

CPSC 416 Distributed Systems

Winter 2022 Term 2 (March 2, 2023)

Tony Mason (fsgeek@cs.ubc.ca), Lecturer



PMMC Walk-Through



“Paxos Made Moderately Complex” Made Moderately Simple

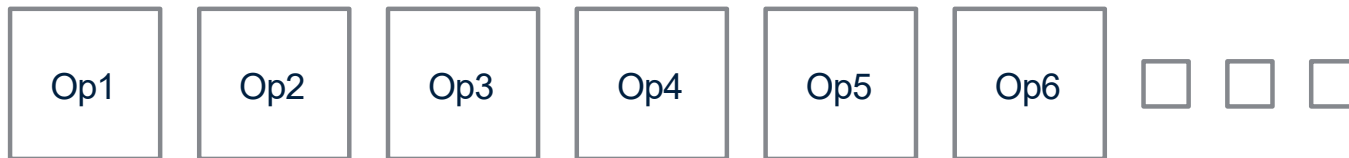
Source: [University of Washington CSE 452](#)

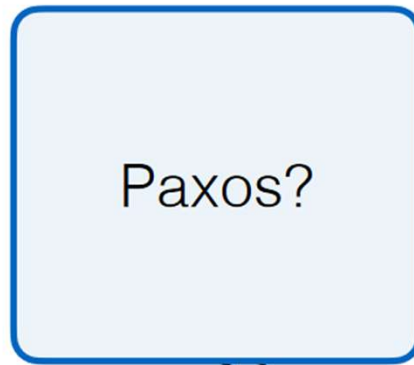
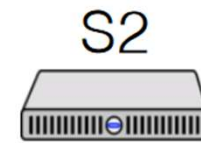
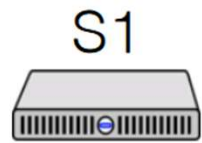


State machine replication

Reminder: want to agree on order of ops

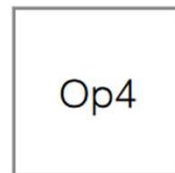
Can think of operations as a log





Put k1 v1

Put k2 v2



Paxos



Phase 1

- Send prepare messages
- = - Pick value to accept

Phase 2

- Send accept messages

Can we do better?



Phase 1: “leader election”

- Deciding whose value we will use

Phase 2: “commit”

- Leader makes sure it's still leader, commits value

What if we split these phases?

- Lets us do operations with one round-trip

Roles in PMMC



Replicas (like learners)

- Keep log of operations, state machine, configs

Leaders (like proposers)

- Get elected, drive the consensus protocol

Acceptors (*simpler* than in Paxos Made Simple!)

- “Vote” on leaders

A note about ballots in PMMC

(leader, seqnum) pairs

Isomorphic to the system we discussed earlier



① 0, 4, 8, 12, 16, ...

② 1, 5, 9, 13, 17, ...

③ 2, 6, 10, 14, 18, ...

④ 3, 7, 11, 15, 19, ...

A note about ballots in PMMC

(leader, seqnum) pairs

Isomorphic to the system we discussed earlier



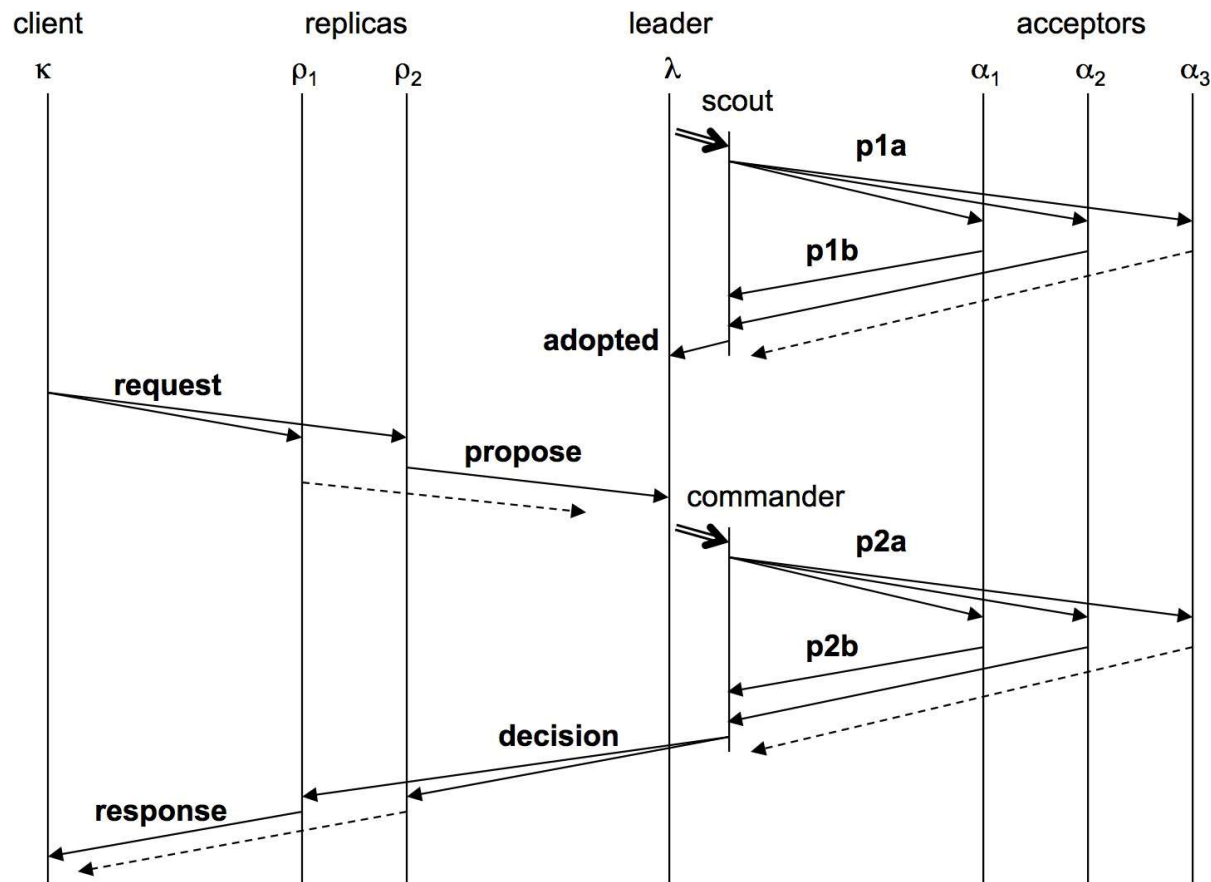
① 0.0, 1.0, 2.0, 3.0, 4.0, ...

② 0.1, 1.1, 2.1, 3.1, 4.1, ...

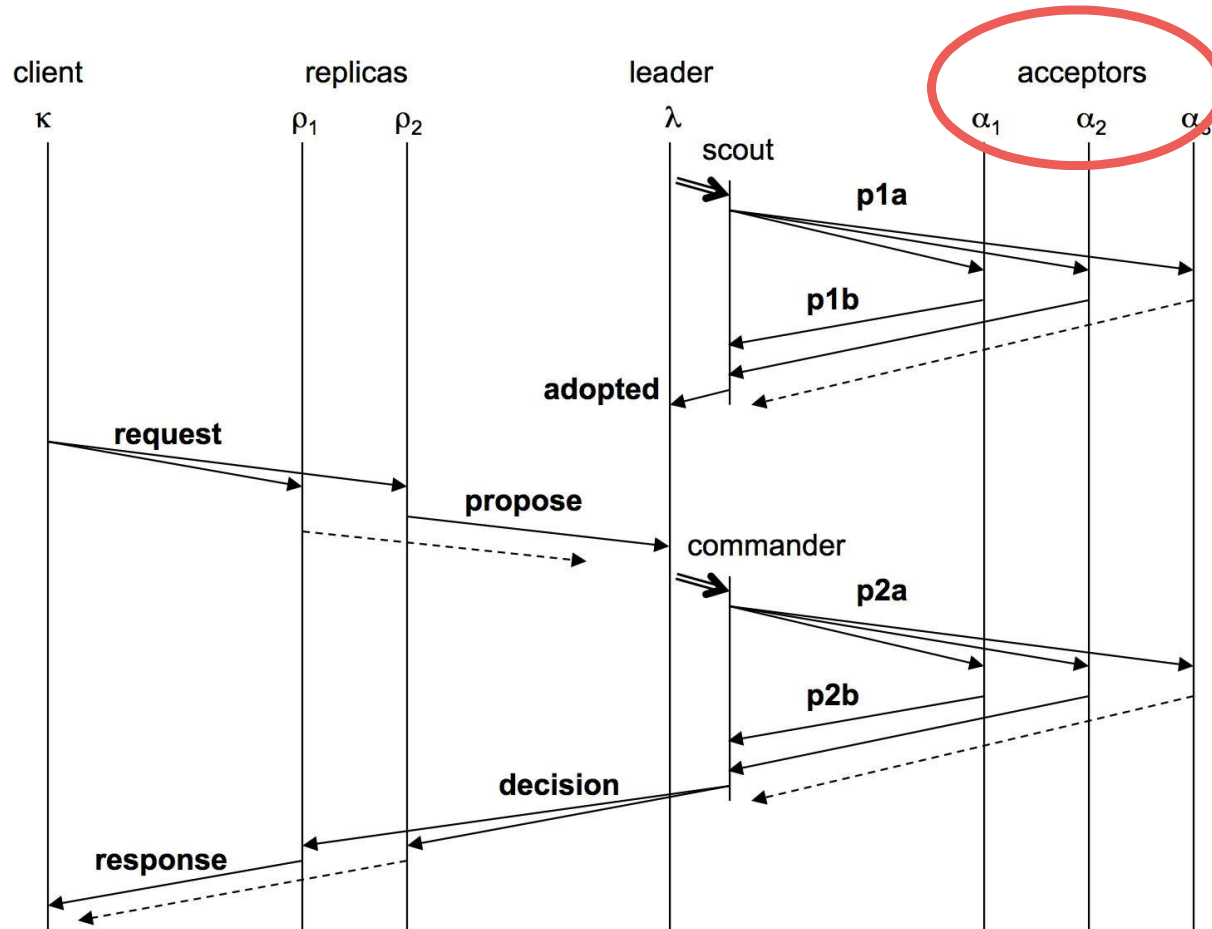
③ 0.2, 1.2, 2.2, 3.2, 4.2, ...

④ 0.3, 1.3, 2.3, 3.3, 4.3, ...

Paxos Made Moderately Complex Made Simple



Paxos Made Moderately Complex Made Simple



Acceptors

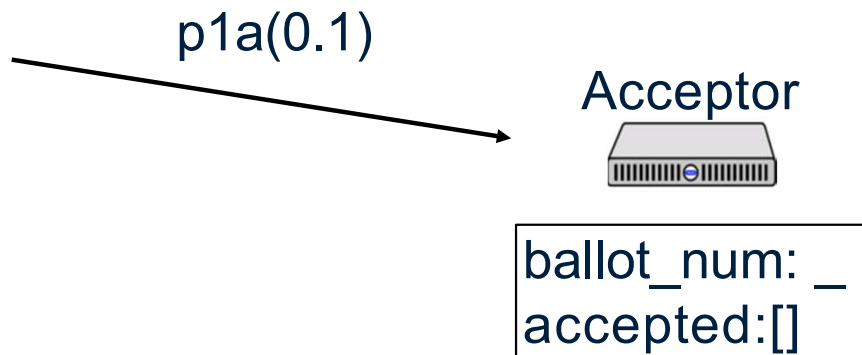


Acceptor

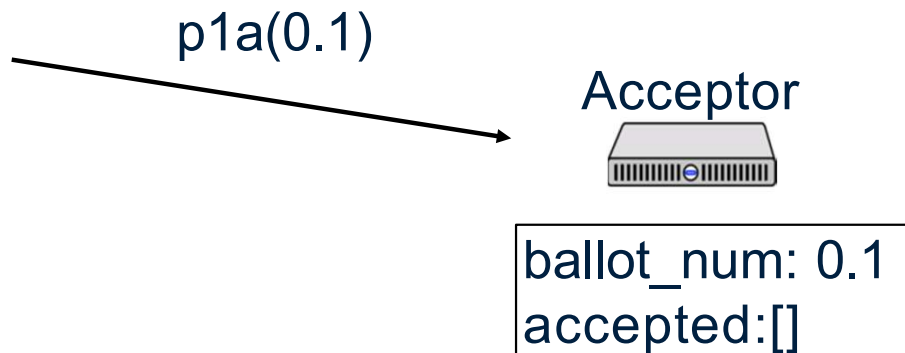


```
ballot_num: _  
accepted: []
```

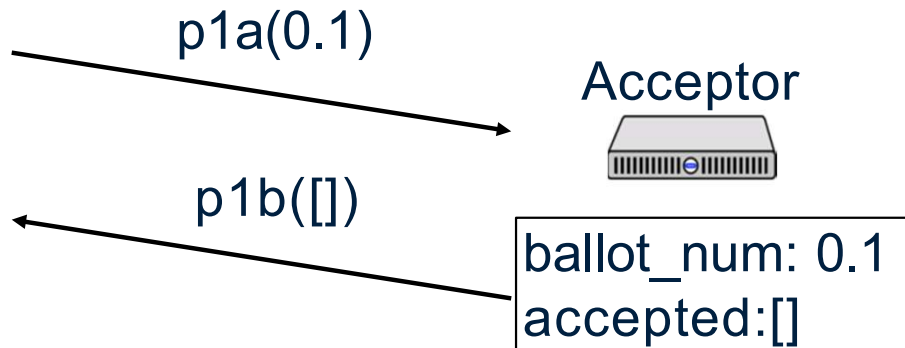
Acceptors



Acceptors



Acceptors



Acceptors

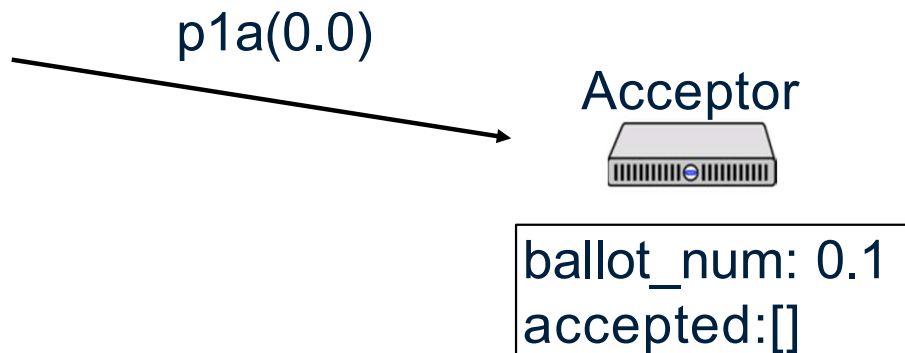


Acceptor

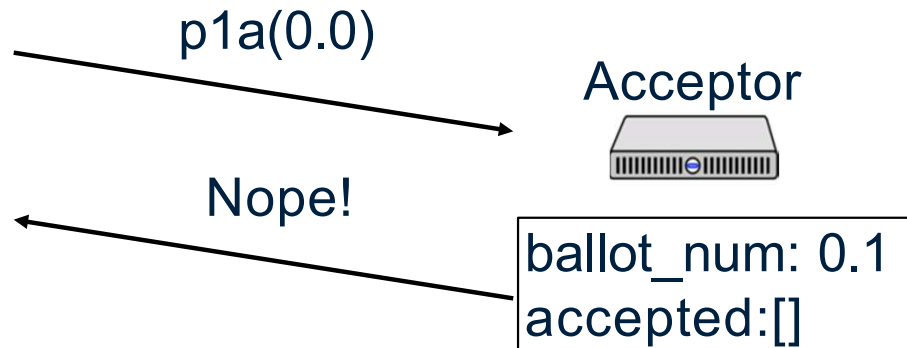


```
ballot_num: 0.1  
accepted: []
```

Acceptors



Acceptors



Acceptors

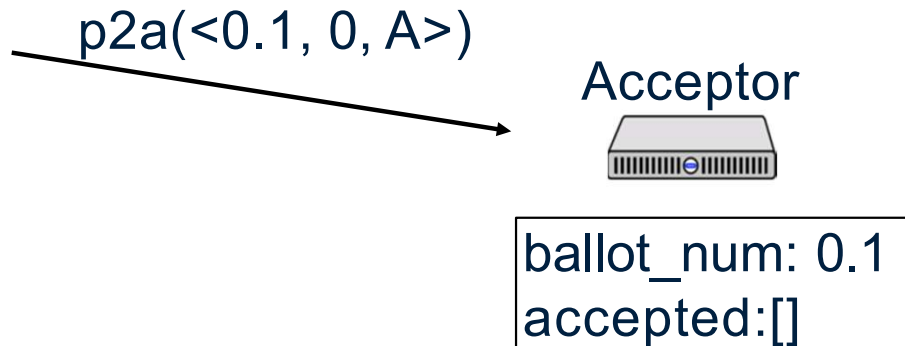


Acceptor

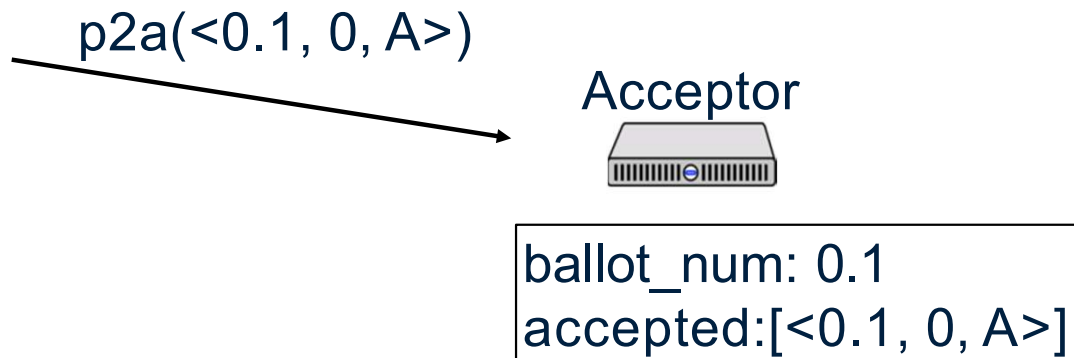


```
ballot_num: 0.1  
accepted: []
```

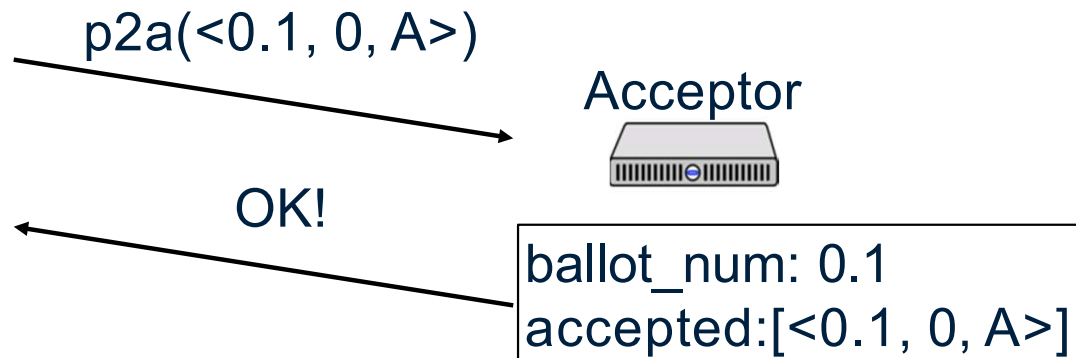

Acceptors



Acceptors



Acceptors



Acceptors

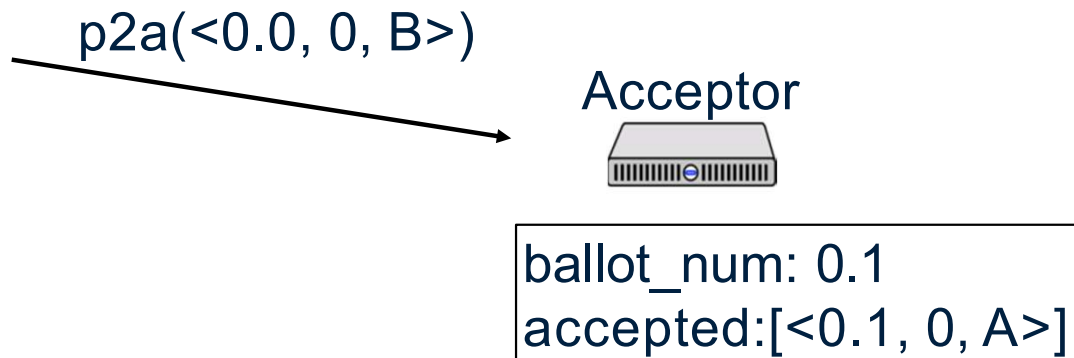


Acceptor

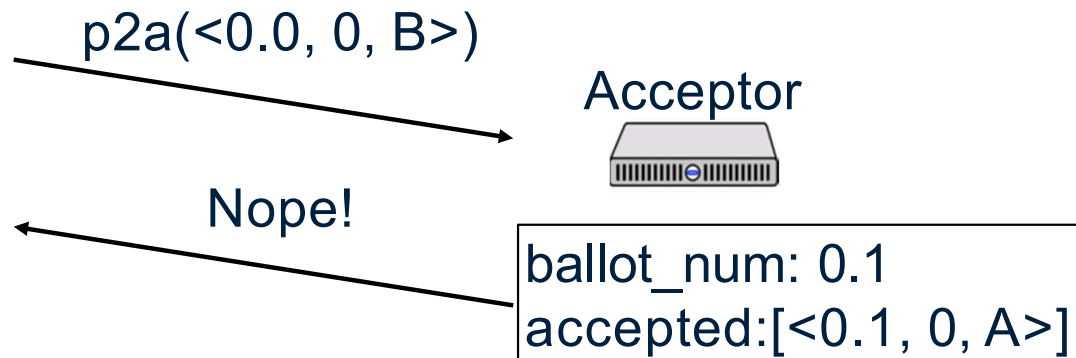


```
ballot_num: 0.1  
accepted:[<0.1, 0, A>]
```

Acceptors



Acceptors



Acceptors



Acceptor



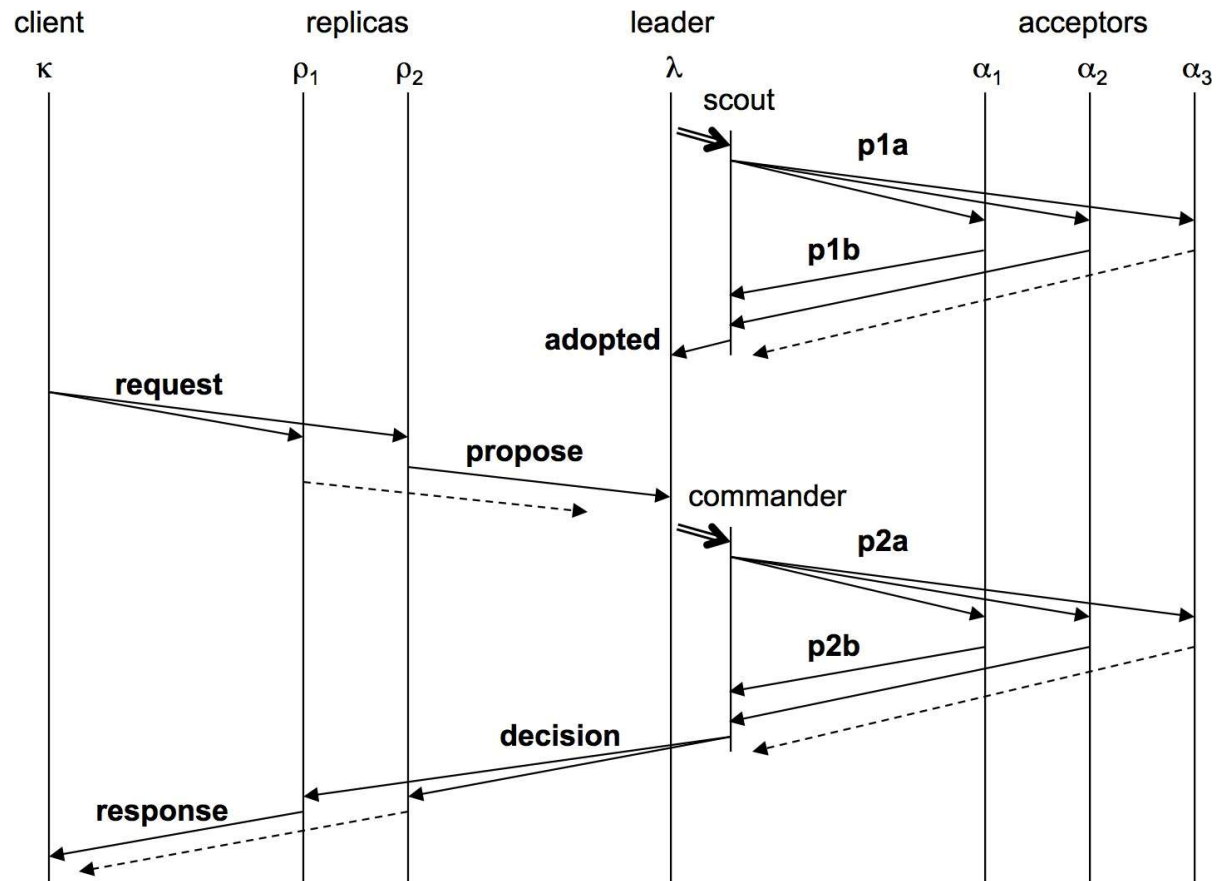
```
ballot_num: 0.1  
accepted:[<0.1, 0, A>]
```

Acceptors

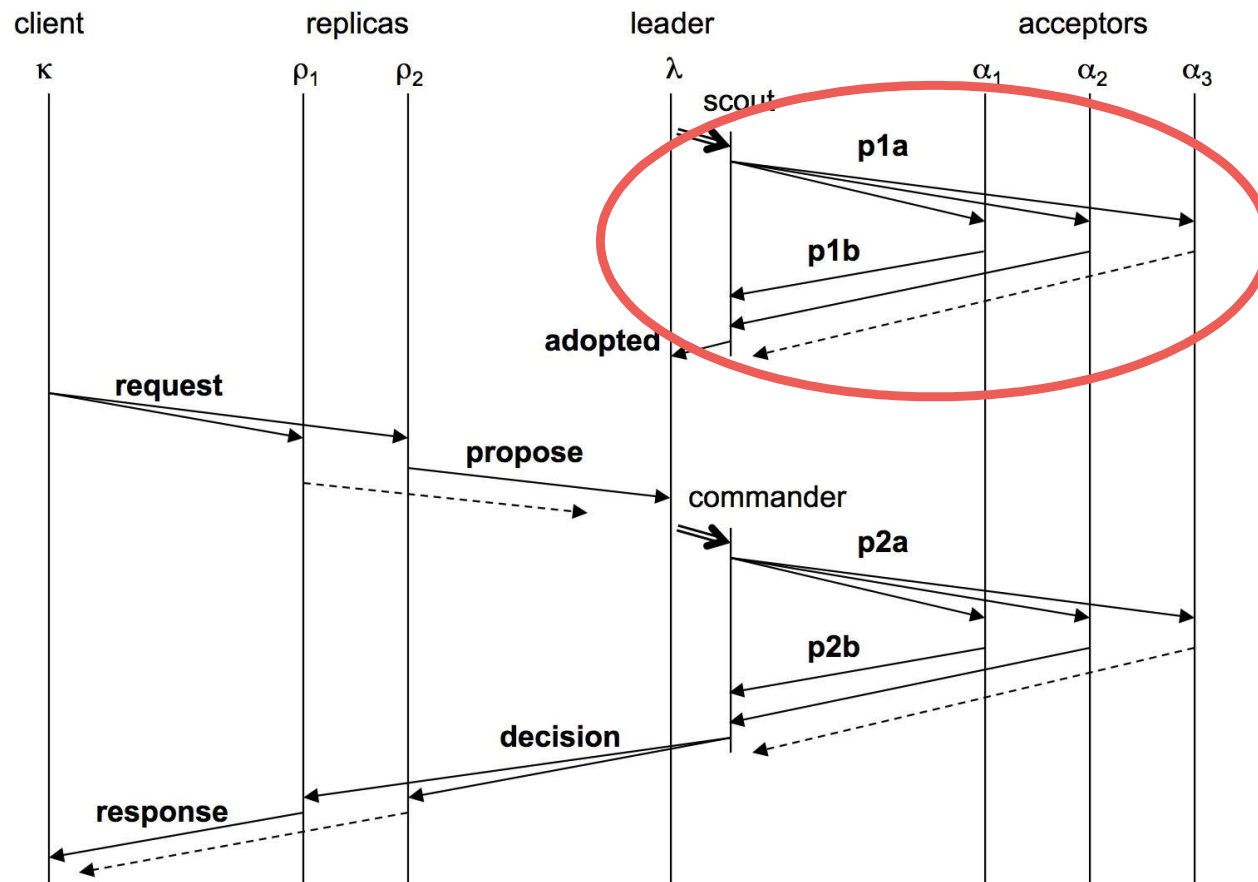
- Ballot numbers increase
- Only accept values from current ballot
- Never remove ballots
- If a value v is chosen by a majority on ballot b , then any value accepted by any acceptor in the same slot on ballot $b' > b$ has the same value



Paxos Made Moderately Complex Made Simple



Paxos Made Moderately Complex Made Simple



Leader: Getting Elected

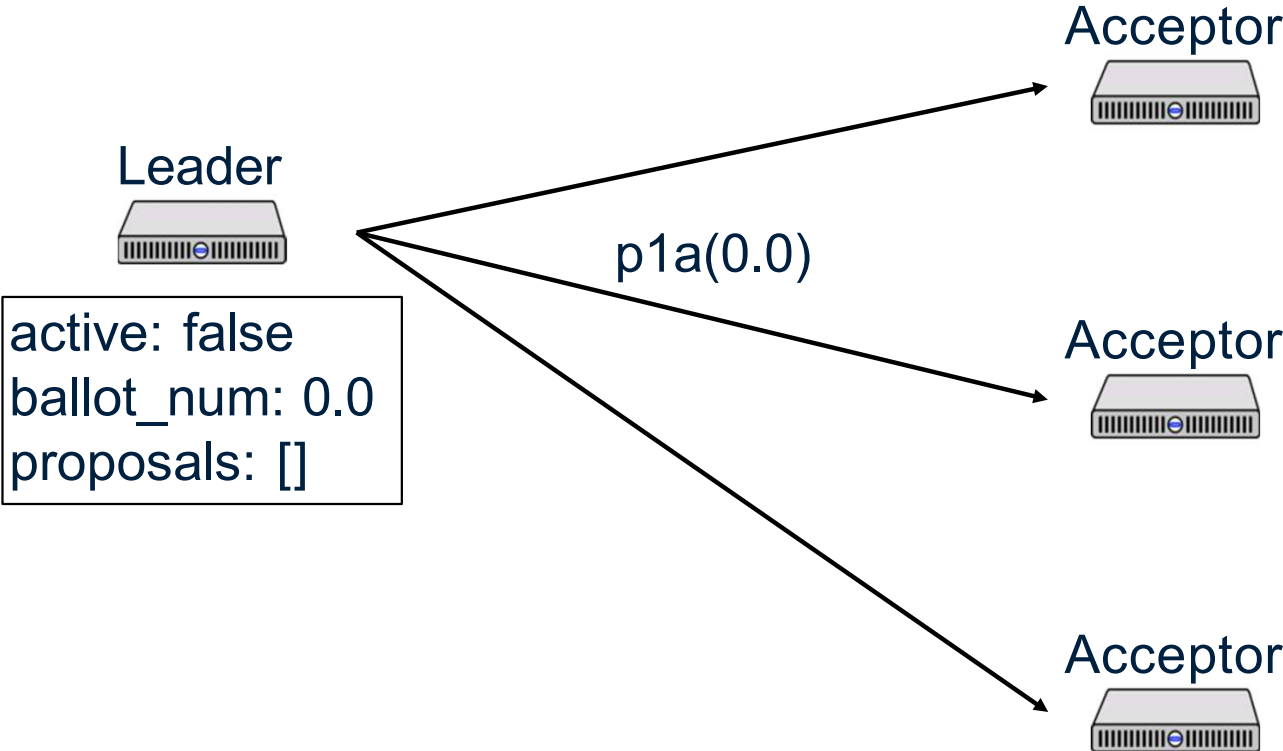
Leader



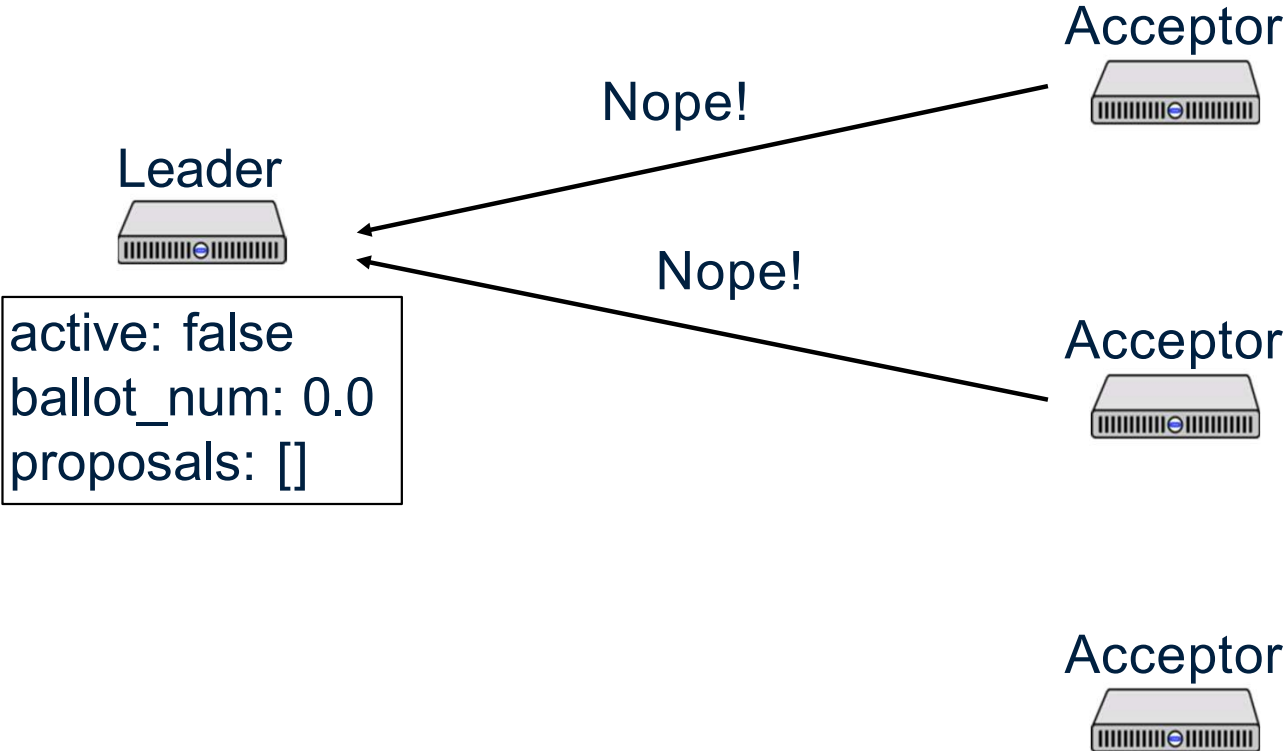
```
active: false  
ballot_num: 0.0  
proposals: []
```



Leader: Getting Elected



Leader: Getting Elected



Leader: Getting Elected

Leader



```
active: false  
ballot_num: 1.0  
proposals: []
```

Acceptor



Acceptor



Acceptor



Leader: Getting Elected

Leader



```
active: false  
ballot_num: 1.0  
proposals: []
```

Or...

Acceptor



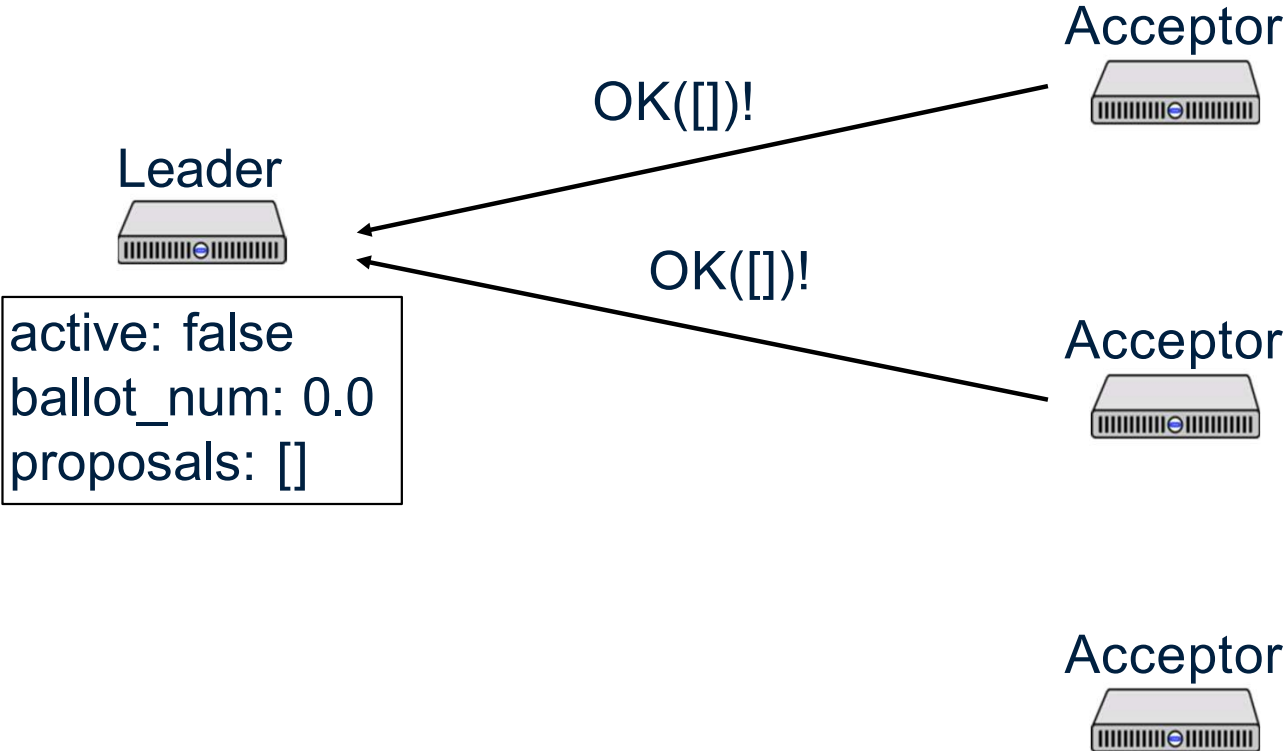
Acceptor



Acceptor



Leader: Getting Elected



Leader: Getting Elected

Leader



```
active: true  
ballot_num: 0.0  
proposals: []
```

Acceptor



Acceptor



Acceptor



When to run for office

When should a leader try to get elected?

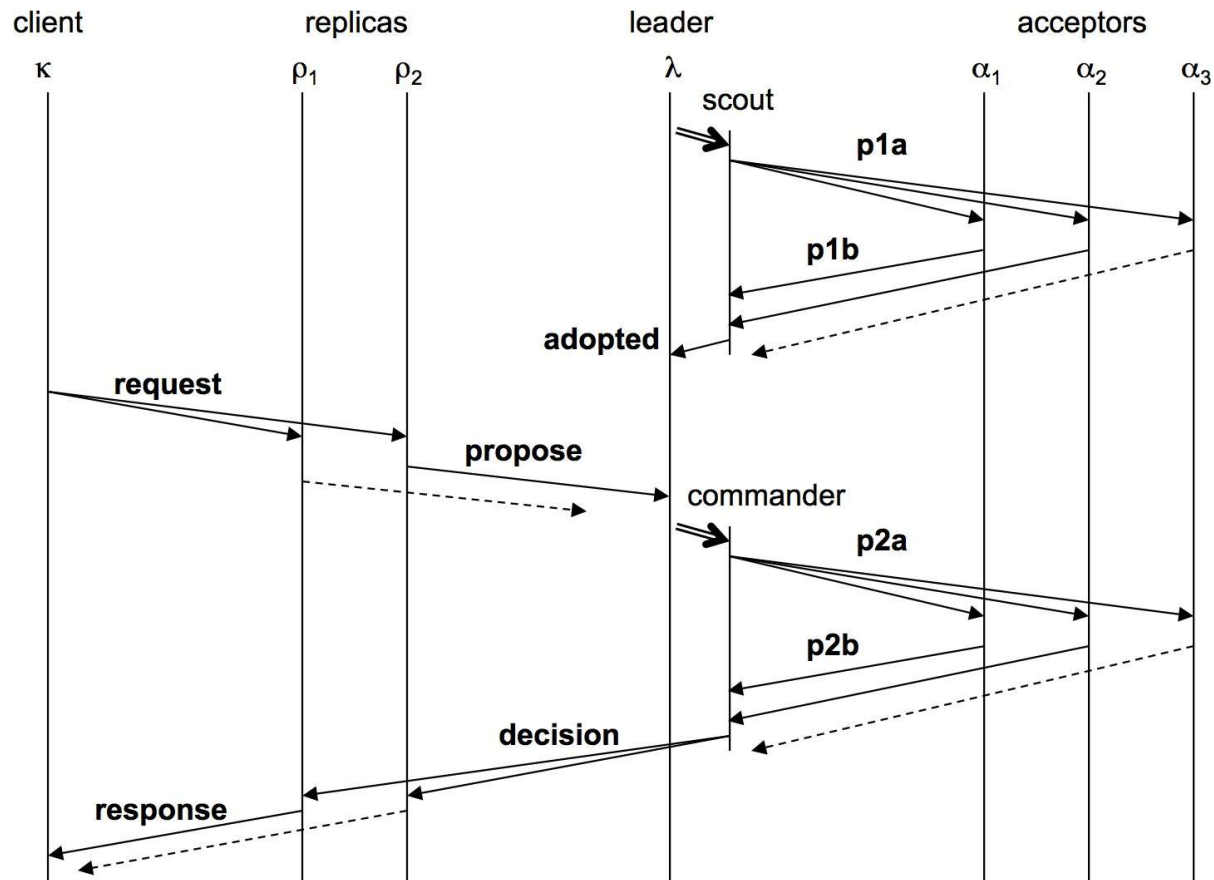
- At the beginning of time
- When the current leader seems to have failed

Paper describes an algorithm, based on pinging the leader and timing out

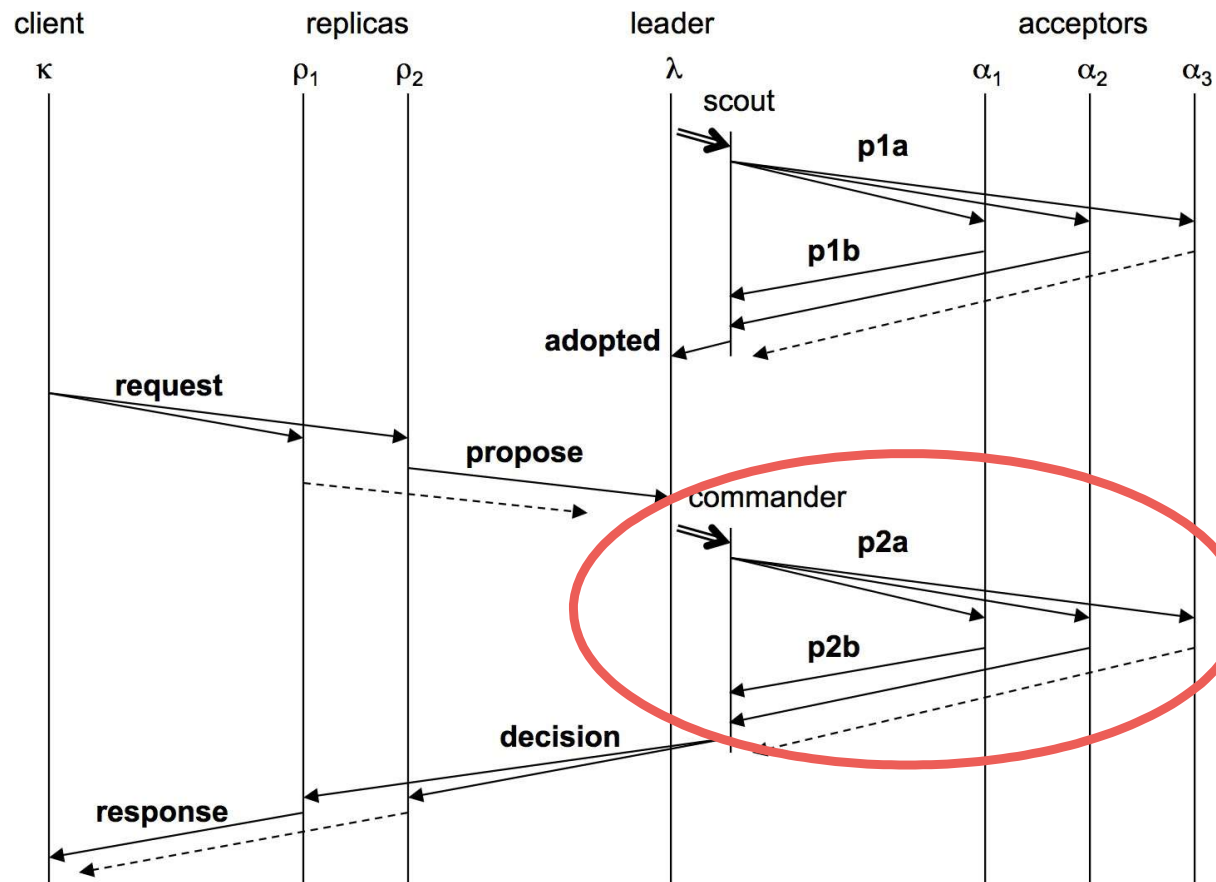
If you get preempted, don't immediately try for election again!



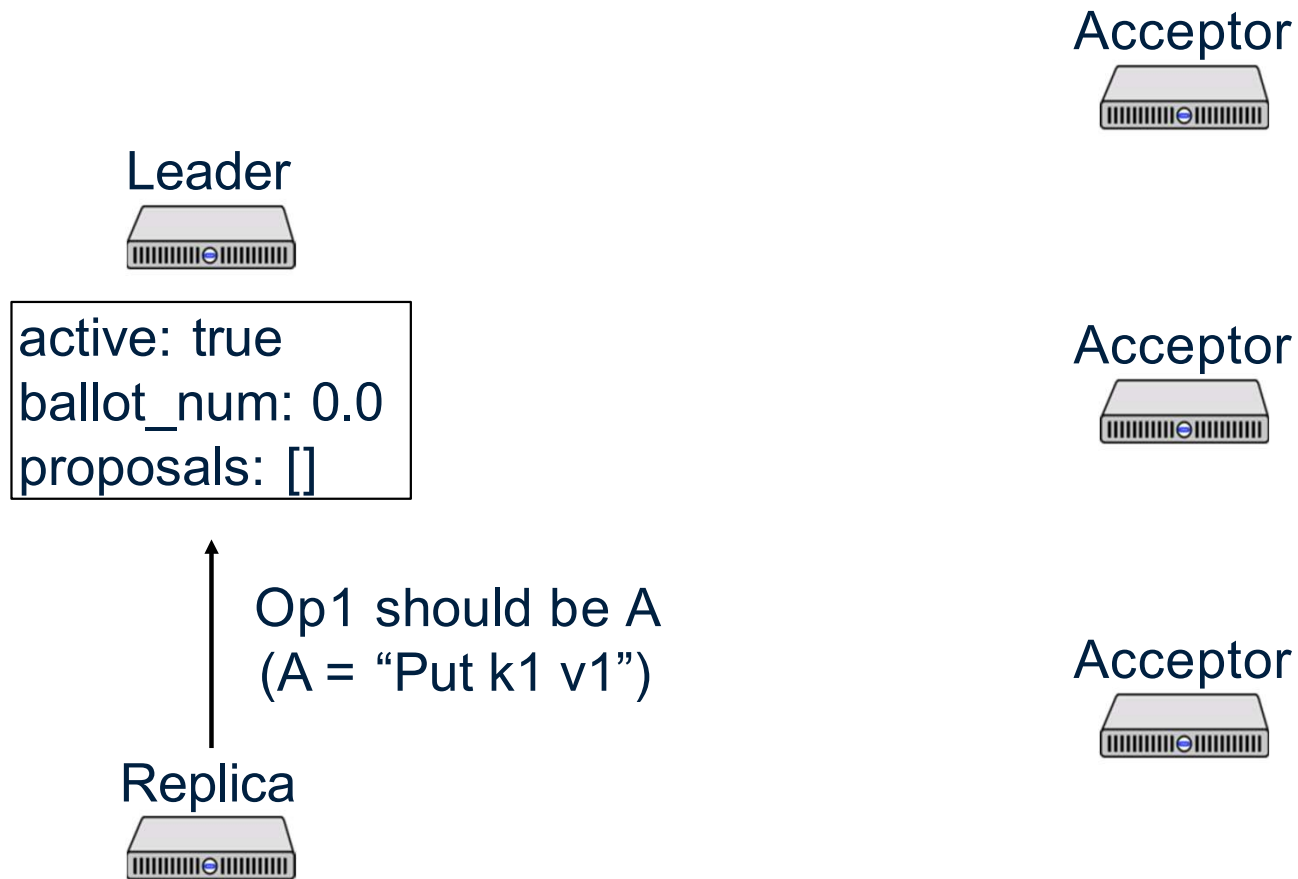
Paxos Made Moderately Complex Made Simple



Paxos Made Moderately Complex Made Simple



Leader: Handling proposals



Leader: Handling proposals

Leader



```
active: true  
ballot_num: 0.0  
proposals: [<1, A>]
```

Replica



Acceptor



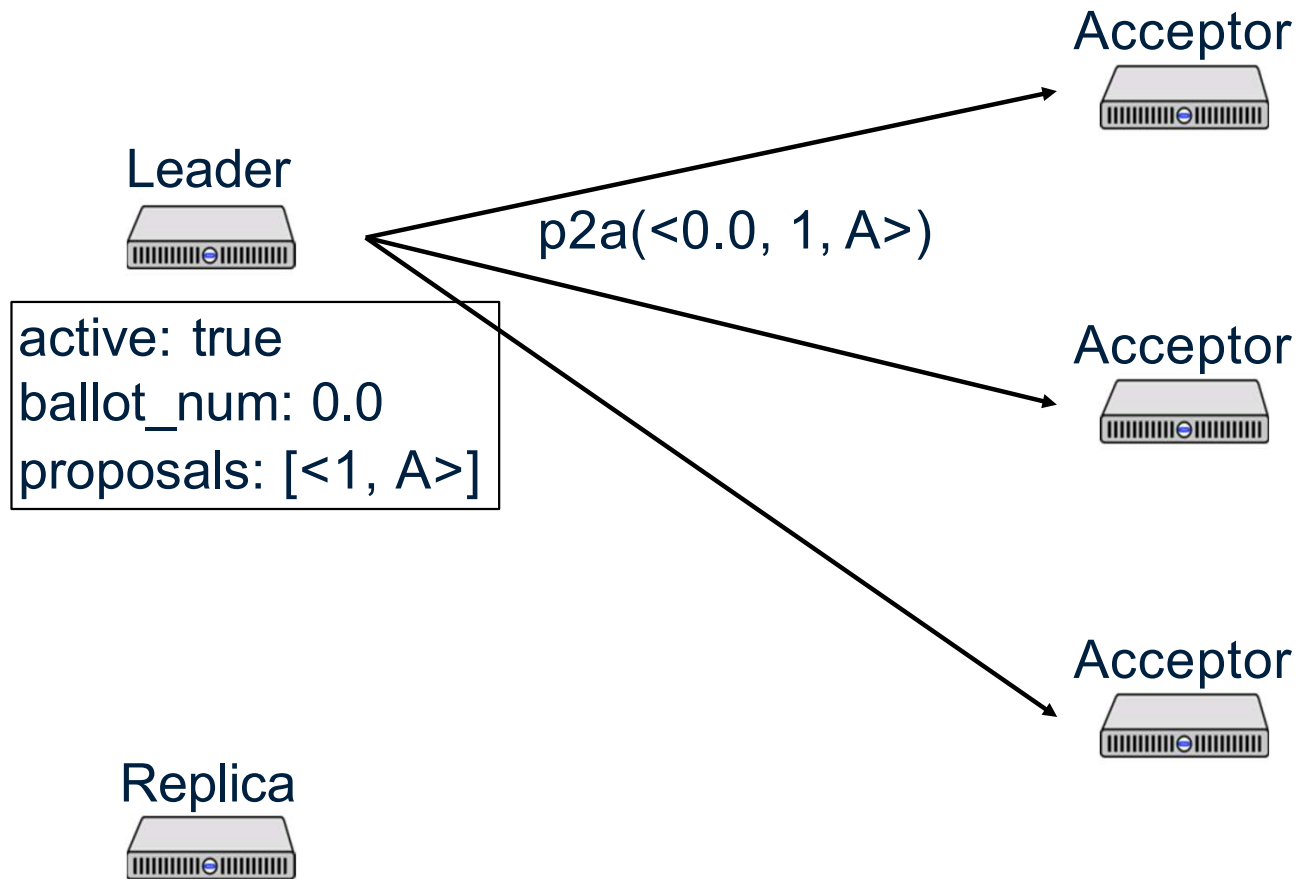
Acceptor



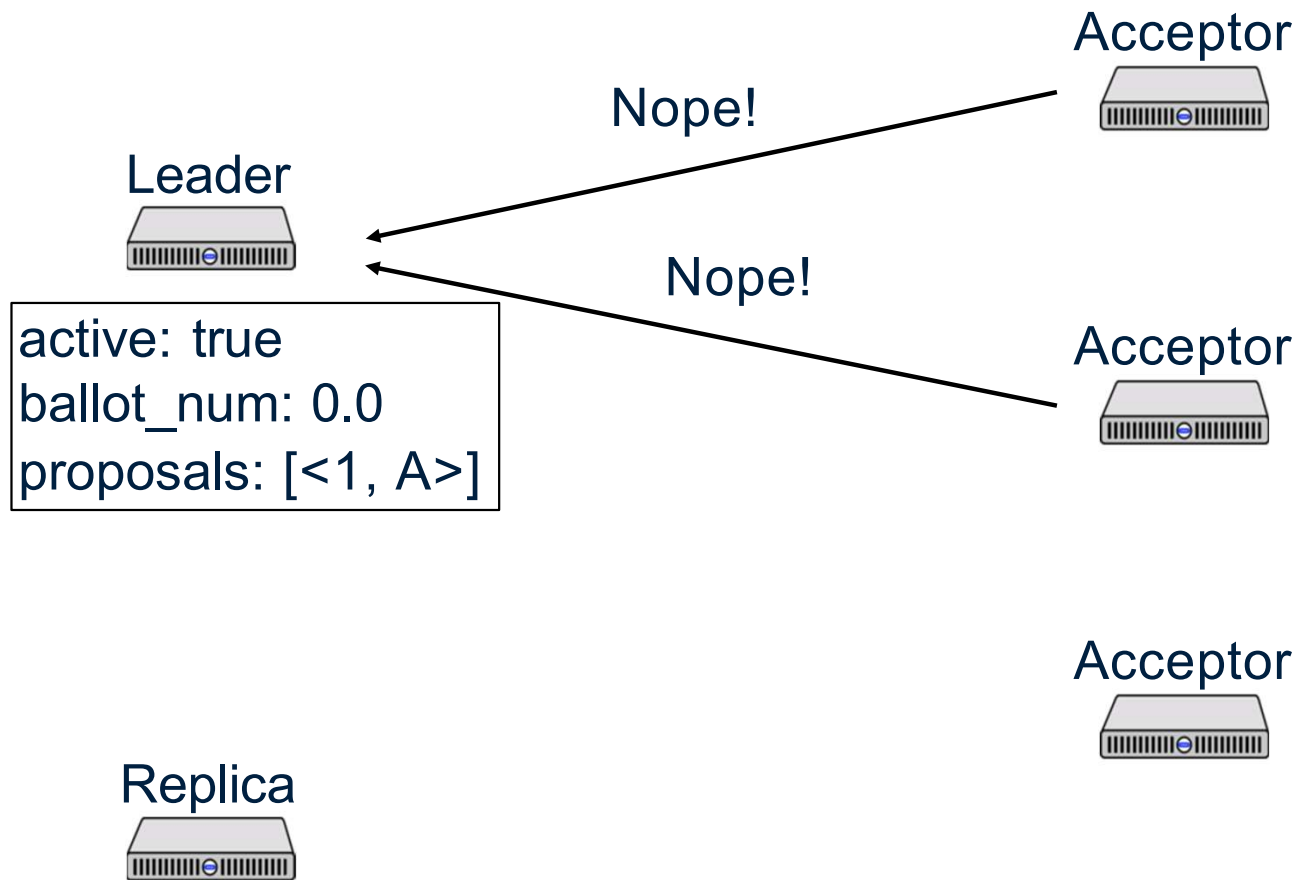
Acceptor



Leader: Handling proposals



Leader: Handling proposals



Leader: Handling proposals

Leader



```
active: false  
ballot_num: 0.0  
proposals: [<1, A>]
```

Replica



Acceptor



Acceptor



Acceptor



Leader: Handling proposals

Leader



```
active: false  
ballot_num: 0.0  
proposals: [<1, A>]
```

Or...

Acceptor



Acceptor



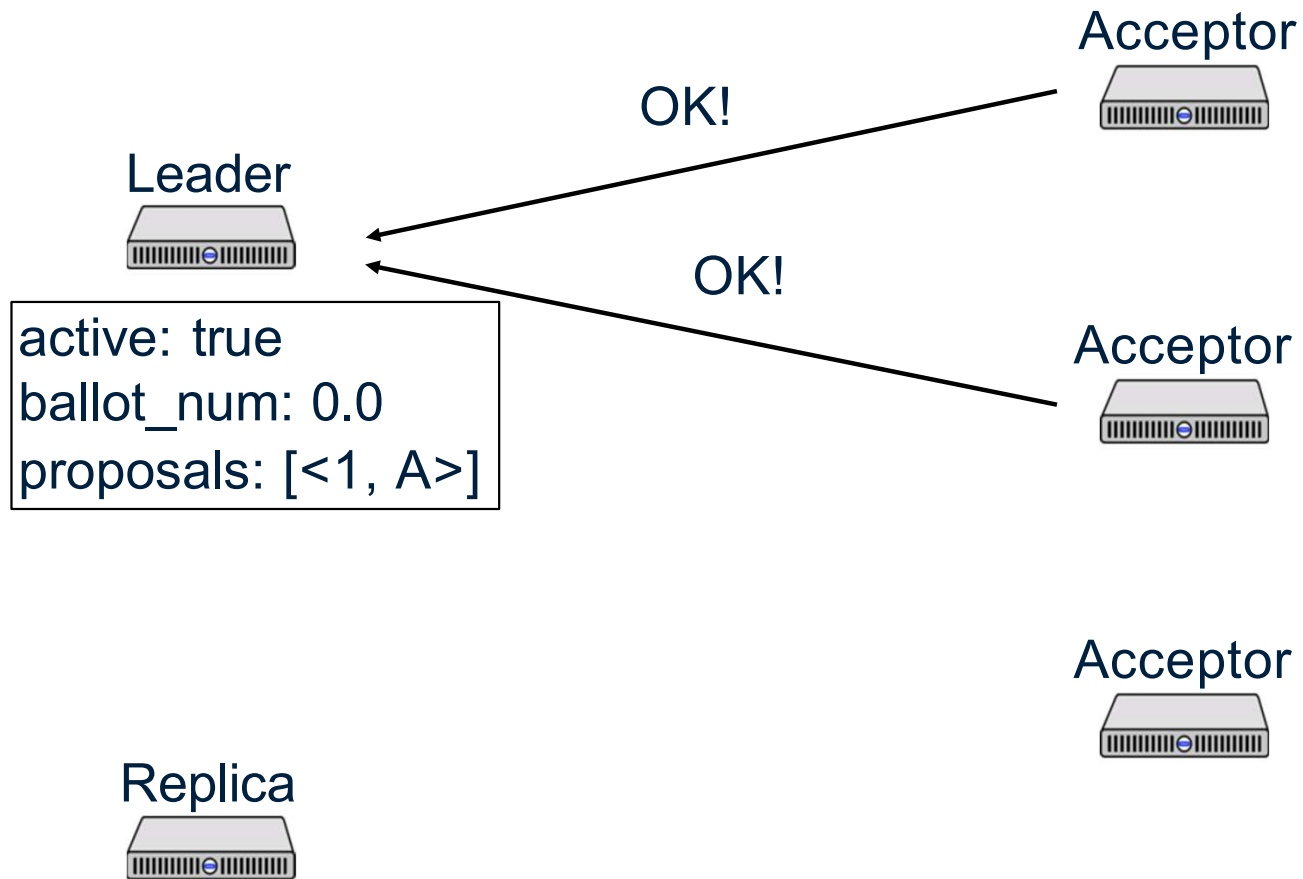
Acceptor



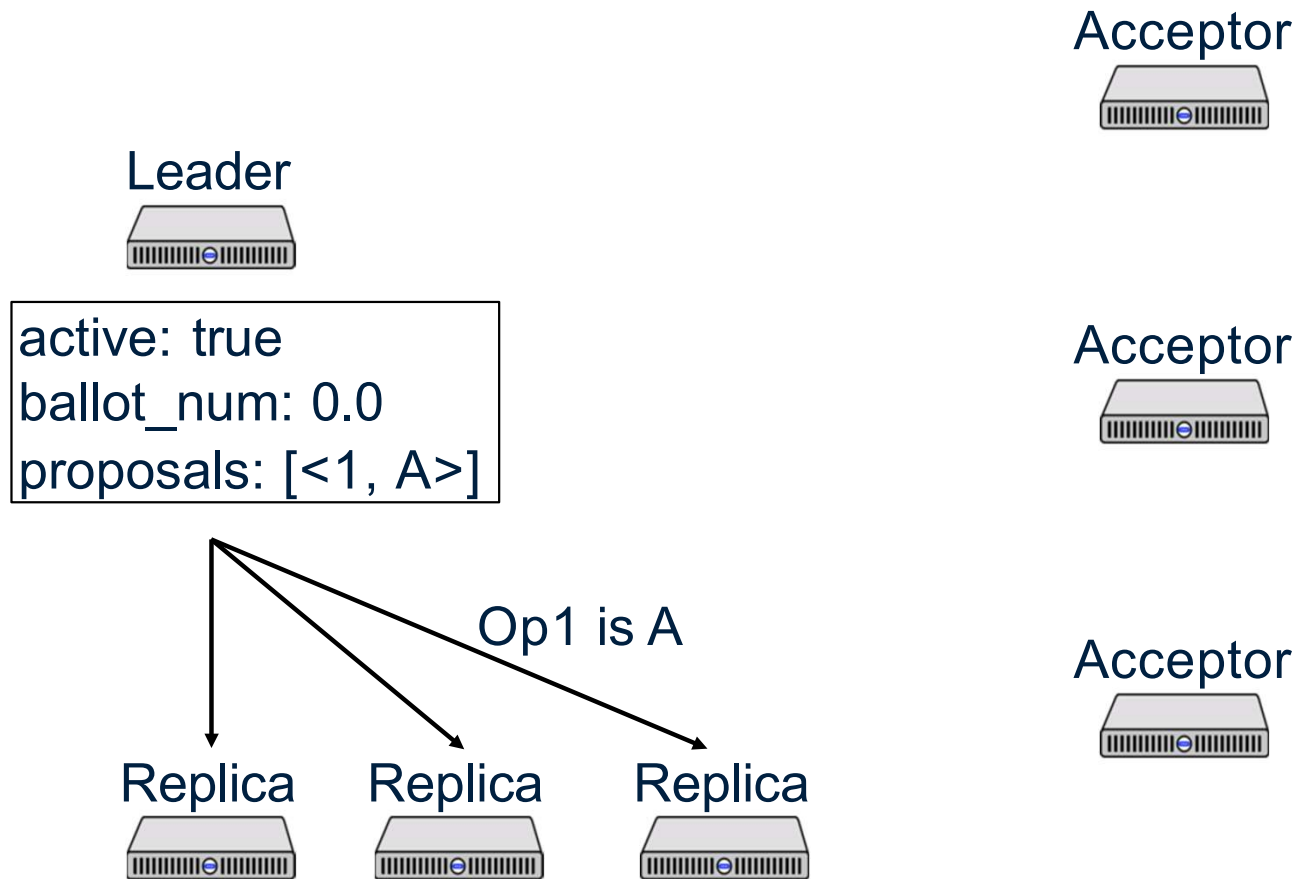
Replica



Leader: Handling proposals



Leader: Handling proposals



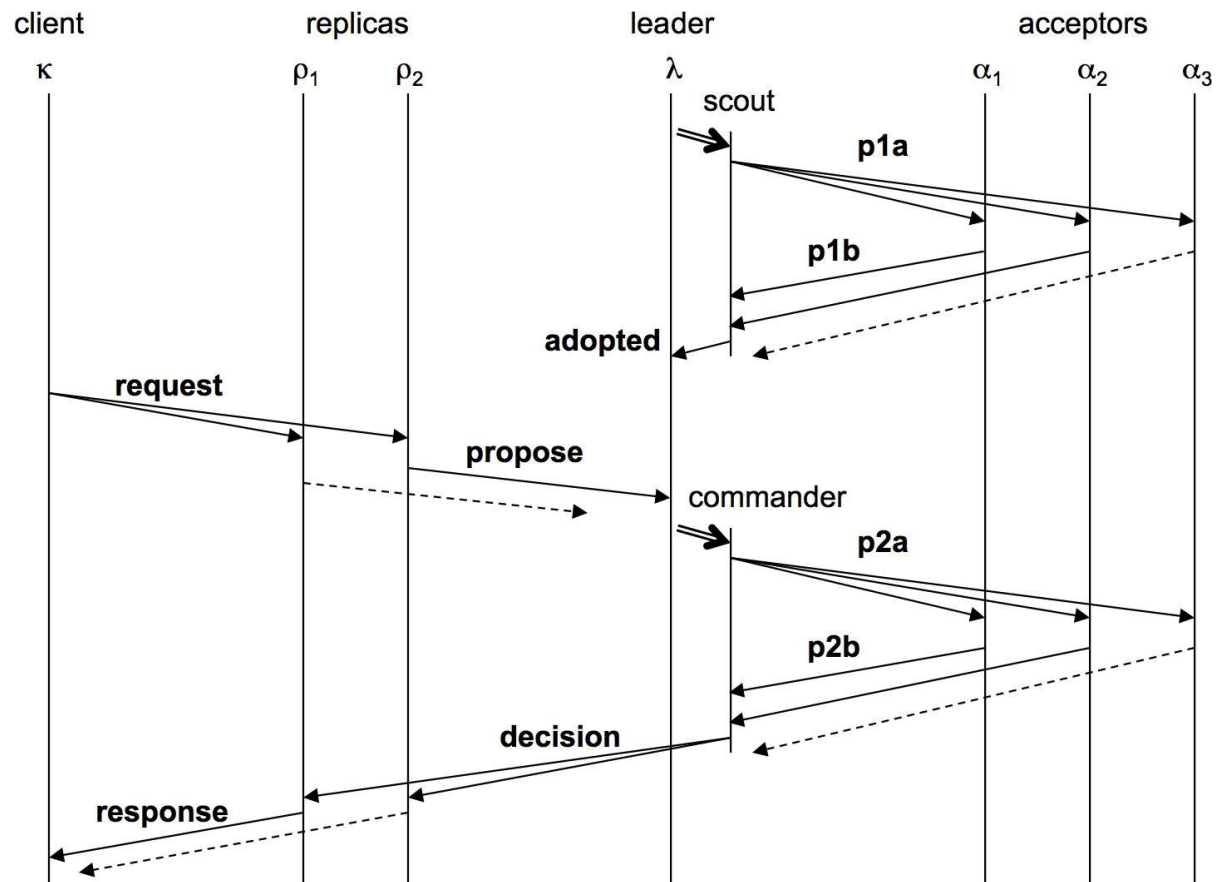
Leaders

Only propose one value per ballot and slot

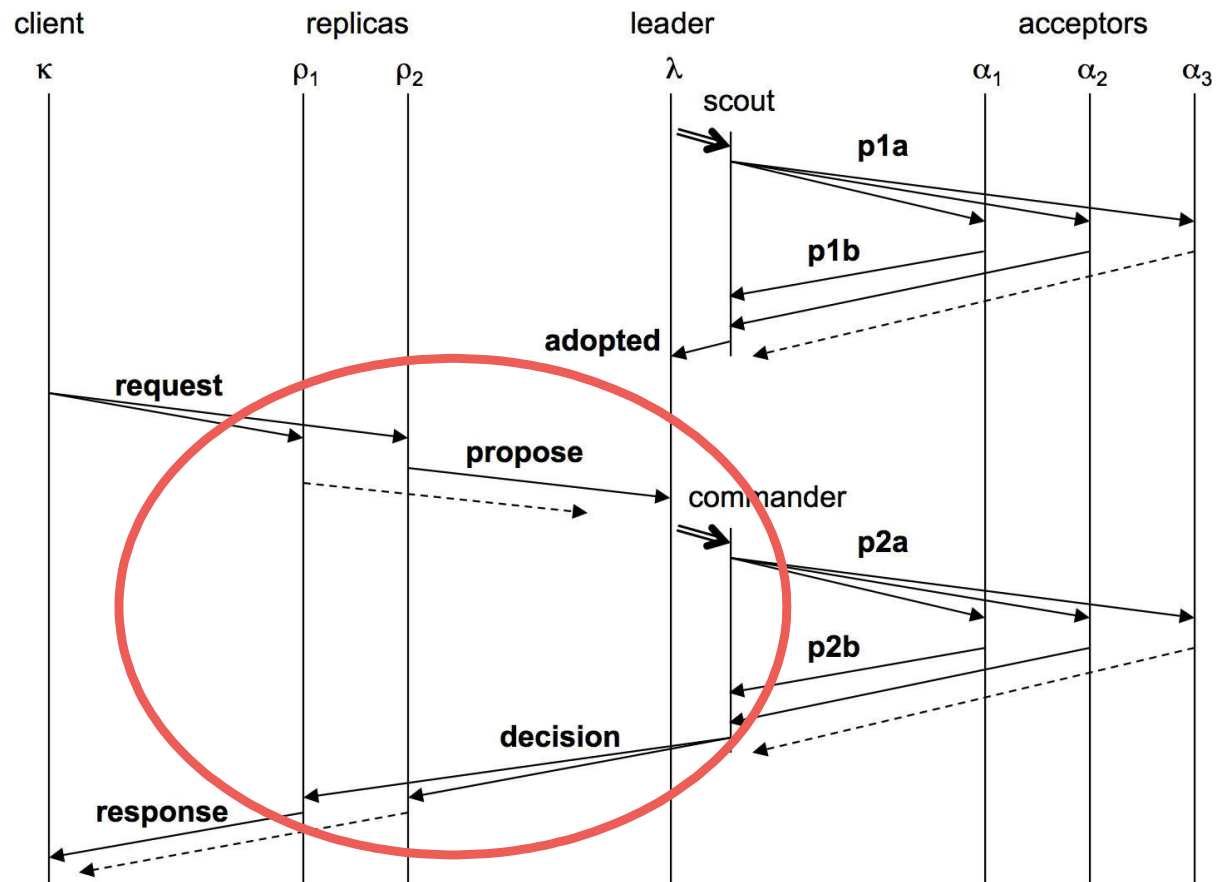
If a value v is chosen by a majority on ballot b , then any value proposed by any leader in the same slot on ballot $b' > b$ has the same value



Paxos Made Moderately Complex Made Simple



Paxos Made Moderately Complex Made Simple



Replicas



Replica



Put k1 v1

Put k2 v2



Replicas



Replica



slot_out

slot_in

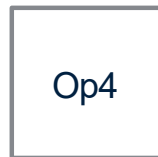


Put k1 v1

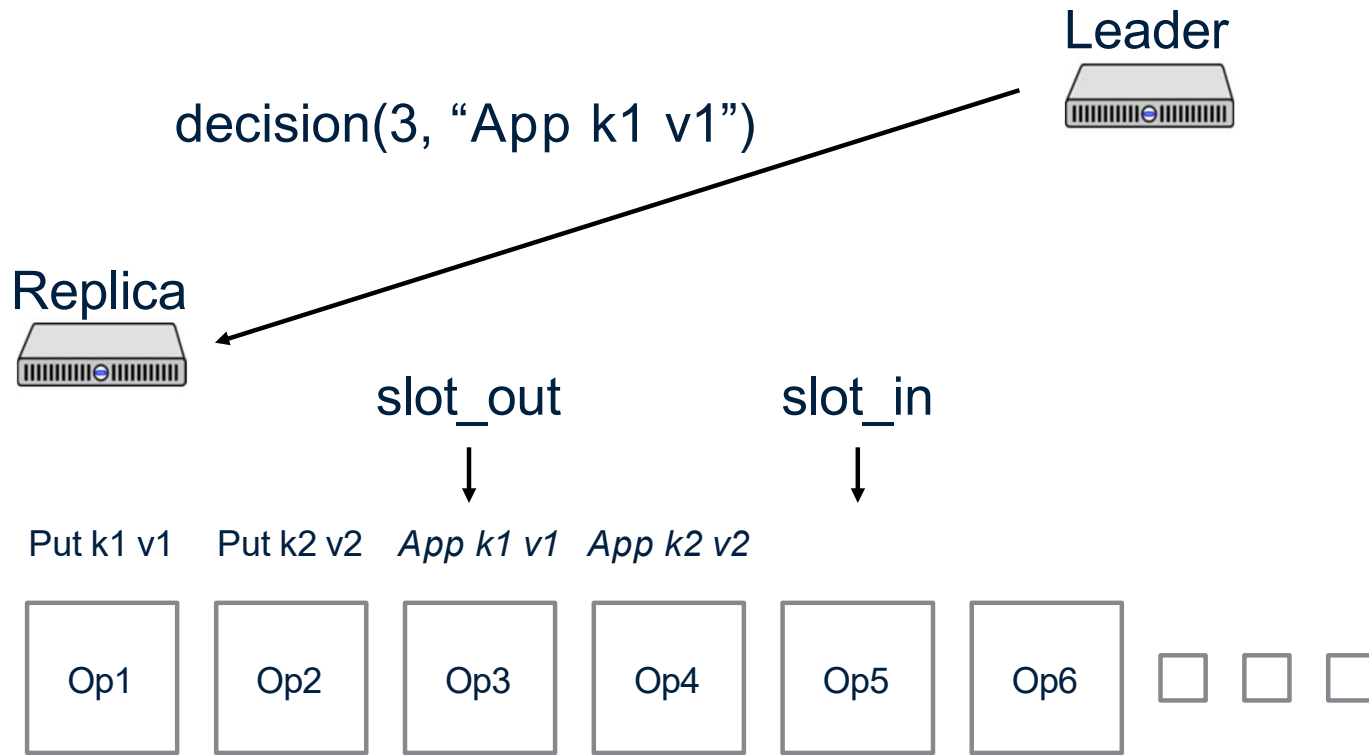
Put k2 v2

App k1 v1

App k2 v2

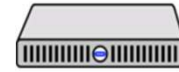


Replicas



Replicas

Leader



Replica



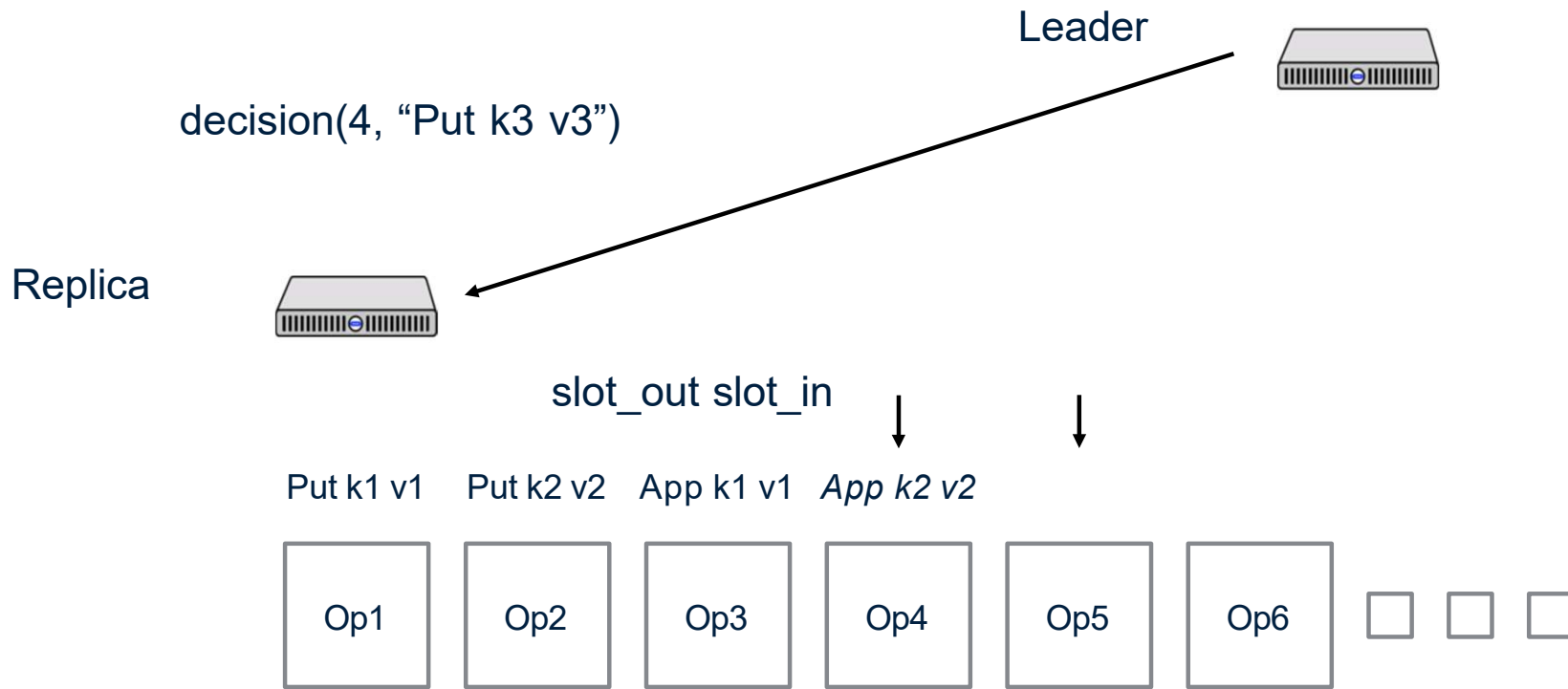
slot_out slot_in



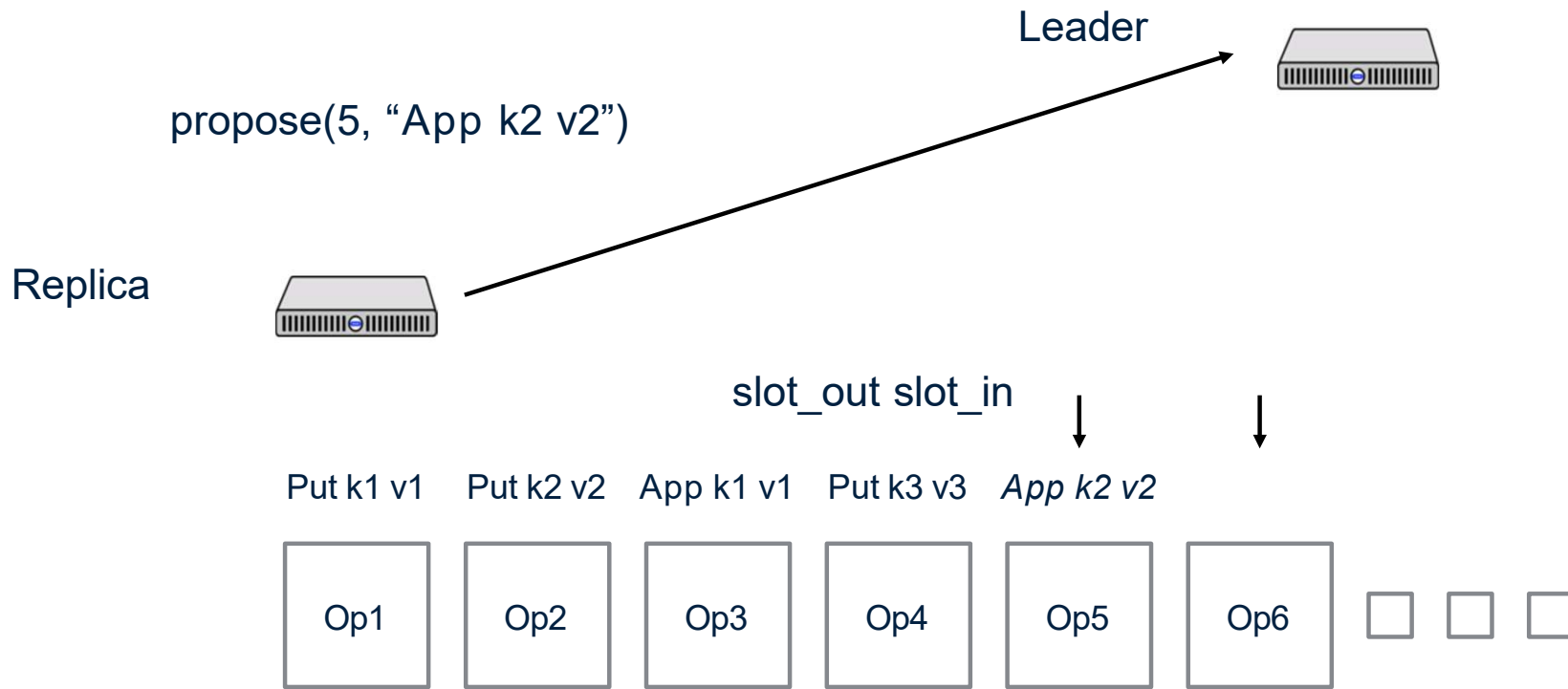
Put k1 v1 Put k2 v2 App k1 v1 *App k2 v2*



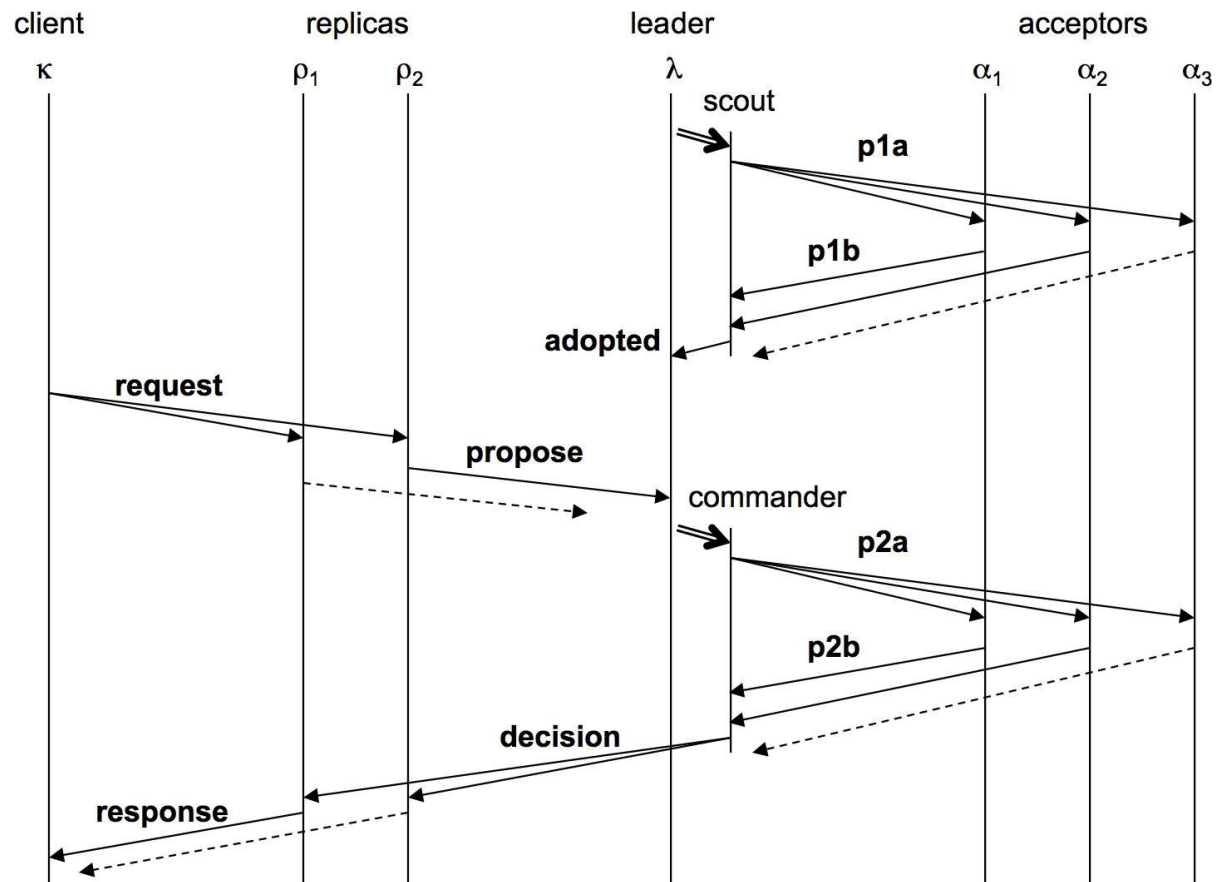
Replicas



Replicas



Paxos Made Moderately Complex Made Simple



Reconfiguration



All replicas *must* agree on who the leaders and acceptors are

How do we do this?

Reconfiguration

All replicas *must* agree on who the leaders and acceptors are

How do we do this?

- Use the log!
- Commit a special reconfiguration command
- New config applies after WINDOW slots



Reconfiguration

What if we need to reconfigure *now* and client requests aren't coming in?



Questions

What should be in stable storage?

Question

- What are the costs to using Paxos? Is it practical enough?



THE UNIVERSITY OF BRITISH COLUMBIA

