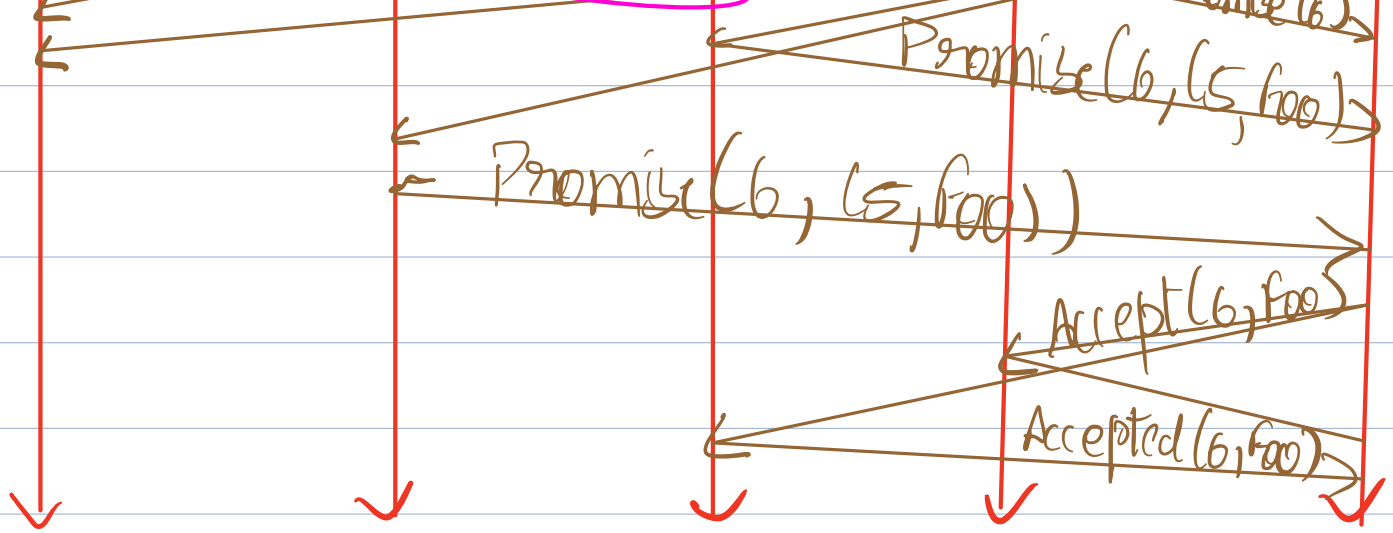
- The point at which majority of acceptors send out an Accepted (n, val) message, is when consensus is reached (milestone 2)

- when each individual process receives Accepted (n, val) message from majority of acceptors, they are now aware that consensus has been reached.

Addressing Asterisk(*) points:

- Lets assume we have multiple proposers





First Astrisk (*)

Acceptor:

On receiving a Prepare(n) message:

"Did I previously promise to ignore requests with proposal number"

- If yes: Ignore it
- If not: it asks

"Have I previously accepted anything?"

If yes: it replies with

Key difference!

Promise(n, n_{prev}, val_{prev})

n_{prev} - highest previously accepted proposal number

val_{prev} : previously accepted value

If not: Reply with Promise(n)

Second asterisk (**)

(or Promise(n, n_{prev}, val_{prev}))

- Proposer has received promise(n) from a majority of acceptors for some 'n'.
- Send Accept(n, val) message to at least a majority of acceptors where

n - proposal number that was promised

val - Actual value it wants to propose

~~**~~ (Addressed later)

val is chosen as follows:

The val_{prev} that went with highest n_{prev}

OR

If there are none, do whatever it wants.

Key difference!

Consensus properties:

termination - When will Paxos not terminate?
Agreement
Validity