Lec-15 : Paxos: the interesting parts.

6:10

Milestones in the Paxos run (of two clients)

(1) ~~When~~ First milestone comes when a majority of acceptors have promised a particular proposal number

(11) Second milestone comes when consensus reached
and that comes when majority of acceptors send the Accepted(T,0), even though nobody known at ~~that time~~

(111) when a proposer or learner gets an accepted msg from majority of acceptors, that's when they know consensus is reached.

① Proposer

→ when a proposer wants to propose a value it sends Prepare(n) to atleast majority of acceptors.

n must be unique
and should be higher than any proposal no. that that proposer has tried before.

Notice:- That I say, it has to be higher than any proposal number that "that" proposer has tried before, it doesn't necessarily have to be higher than any proposal number in the whole system which is good because. If it did have to be higher than any proposal number in the system then it would have been unreliable.

② Acceptor

→ When an acceptor gets this Prepare(n) msg,

It check whether it previously promised to ignore requests with this proposal number

if yes: ignore it
if no: reply to proposer with Promise(n)

I promise to ignore all msg with prop. no. $\leq n-1$

(3) Proposer

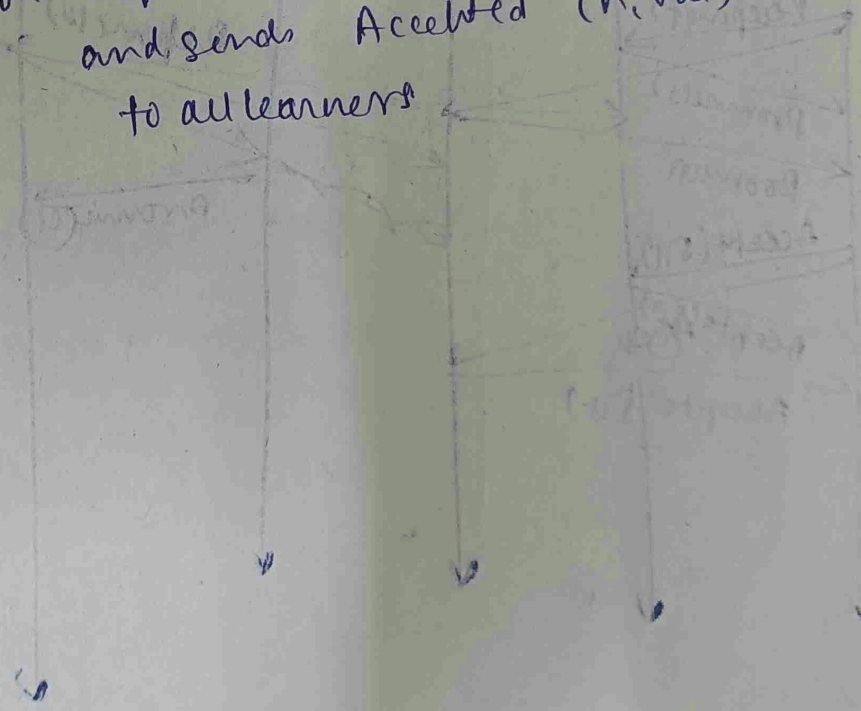
→ when a proposer has received Promise messages from a majority of acceptors for a particular n: it sends Accept (n, val) to atleast to a majority of acceptors, with n and where ~~value is the~~ val is the value it wants. * *

(4) Acceptor

→ when an acceptor gets Accept (n, val) message it asks, "Did I promise to ignore this?"

If yes: ignore it.

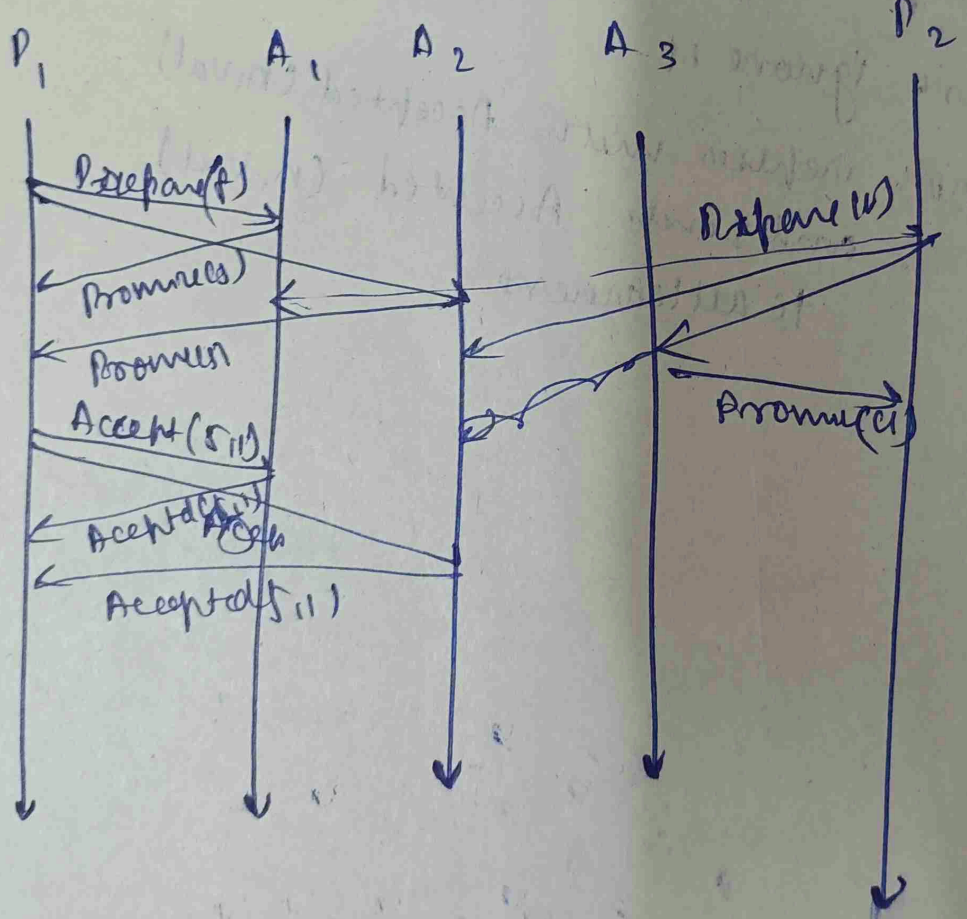
no: replies with Accepted (n, val) and sends Accepted (n, val) to all learners.



What would happen if we had multiple Proposers?

(For making it easier, let's leave out the learners, we will pretend they are there but getting rid of them for now eg, will not mess anything)

Lets say we have two proposers (P_1 and P_2) and three acceptors (A_1, A_2 & A_3)



what happens to the Acceptors when they get 'Prepare(5)' from P_2?

→ A_1 rejects it

A_2 rejects it

A_3 sends Promise(5)

but there's no way that P_2 is going to hear back from a majority of acceptors

So, P_2 will sit here waiting to hear from a majority of acceptors and it won't, so, eventually it will timeout

Hint: P_2 has no idea about what's going on with P_1, all P_2 knows that it has send a prepare and it only heard back from one acceptor, it doesn't know why? It doesn't know if the acceptors other are slow or etc

So what can P_2 do know?

→ It turns out that it's 'ok', always 'ok' to just go ahead and send a higher number prepare msg, as long as the proposal no. is unique, you are okey

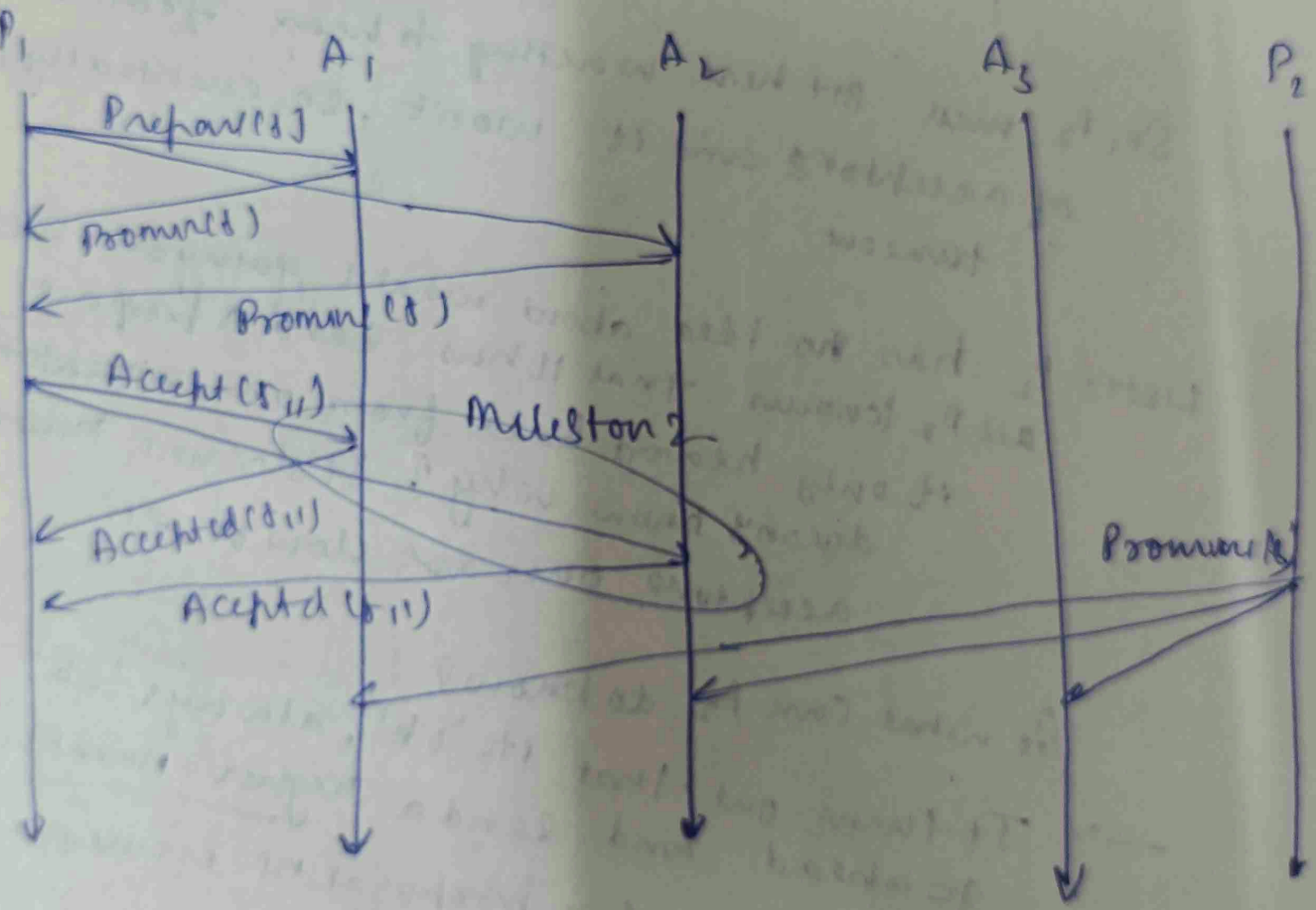
we are assuming P_1 → send odd numbered proposal

P_2 → " even " "

In practice, the way this would work is that the proposer has some kind of a timeout

"If it sends a Prepare message and enough time goes by and it never hears from majority of acceptors then it can send a higher numbered one"

let's say after few time when P_2 timer expires, it decides to send Prepare(8)



how any of the acceptors have promised to ignore requests with a proposal number of 8?

→ No, so acceptor you have to respond but we have already reached our second milestone in our Paxos run

So, consensus has been reached,
it's just that P2 didn't know it yet

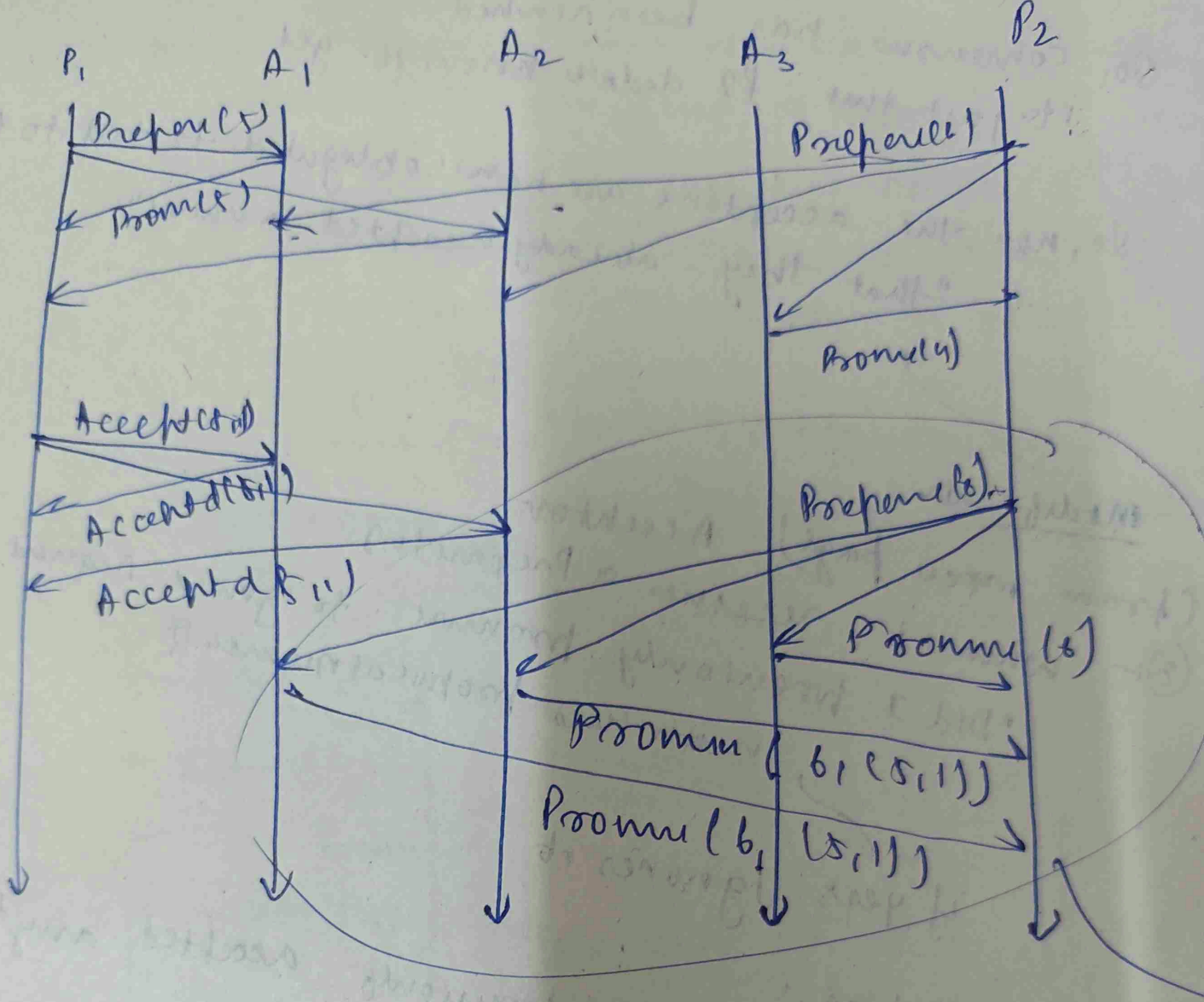
So, now the acceptors are now obligated to tell to P2
"that they already accepted a value"

Modification

(from prev page) Acceptor

(2) - when it receives a Prepare(n):
"Did I previously promise to ignore request with this proposal number?"
↳ if yes: ignores it
↳ if no:
↳ Have I previously accepted anything
↳ if yes
↳ respond to Proposer with Promise
(n, (n_prev, val_prev))
n_prev: its (acceptor) highest previously accepted proposal no.
val_prev: previous accepted proposal value.
↳ if no
↳ replies to Proposer with
Promise(n)
which means "I promise to ignore requests with proposal no. lower than n"

4:42 PM.



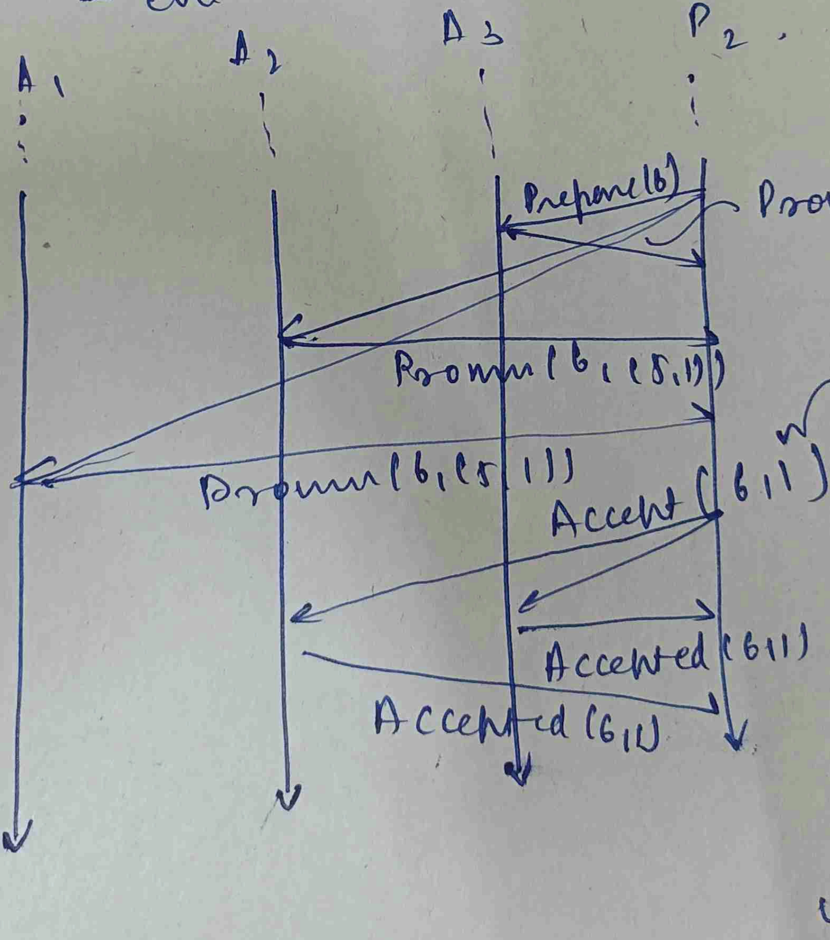
What is P_2 supposed to do when he gets Promise$(6, (5,1))$

modifying (3)

(3) Proposer :-

when a proposer has received Promise messages from a majority of acceptors for a particular n, it sends Accept (n, val) to a majority of acceptors, with that 'n' and where val is chosen as follows:

- if the proposer has gotten any (n_{prev}, val_{prev}) pairs, it must choose the val_{prev} that went with the highest n_{prev}.
- else it can pick anything.



P2 ko
mile
(6, (5,1))
(n_{prev}, value)
highest
n_{prev} = 5
uske saath ka
val_{prev} = 1