CSE P 552

Distributed Systems

Week 3

- why forward get operations to backup?
  - not linearizable — old primary may not still be primary
  - get requests "write" to the seq num cache! — primary crashes, backup won't know what to retransmit
- What messages need a view number?
  - clients?
  - if you receive a msg w/ a different view #, ignore the message

- at most 1 primary that can execute requests
  - new primary is old backup

✓ clock skew on pingtimer

- backup ack state transfer complete?
- backup crashes, primary keep going?

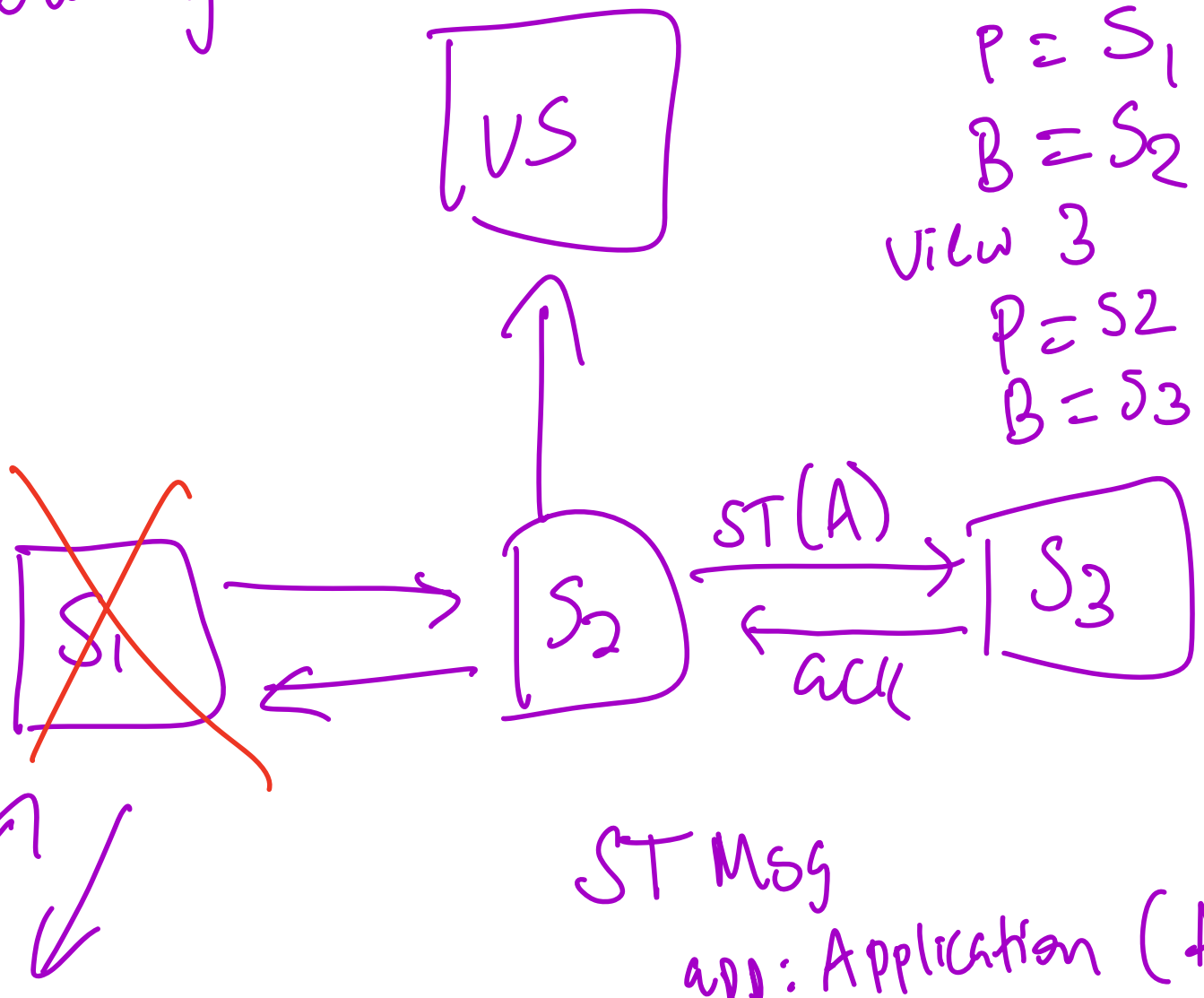
✓ deltas? ✓ nacks vs drops

✓ server timers? - usings during ST

- continue executing during ST?

- When should a backup say "no"

"piggybacking"



View 2
P = S1
B = S2
View 3
P = S2
B = S3

ST Msg
app: Application (AMOApplication)
viewNum

Failures during State transfer

- drop ST ack ——— sender re-request
  receiver re-respond
- drop ST ——— retransmit
- delayed ST

Knowledge

- K_N("s_2 received ST in view 3")
  $\equiv$ N knows that "..."
- D("...") if you had bird's eye view then you would know "..."
- E("...") everyone knows "..."

"bird's eye view"

- to take an action based on a fact being true some node must know that fact

- Messages communicate knowledge
- convert $D(\varphi)$ into $K_W(\varphi)$

$D(E(\text{current view\#} \geq 10))$

$K_{VS}(E(\text{current view\#} \geq 10))$

$E(E(E(\cdots)))$

$D_G(\ldots)$ bird's eye view of nodes in G would know ...

$E_G(\ldots)$ every node in G knows ...

Protocol problems w/ Lab 2

- VS single source of failure
  - ugly
- re execute every request twice
- poorly timed crashes of P/B (during ST) lead to stuckness
- no parallel execution of requests
- unrealistic message sizes

- finite set of servers

  - no reboot

Paxos

- Leslie Lamport

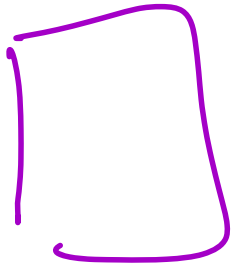
  Part-time Parliament

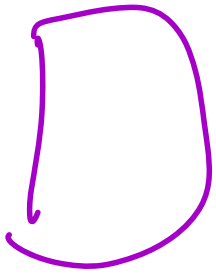
  Paxos Made Simple

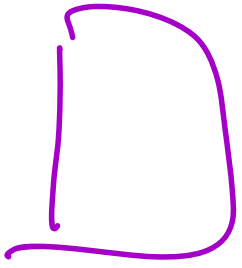
Single-decree Paxos

- One decision

MultiPaxos = Single decree repeated







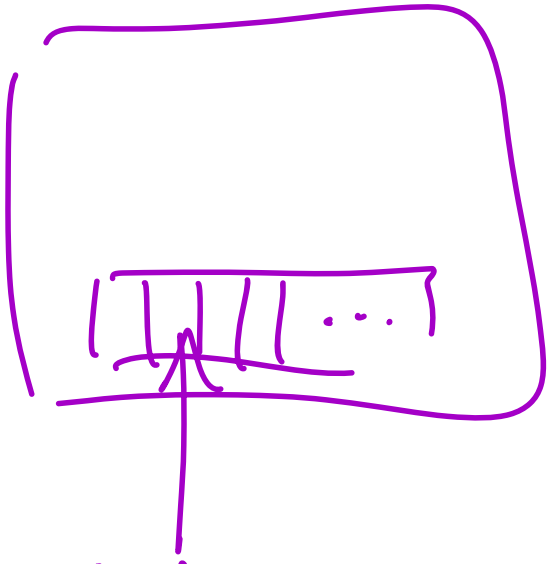
majority voting

2 out of 3

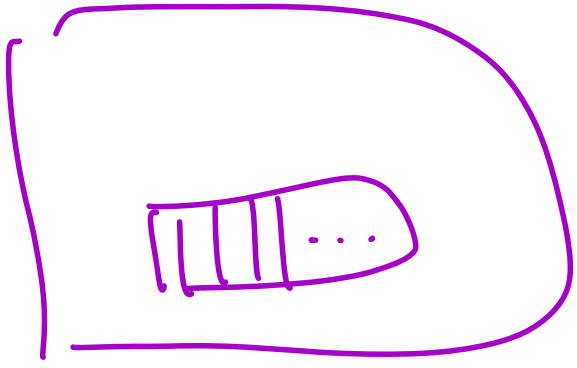
tolerates 1 failur

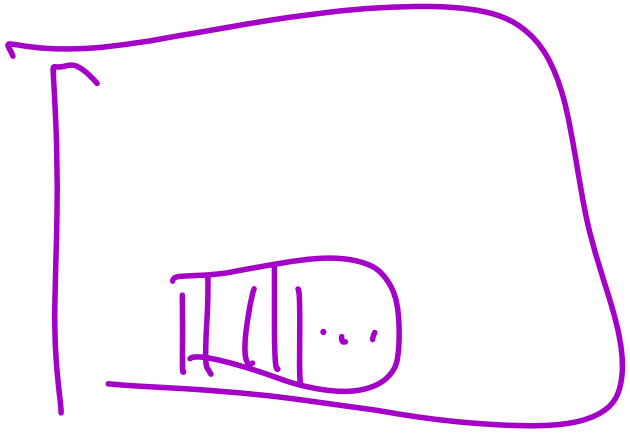
Multi-Paxos
log replication

fixed # of nodes
$E(\vee)$



index into log
"slot"

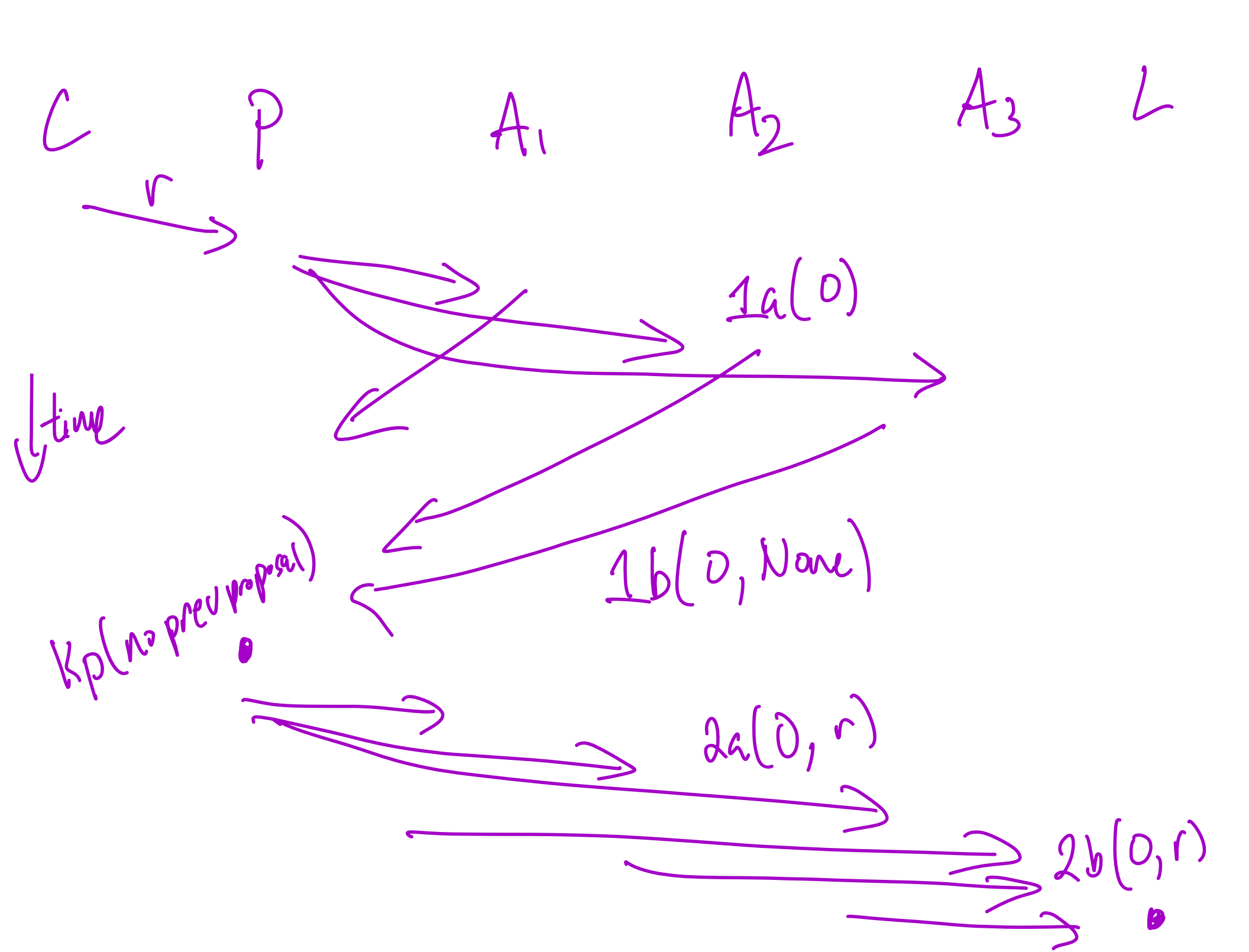




Clients want to execute requests

servers

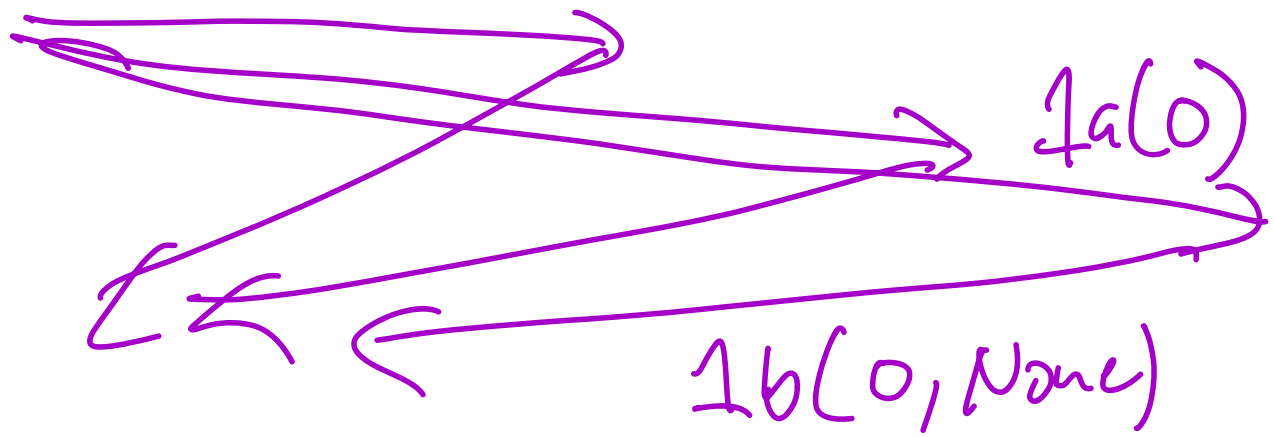
- Proposers: listen to clients and try to get their requests executed
- Acceptors: "accept" proposals (voting)
- Learners: learn what got decided (count votes)

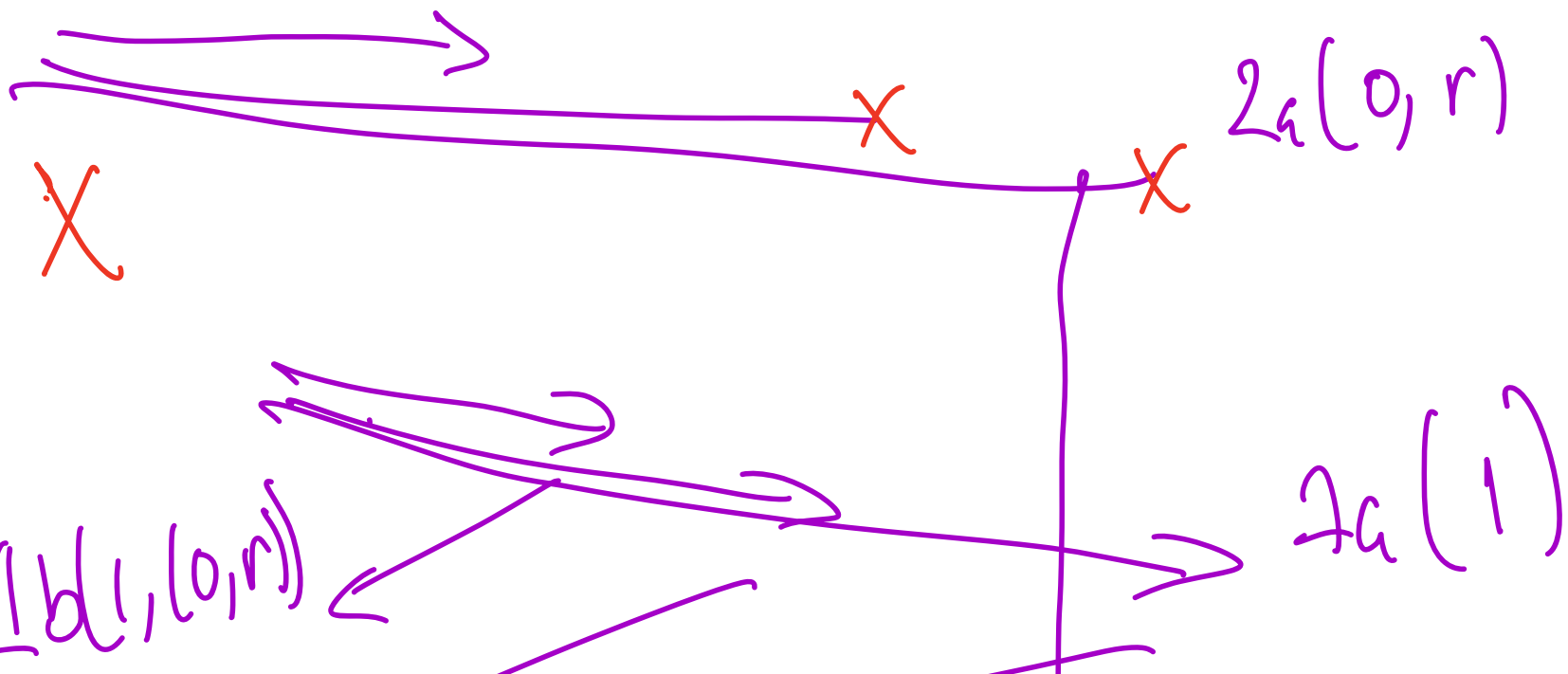

C
P
A1
A2
A3
L
r
1a(0)
time
1b(0, None)
Kp(no prev proposal)
2a(0, r)
2b(0, r)

$KL(v$
was decided)

ballot $= (O, P_1) \quad (O, P_2)$

P_1 P_2 A_1 A_2 A_3 L






26(1, Now)
26(2, r)

Phase 1

1a : Proposer → Acceptor

- ballot : (ballot_num, proposer_id)

1b : Acceptor → Proposer

- ballot
- highest balloted proposal accepted by this acceptor in a smaller ballot

(sending 1b is a promise not to accept any proposals in smaller ballots

ever again)

Phase 2 (Proposer waits for a maj- of 1b msgs)

2a: Proposer → Acceptor

- ballot
- proposed request | value
  - highest balloted proposal on any received 1b
  - or any value if all are None