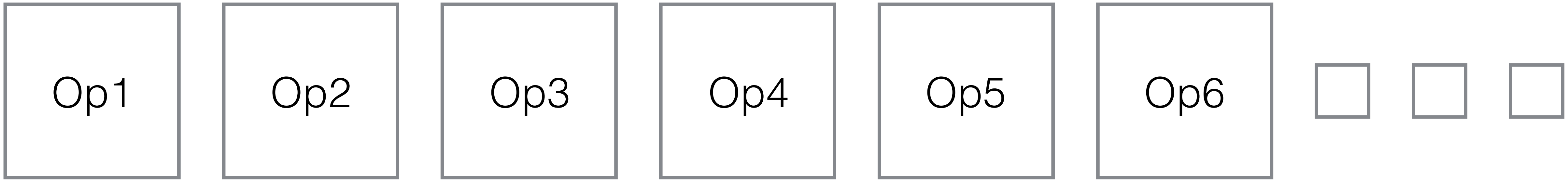
“Paxos Made Moderately Complex” Made Moderately Simple

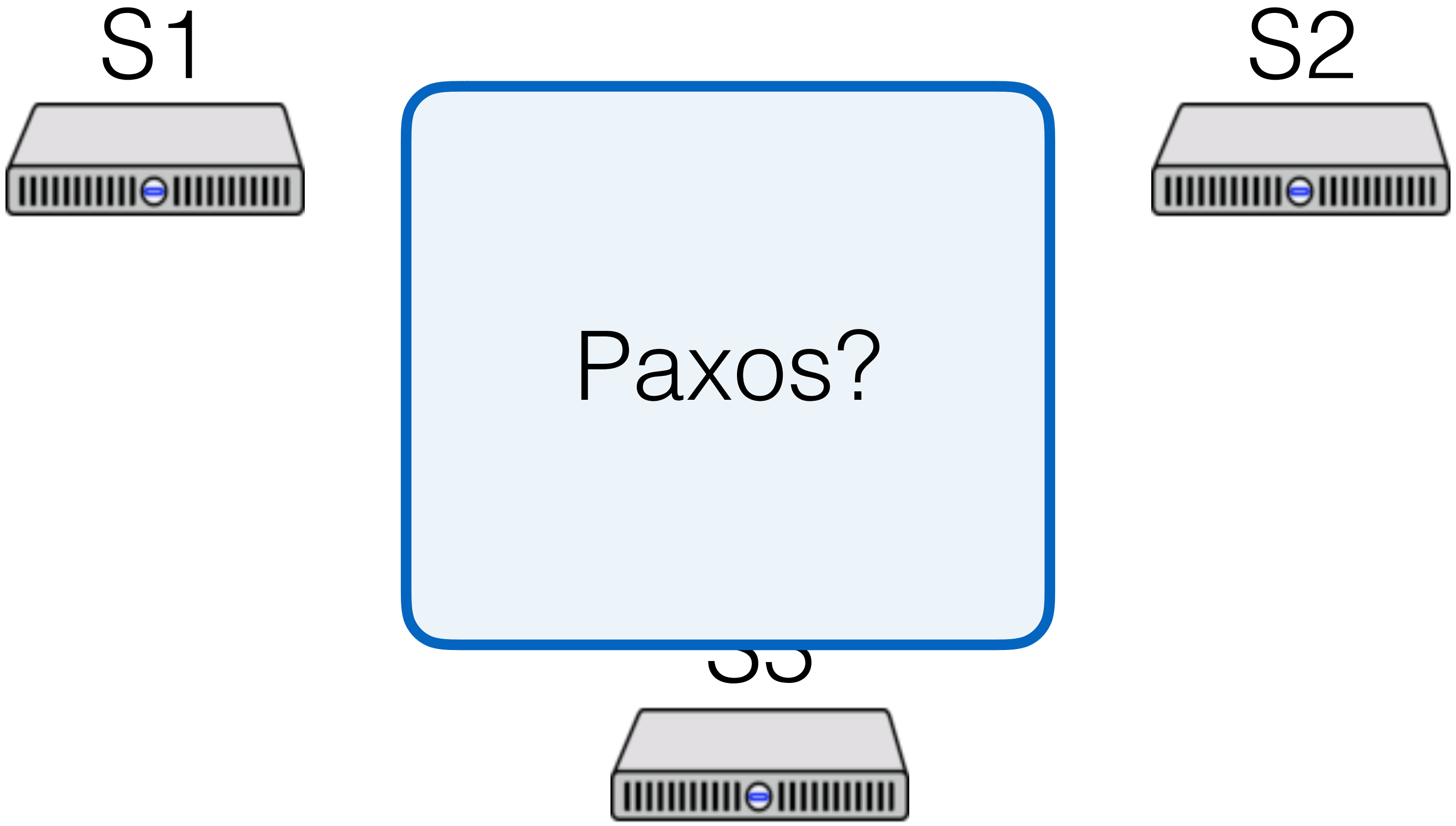
State machine replication

Reminder: want to agree on order of ops

Can think of operations as a log

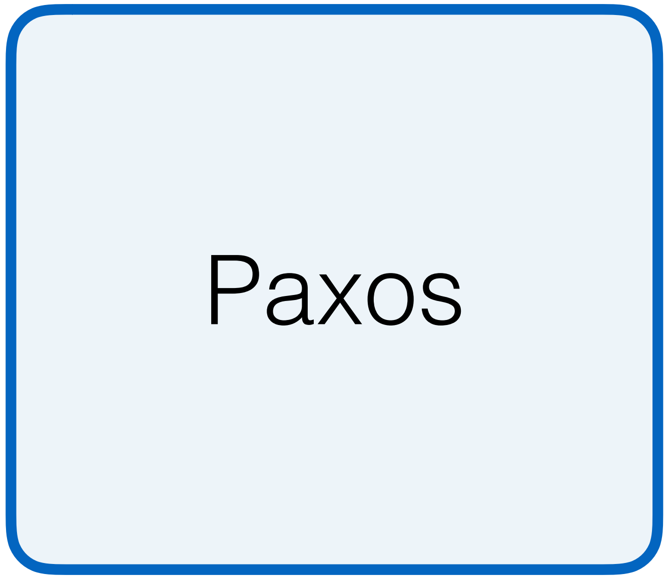


Op1 Op2 Op3 Op4 Op5 Op6



Put k1 v1	Put k2 v2							
Op1	Op2	Op3	Op4	Op5	Op6	□	□	□

Paxos



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

0	0, 4, 8, 12, 16, …
1	1, 5, 9, 13, 17, …
2	2, 6, 10, 14, 18, …
3	3, 7, 11, 15, 19, …

A note about ballots in PMMC

(leader, seqnum) pairs

Isomorphic to the system we discussed earlier

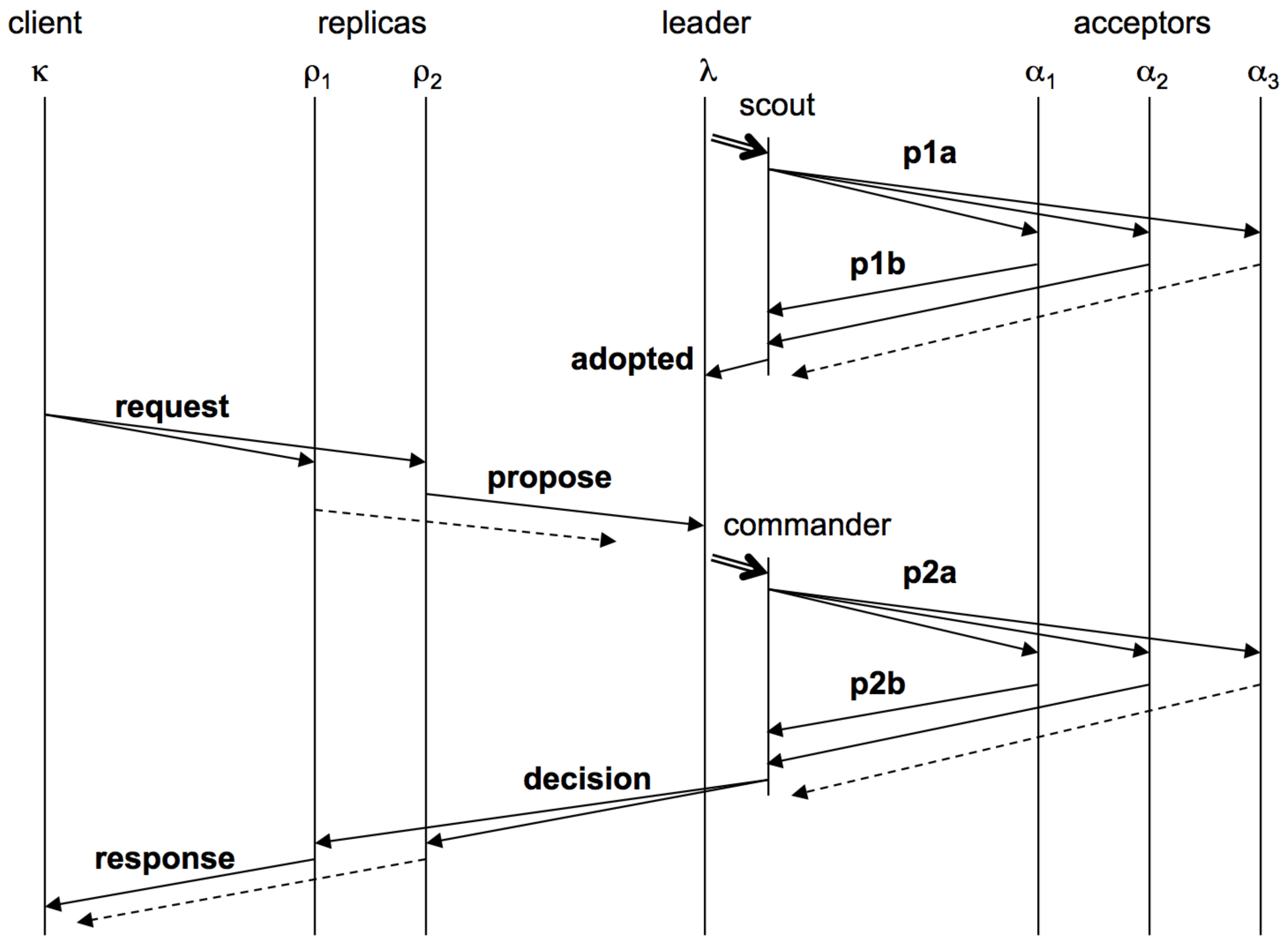
0 0.0, 1.0, 2.0, 3.0, 4.0, …

1 0.1, 1.1, 2.1, 3.1, 4.1, …

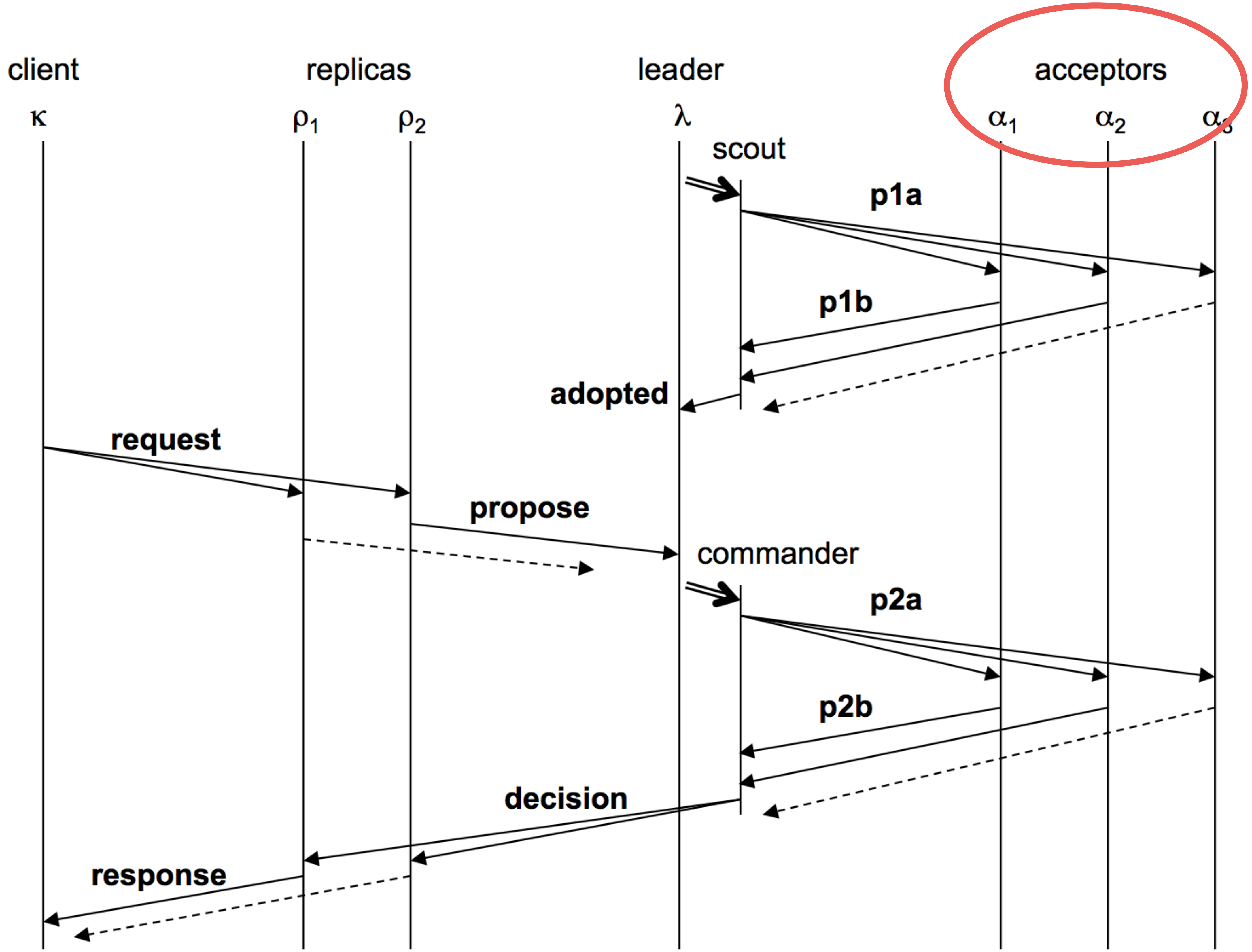
2 0.2, 1.2, 2.2, 3.2, 4.2, …

3 0.3, 1.3, 2.3, 3.3, 4.3, …

Paxos Made Moderately Complex Made Simple



Paxos Made Moderately Complex Made Simple

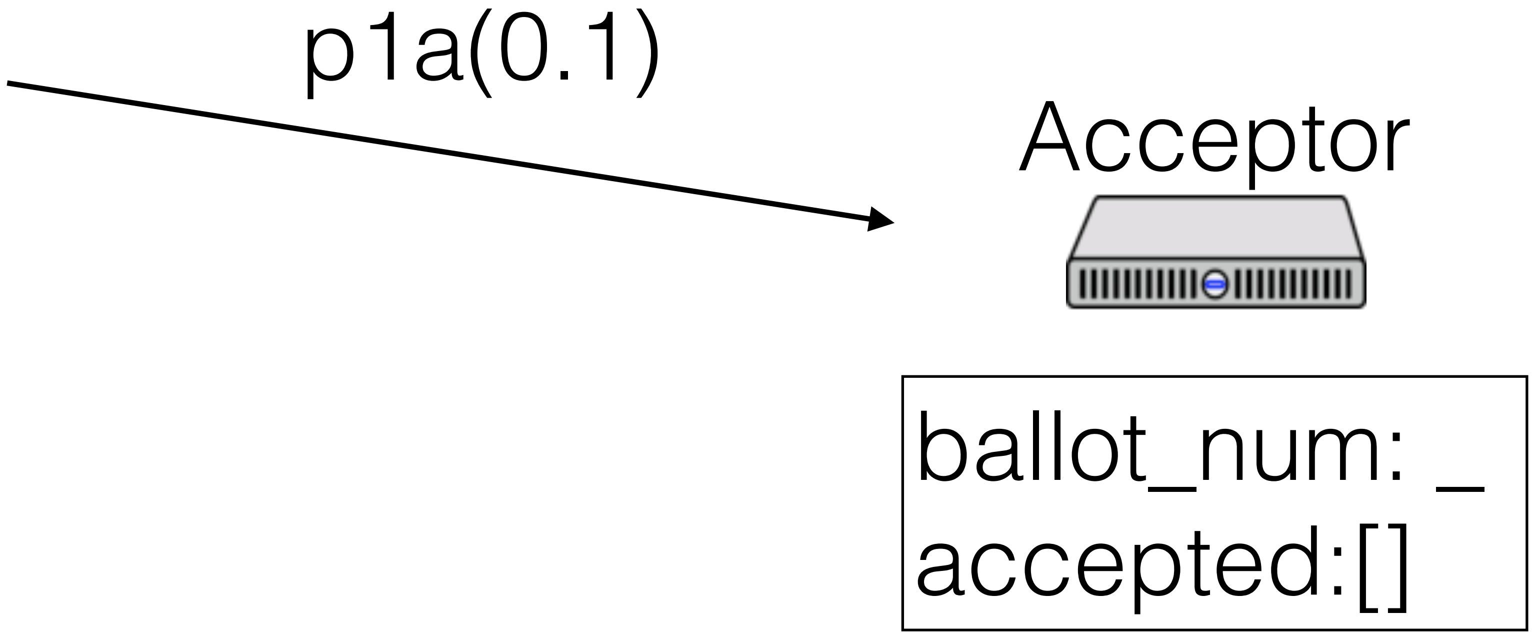


Acceptors

Acceptor

ballot_num: 0
accepted:[]

Acceptors

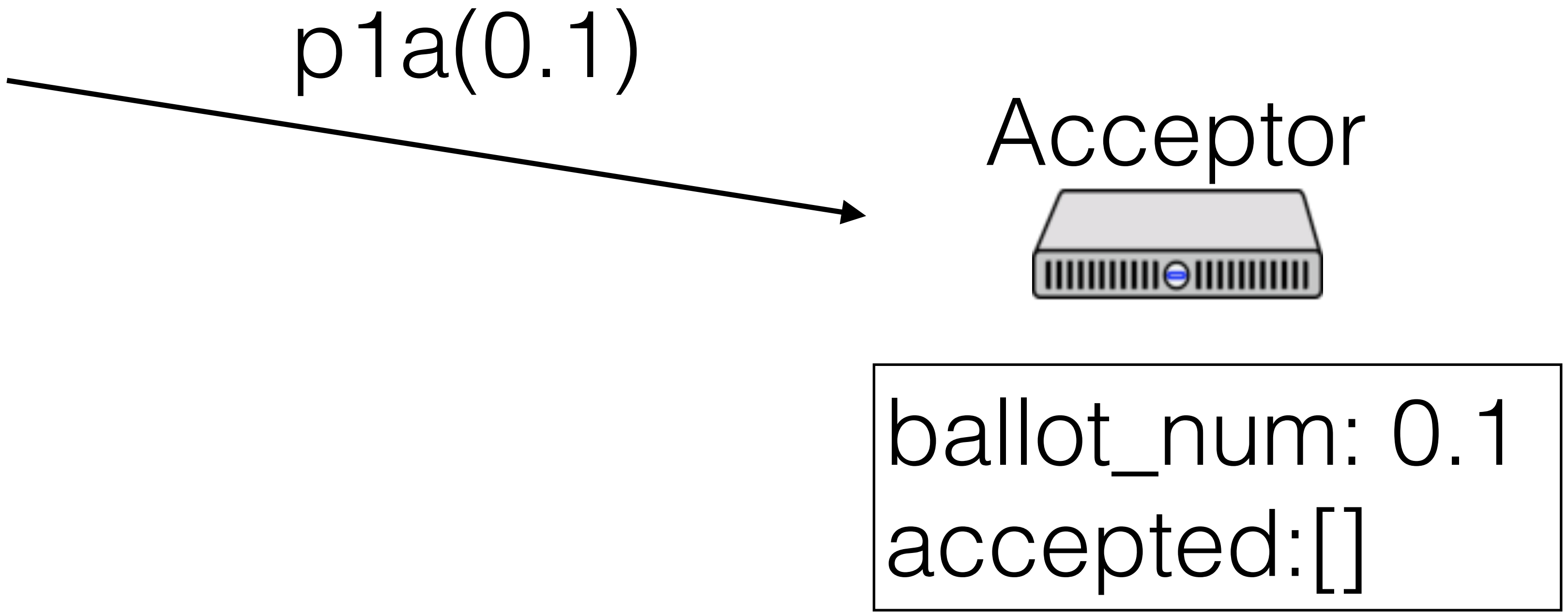


p1a(0.1)

Acceptor

ballot_num: _
accepted:[]

Acceptors

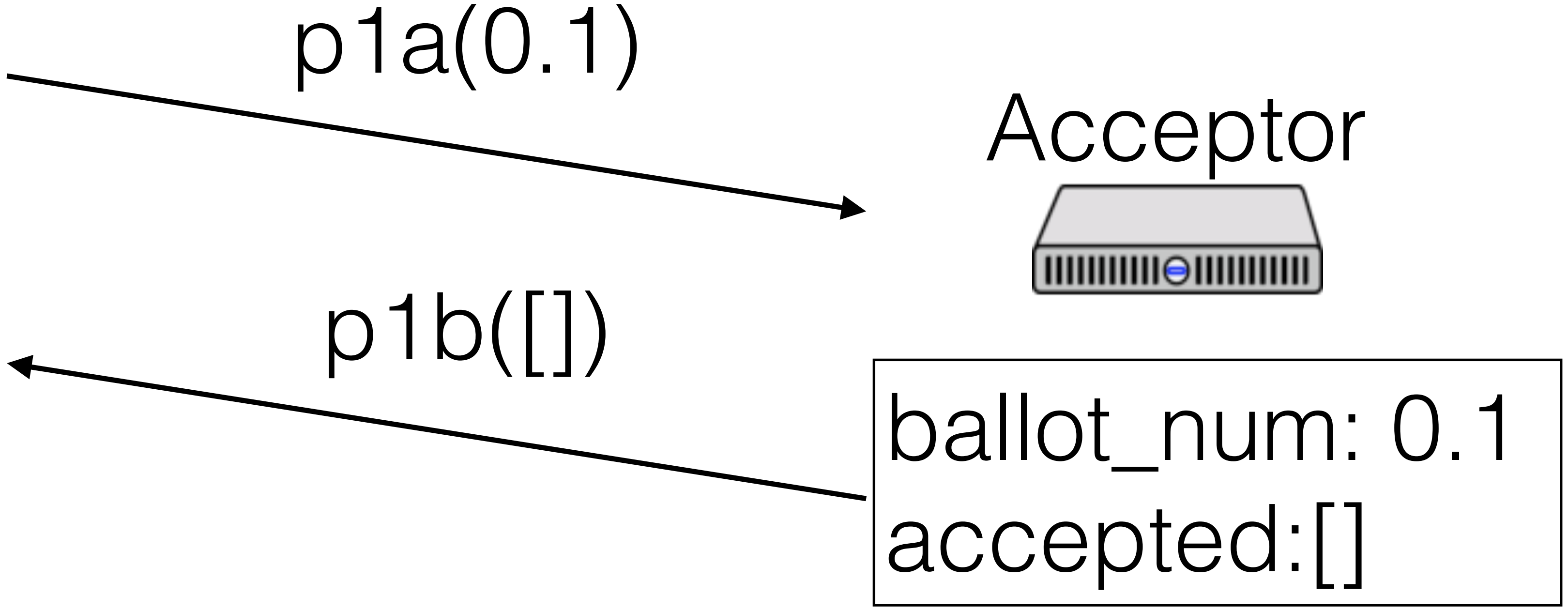


p1a(0.1)

Acceptor

ballot_num: 0.1
accepted:[]

Acceptors



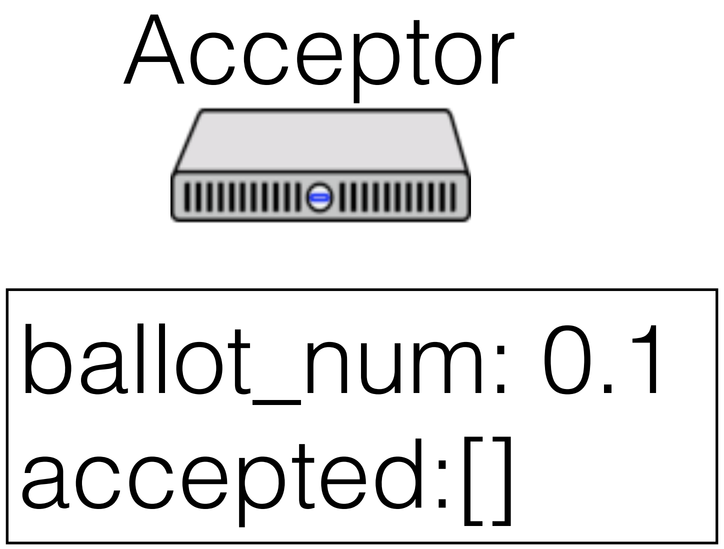
p1a(0.1)

Acceptor

p1b([])

ballot_num: 0.1
accepted:[]

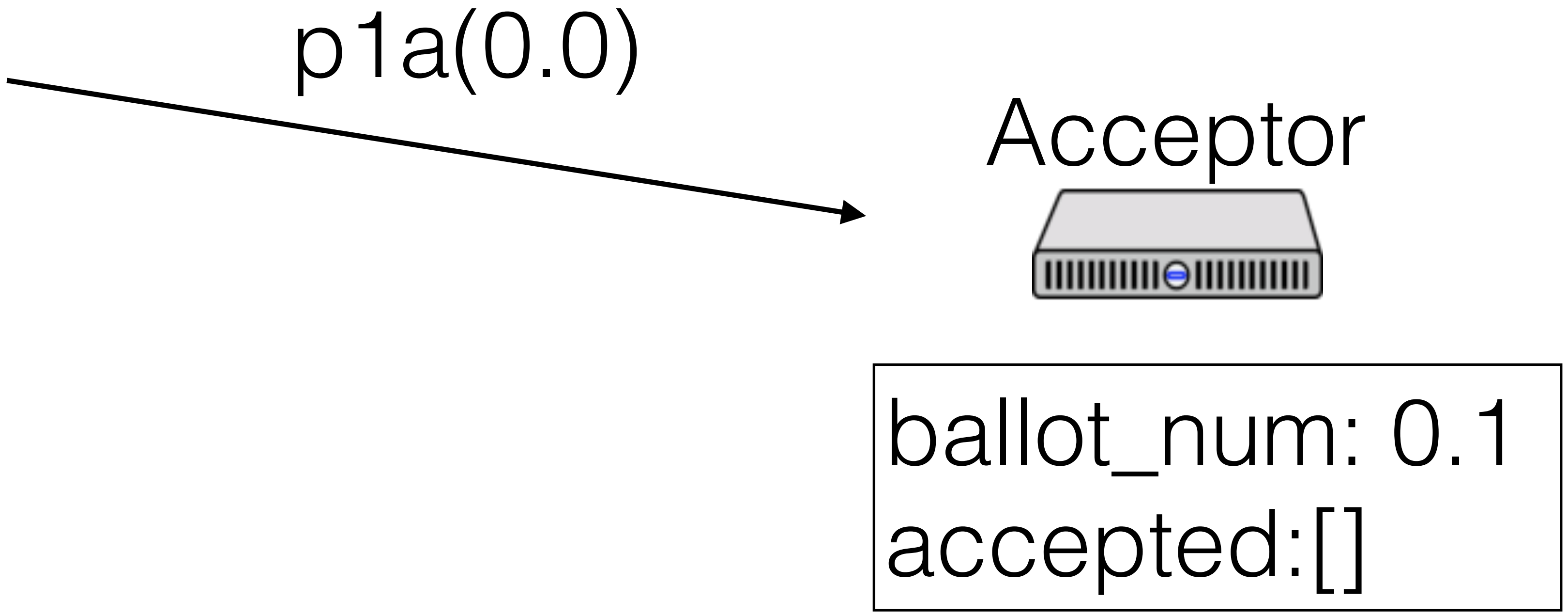
Acceptors



Acceptor

ballot_num: 0.1
accepted:[]

Acceptors

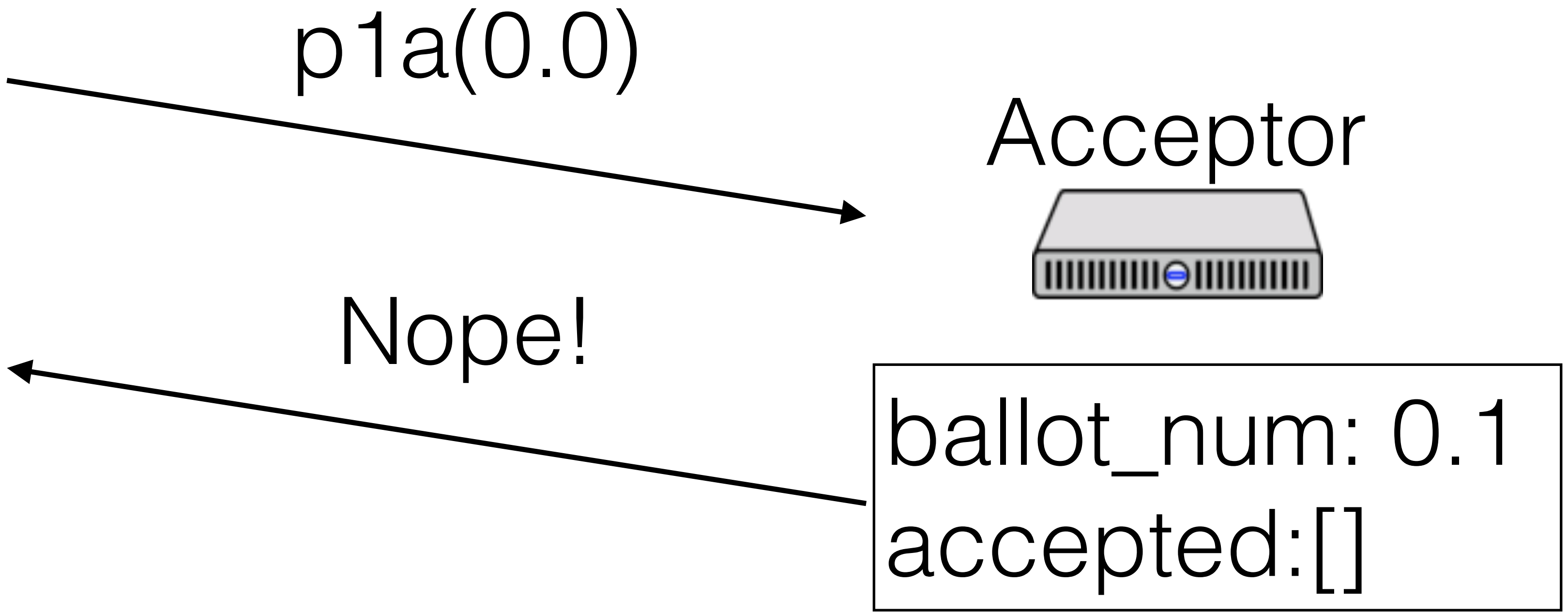


p1a(0.0)

Acceptor

ballot_num: 0.1
accepted:[]

Acceptors



p1a(0.0)

Acceptor

Nope!

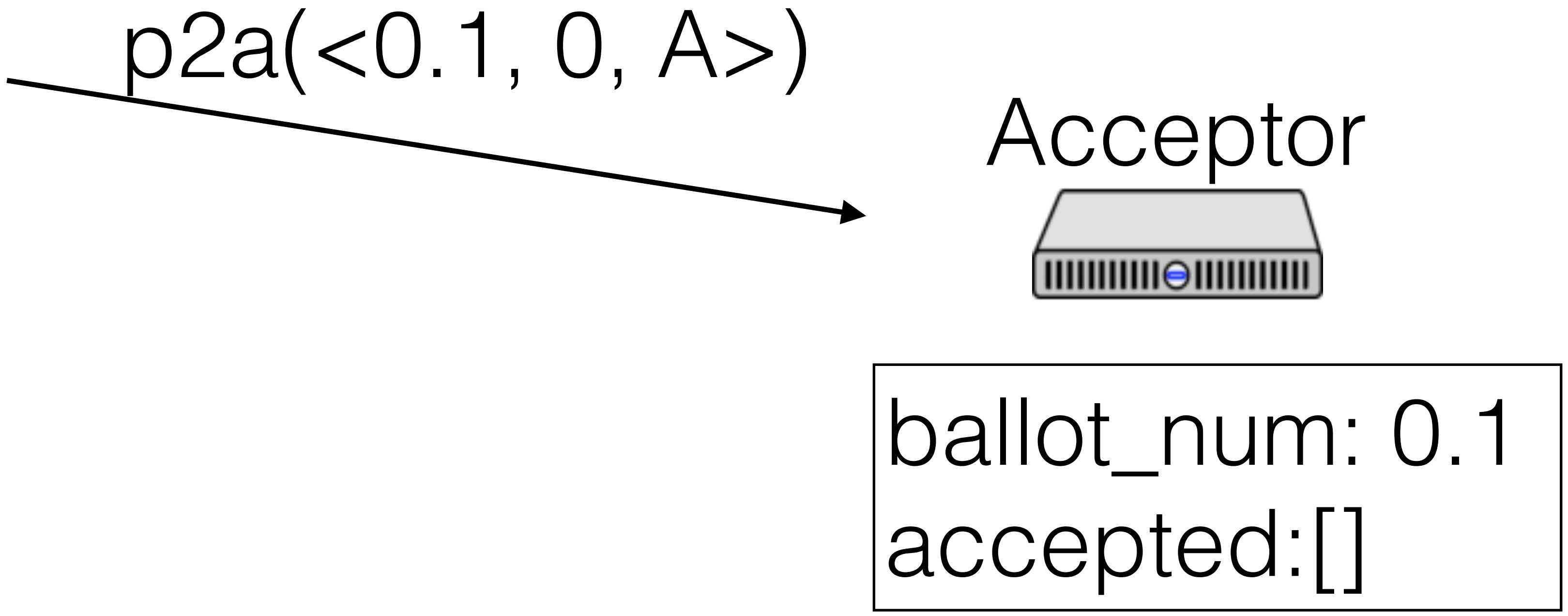
ballot_num: 0.1
accepted:[]

Acceptors

Acceptor

ballot_num: 0.1
accepted:[]

Acceptors

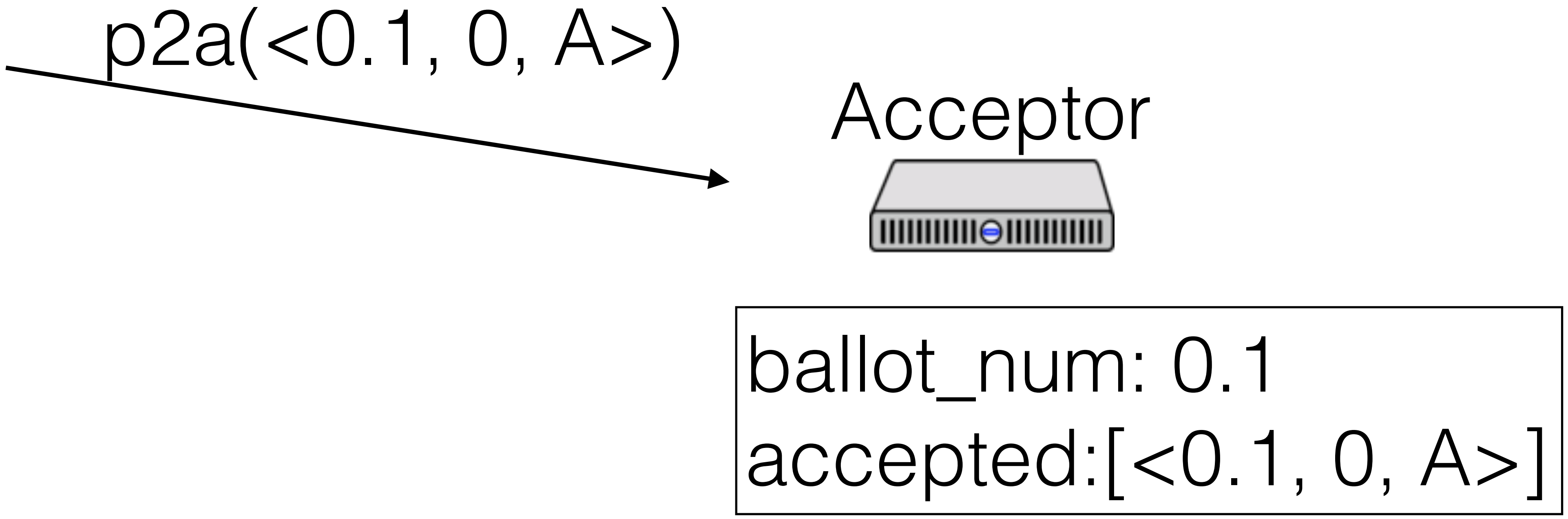


p2a(<0.1, 0, A>)

Acceptor

ballot_num: 0.1
accepted:[]

Acceptors

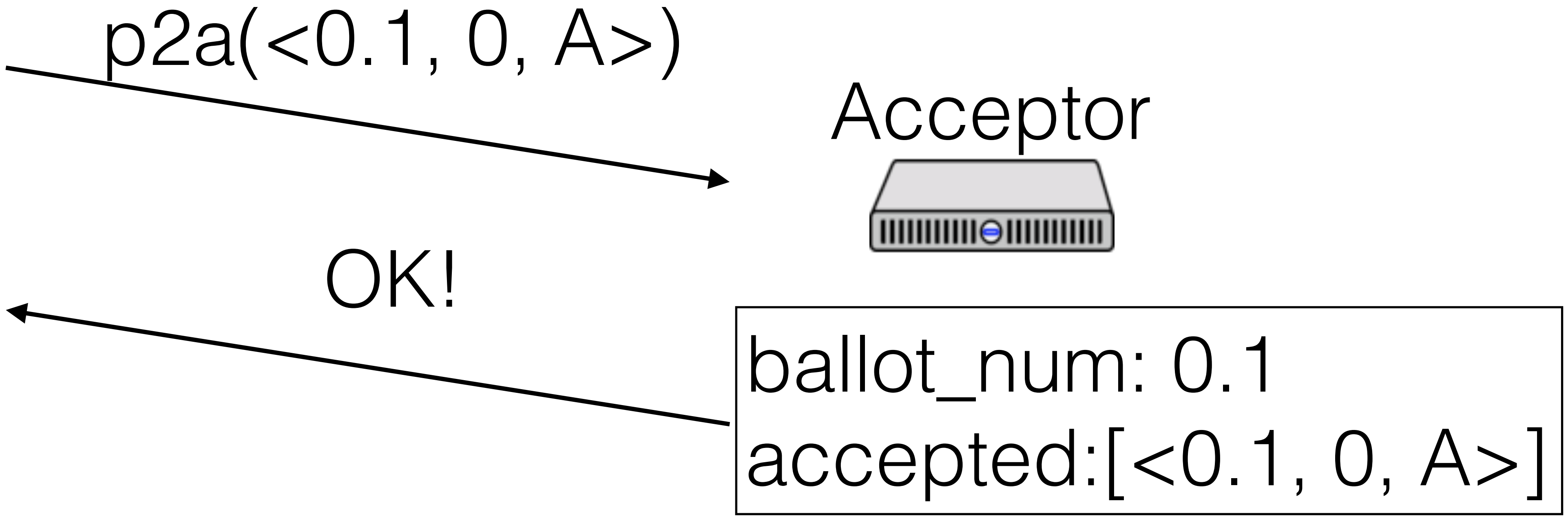


p2a(<0.1, 0, A>)

Acceptor

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

Acceptors



p2a(<0.1, 0, A>)

Acceptor

OK!

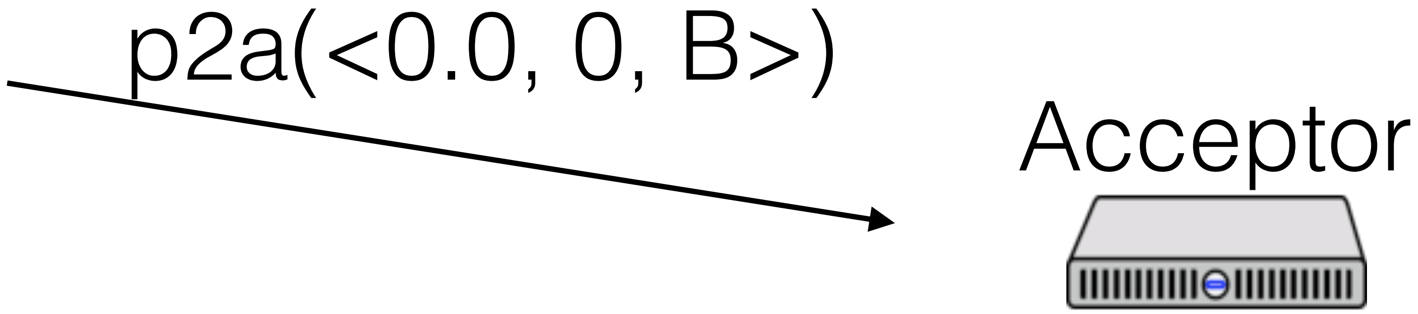
ballot_num: 0.1
accepted:[<0.1, 0, A>]

Acceptors

Acceptor

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

Acceptors

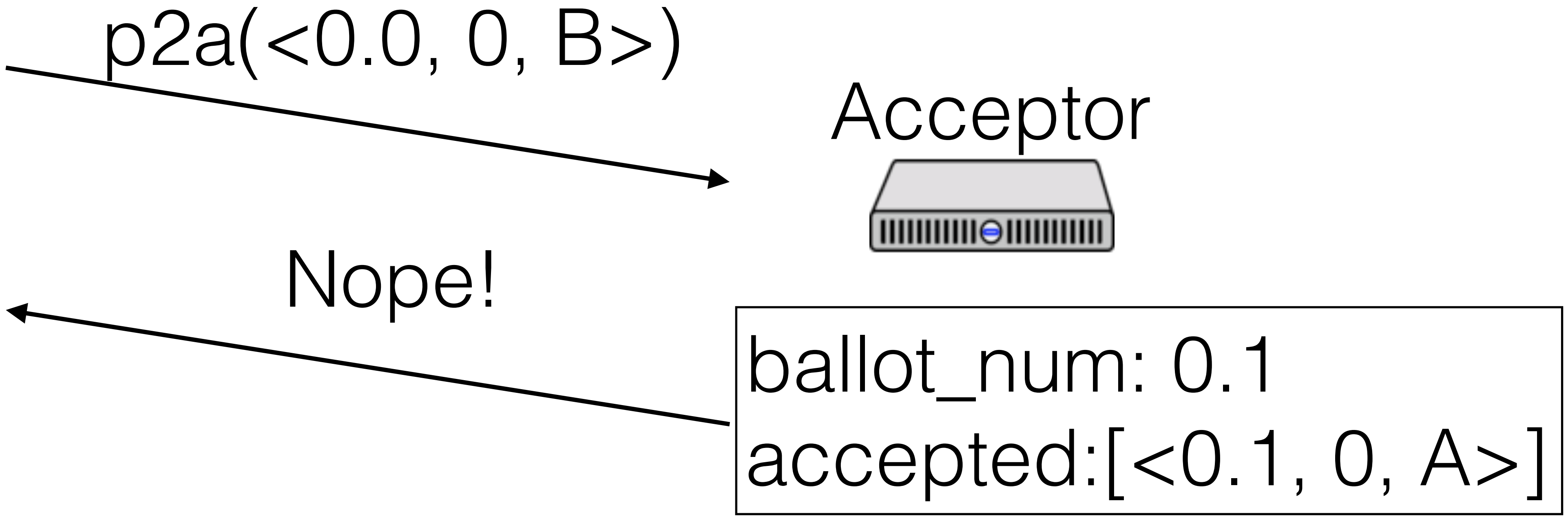


p2a(<0.0, 0, B>)

Acceptor

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

Acceptors



p2a(<0.0, 0, B>)

Acceptor

Nope!

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

Acceptors

Acceptor

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

Acceptors

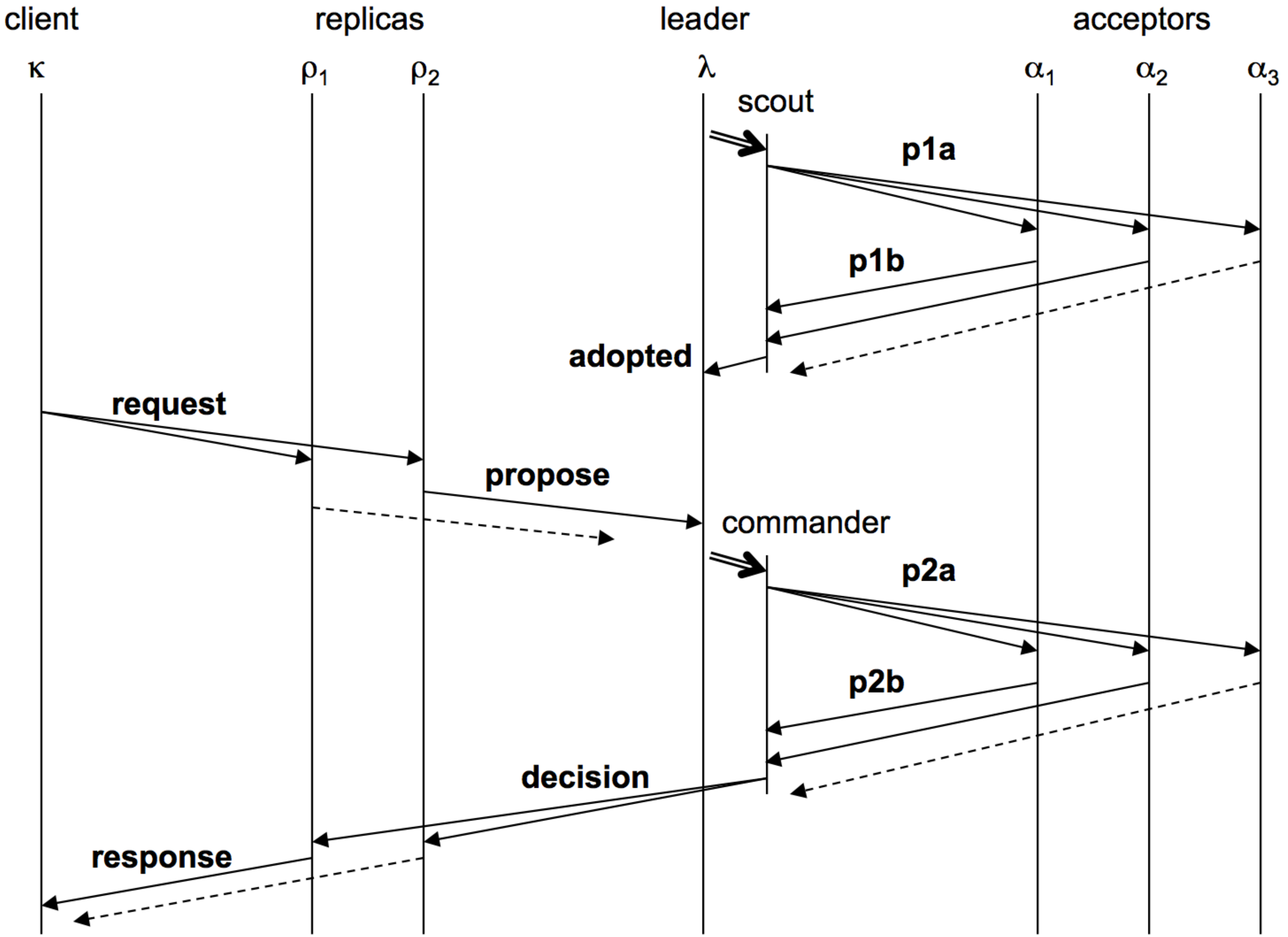
- Ballot numbers increase

- Only accept values from current ballot

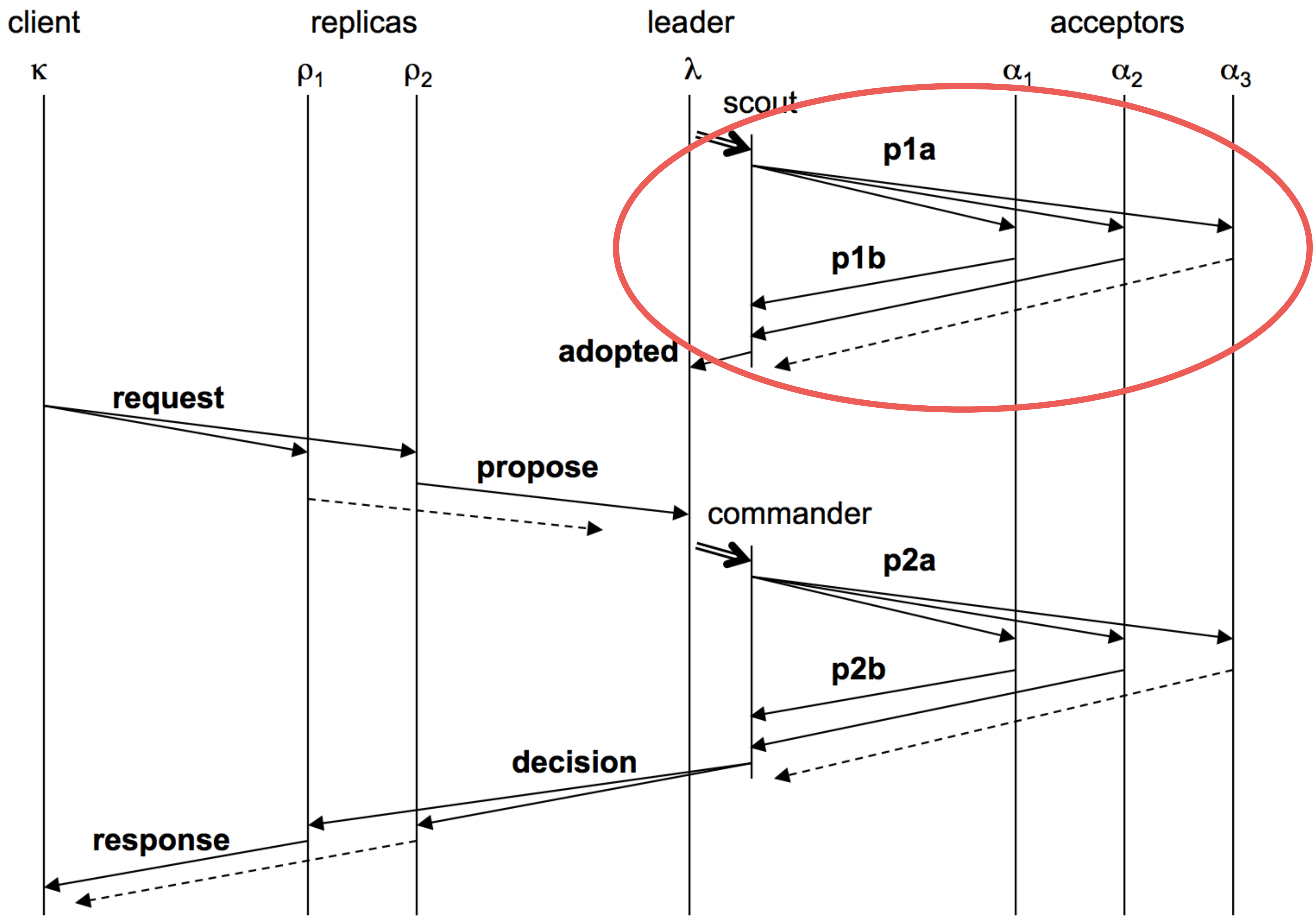
- Never remove ballots

- 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

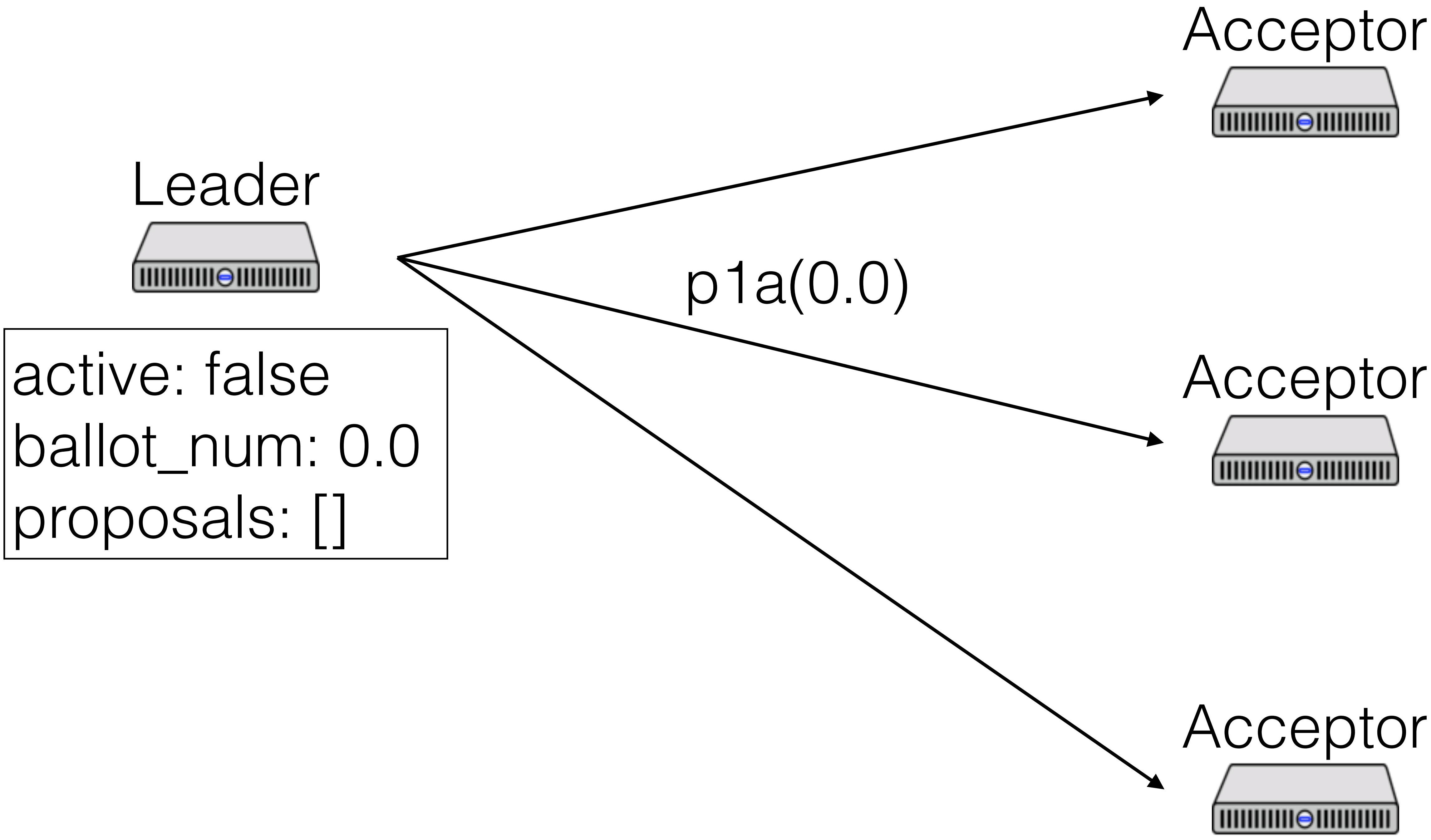


Leader: Getting Elected

Leader

active: false
ballot_num: 0.0
proposals: []

Leader: Getting Elected



Leader

active: false
ballot_num: 0.0
proposals: []

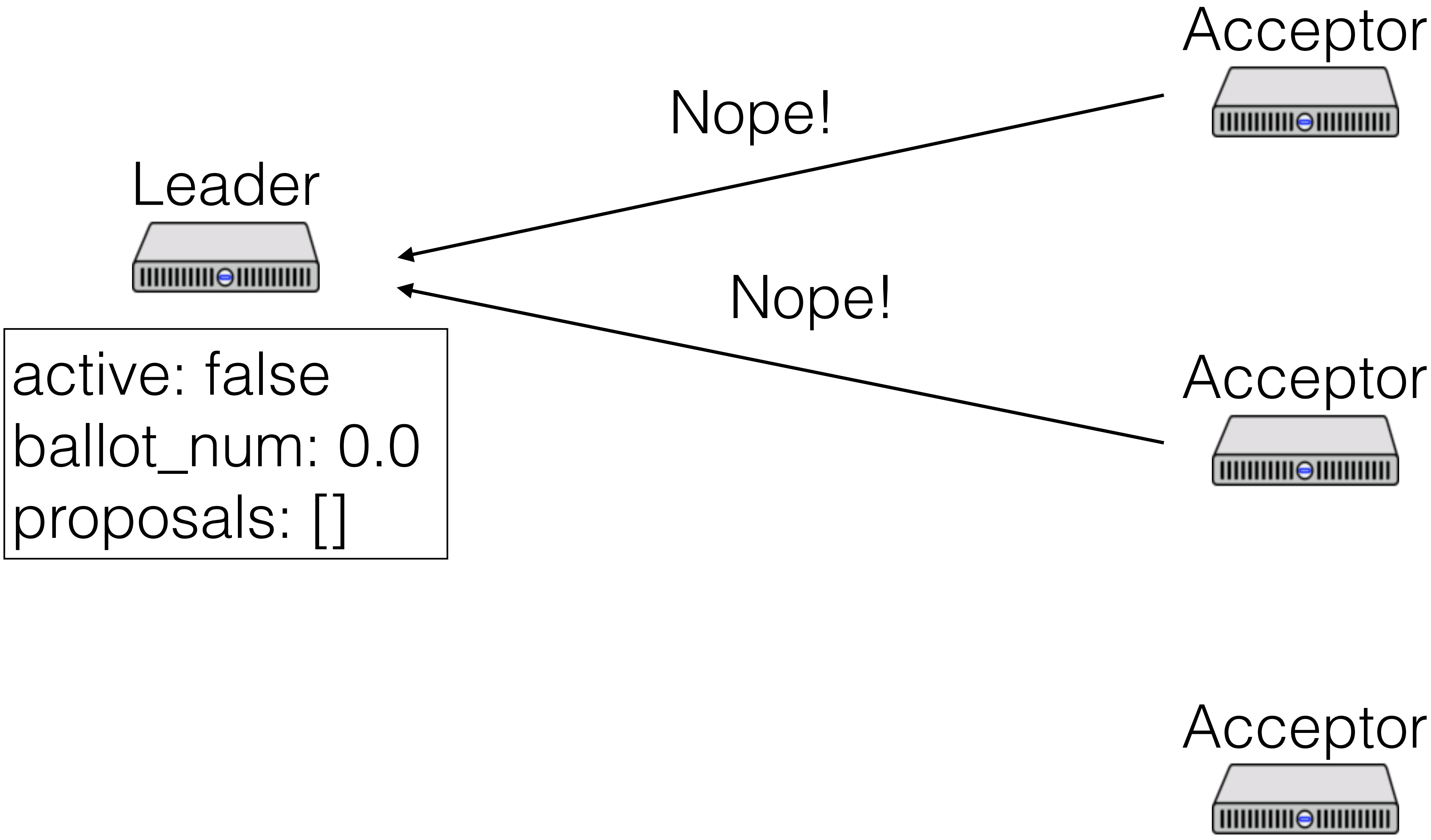
p1a(0.0)

Acceptor

Acceptor

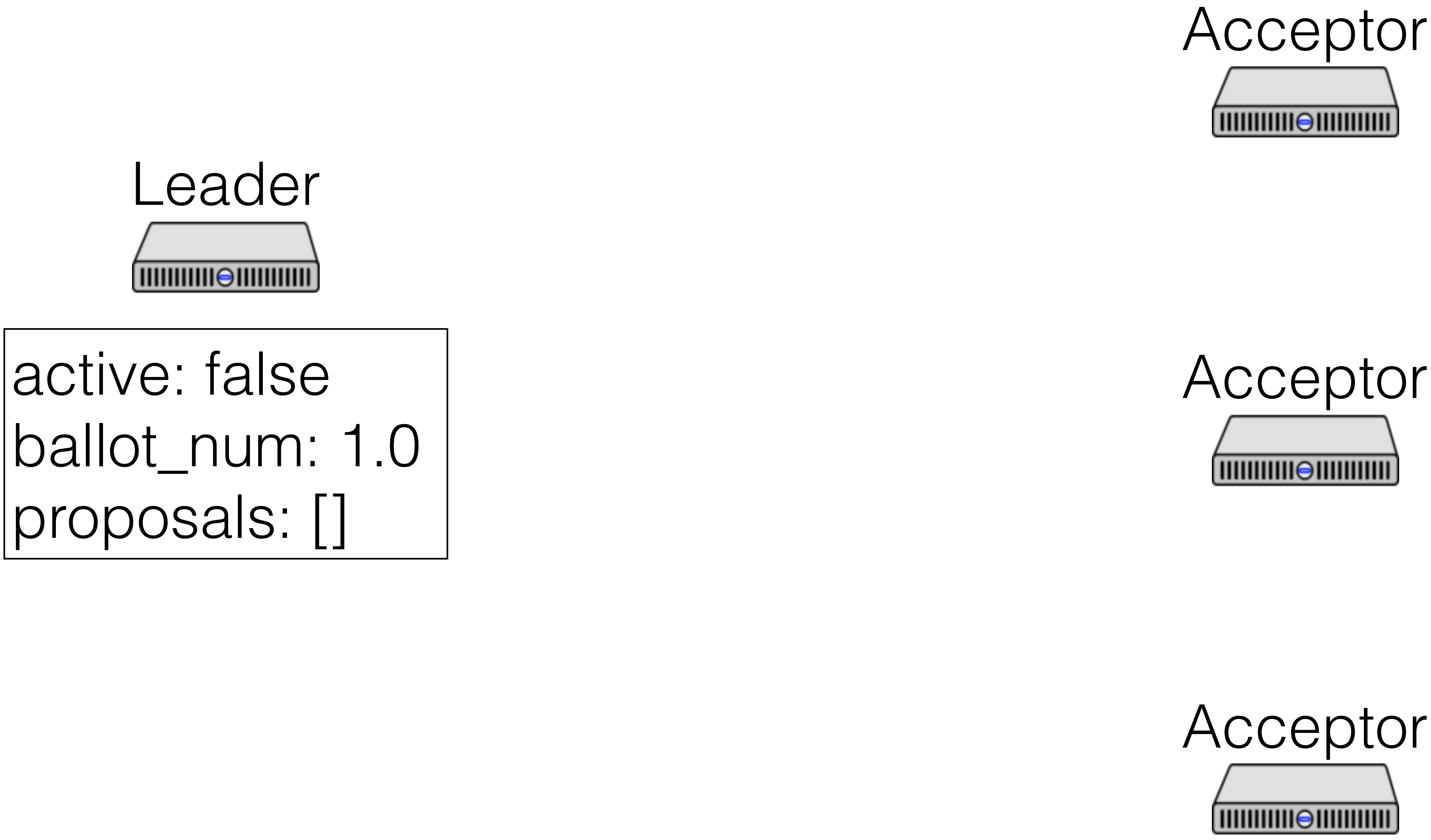
Acceptor

Leader: Getting Elected



active: false
ballot_num: 0.0
proposals: []

Leader: Getting Elected



Acceptor

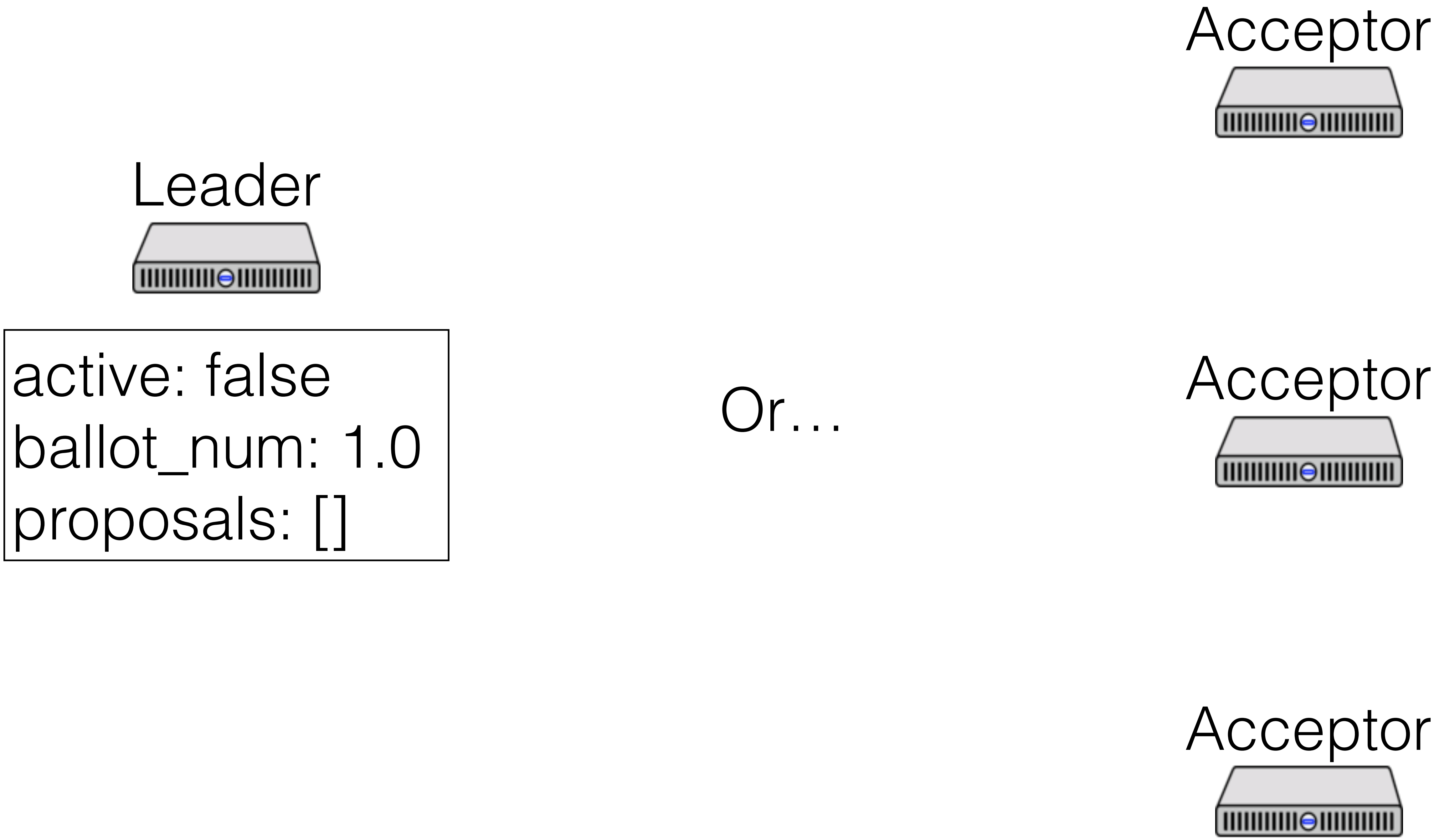
Leader

active: false
ballot_num: 1.0
proposals: []

Acceptor

Acceptor

Leader: Getting Elected



Acceptor

Leader

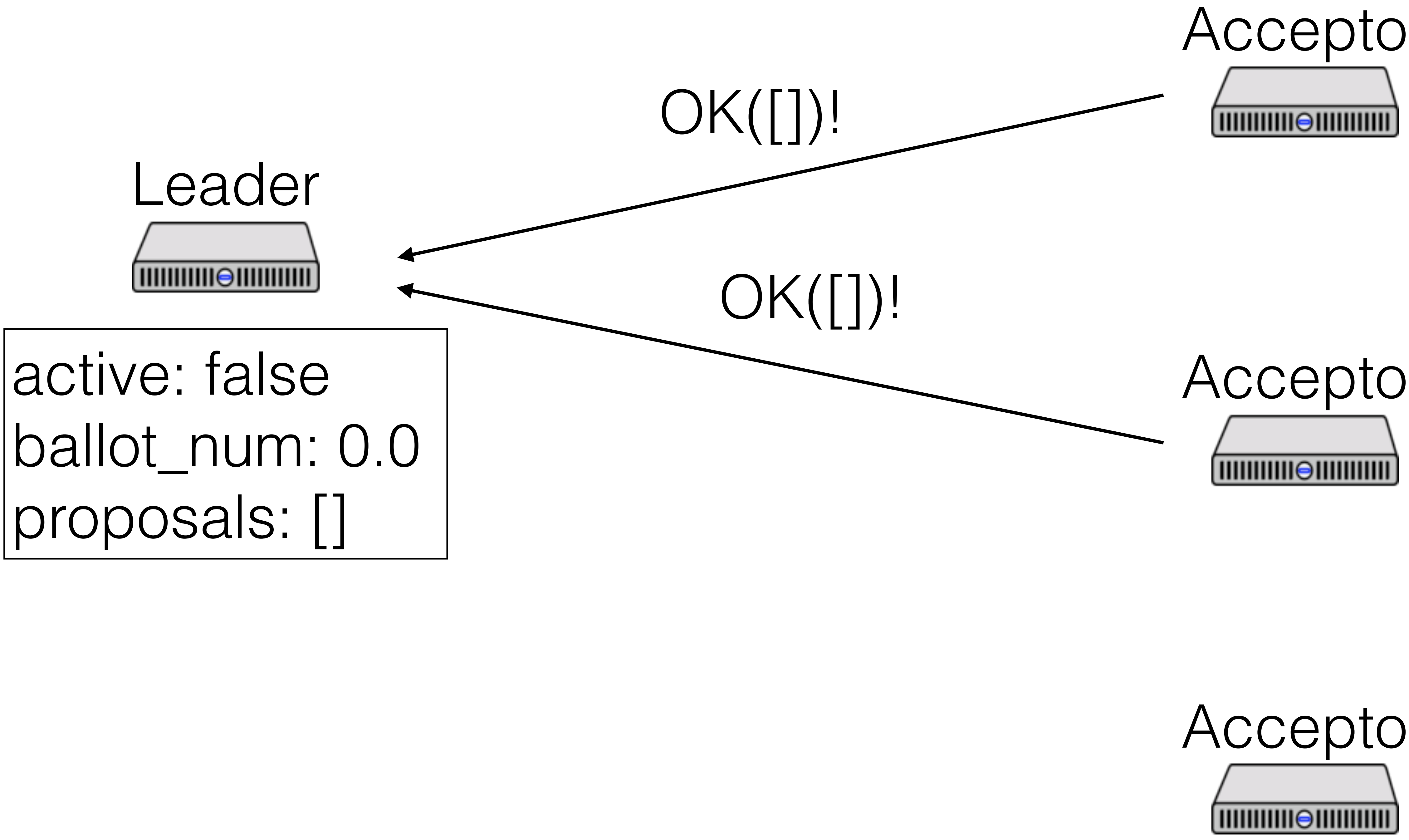
active: false
ballot_num: 1.0
proposals: []

Or…

Acceptor

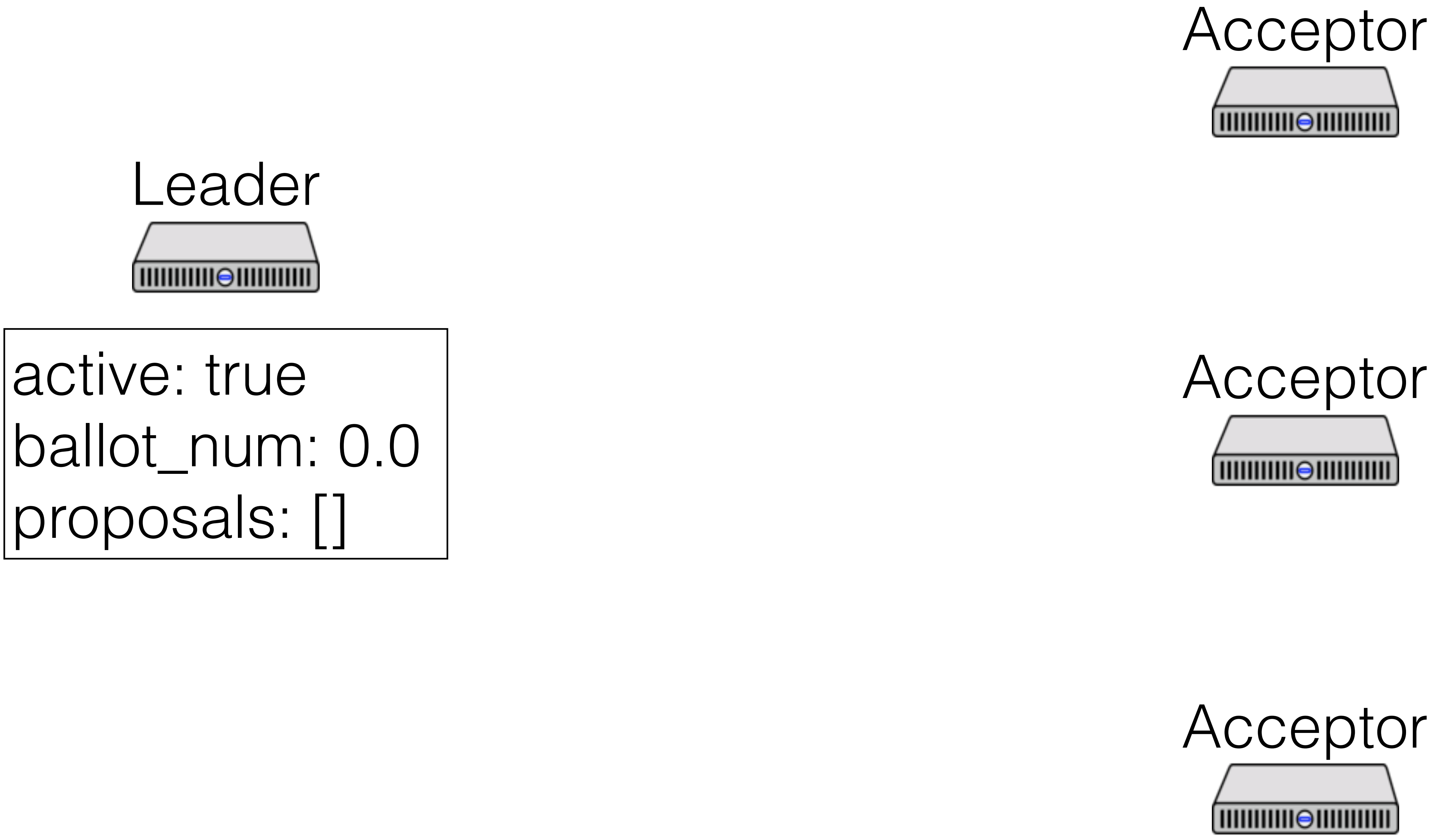
Acceptor

Leader: Getting Elected



active: false
ballot_num: 0.0
proposals: []

Leader: Getting Elected



Acceptor

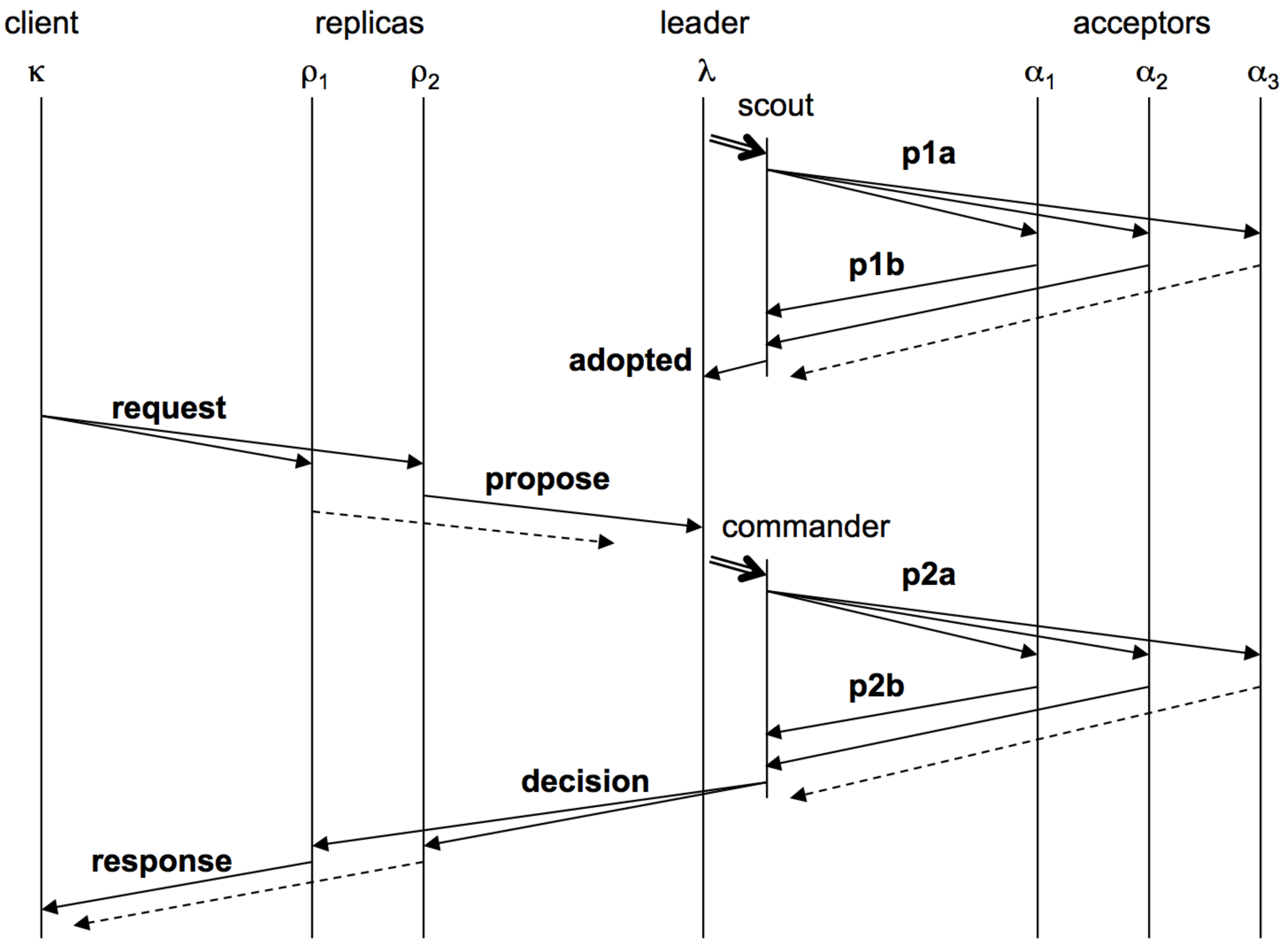
Leader

active: true
ballot_num: 0.0
proposals: []

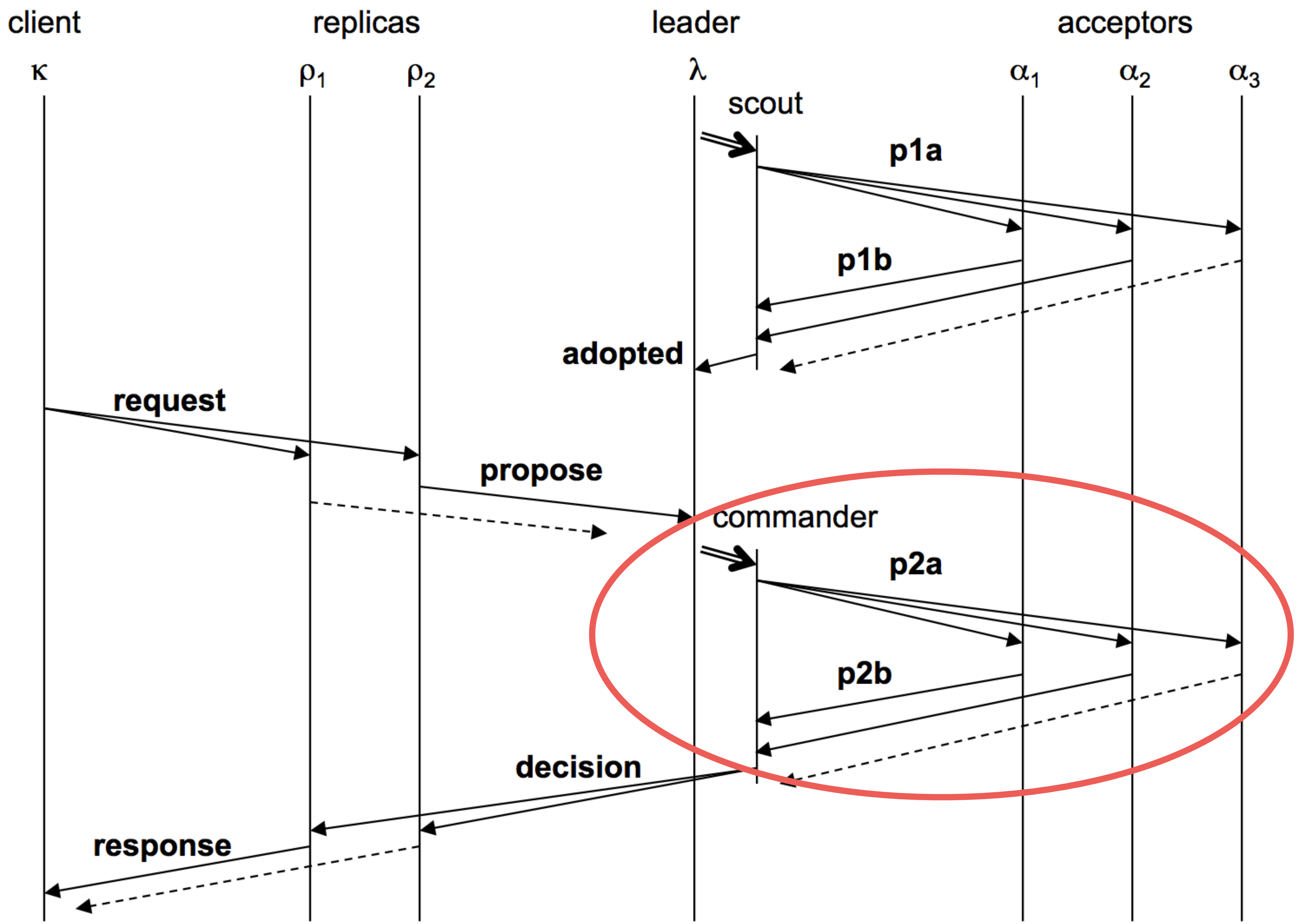
Acceptor

Acceptor

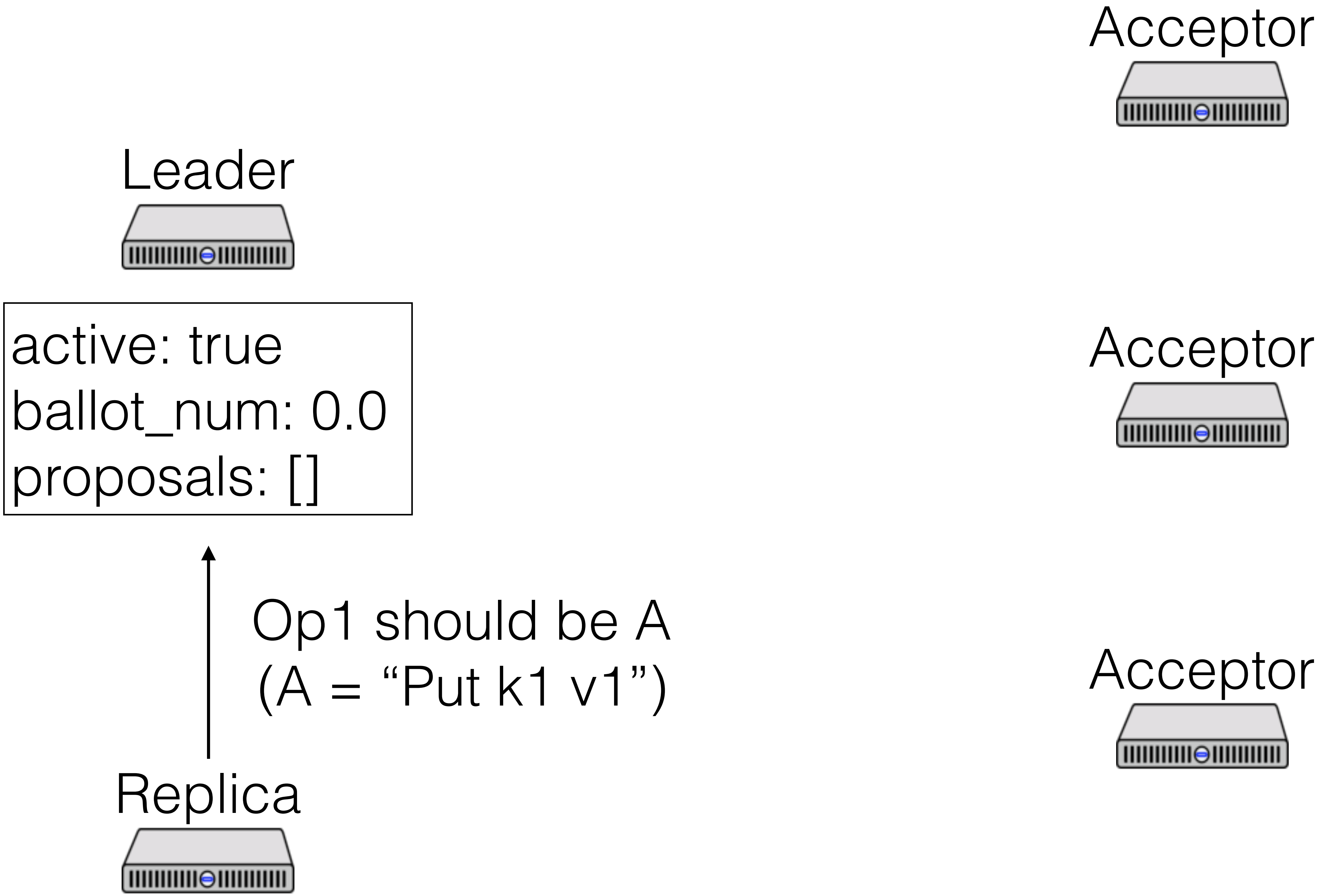
Paxos Made Moderately Complex Made Simple



Paxos Made Moderately Complex Made Simple



Leader: Handling proposals



Acceptor

Leader

active: true
ballot_num: 0.0
proposals: []

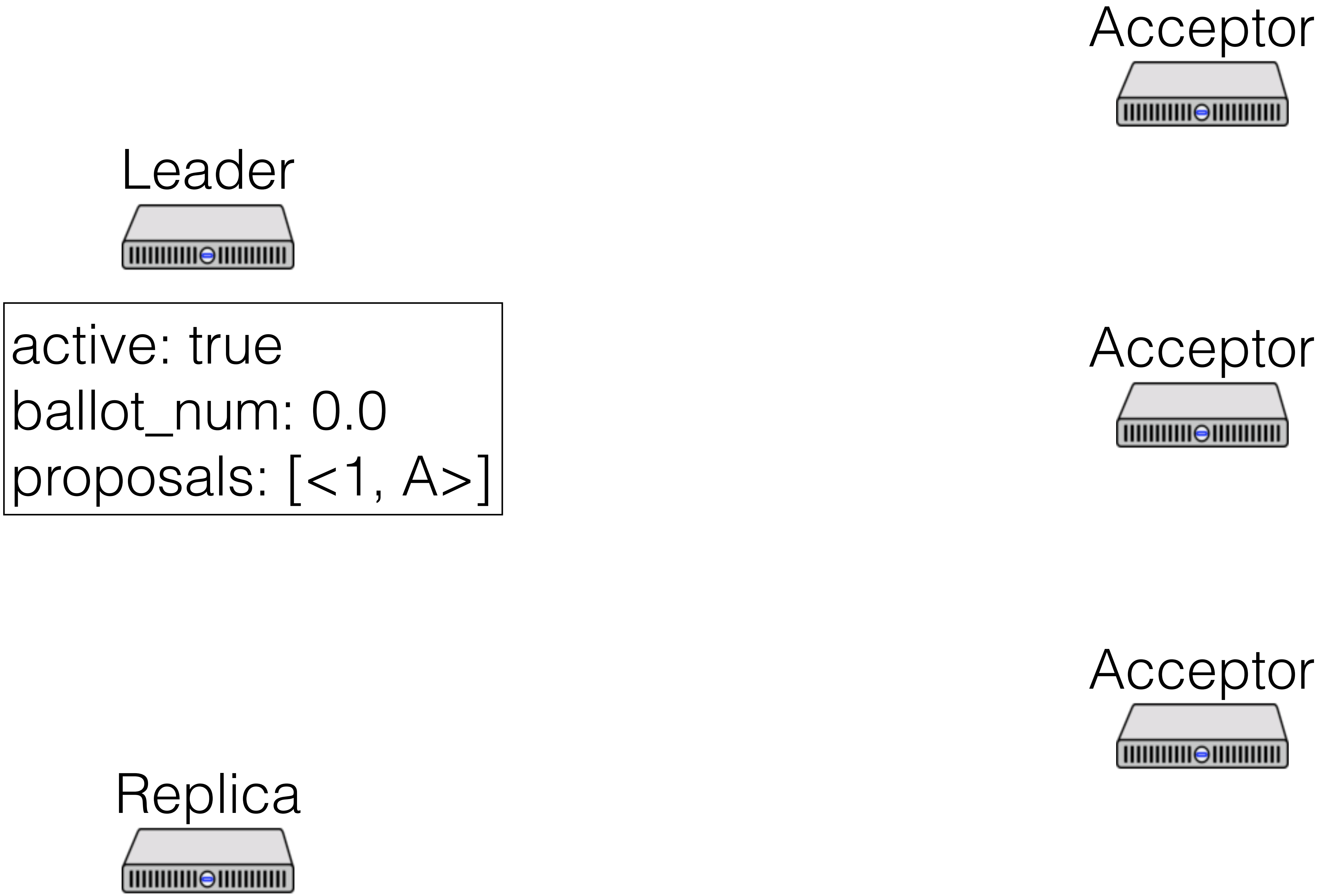
Acceptor

Op1 should be A
(A = “Put k1 v1”)

Acceptor

Replica

Leader: Handling proposals



Acceptor

Leader

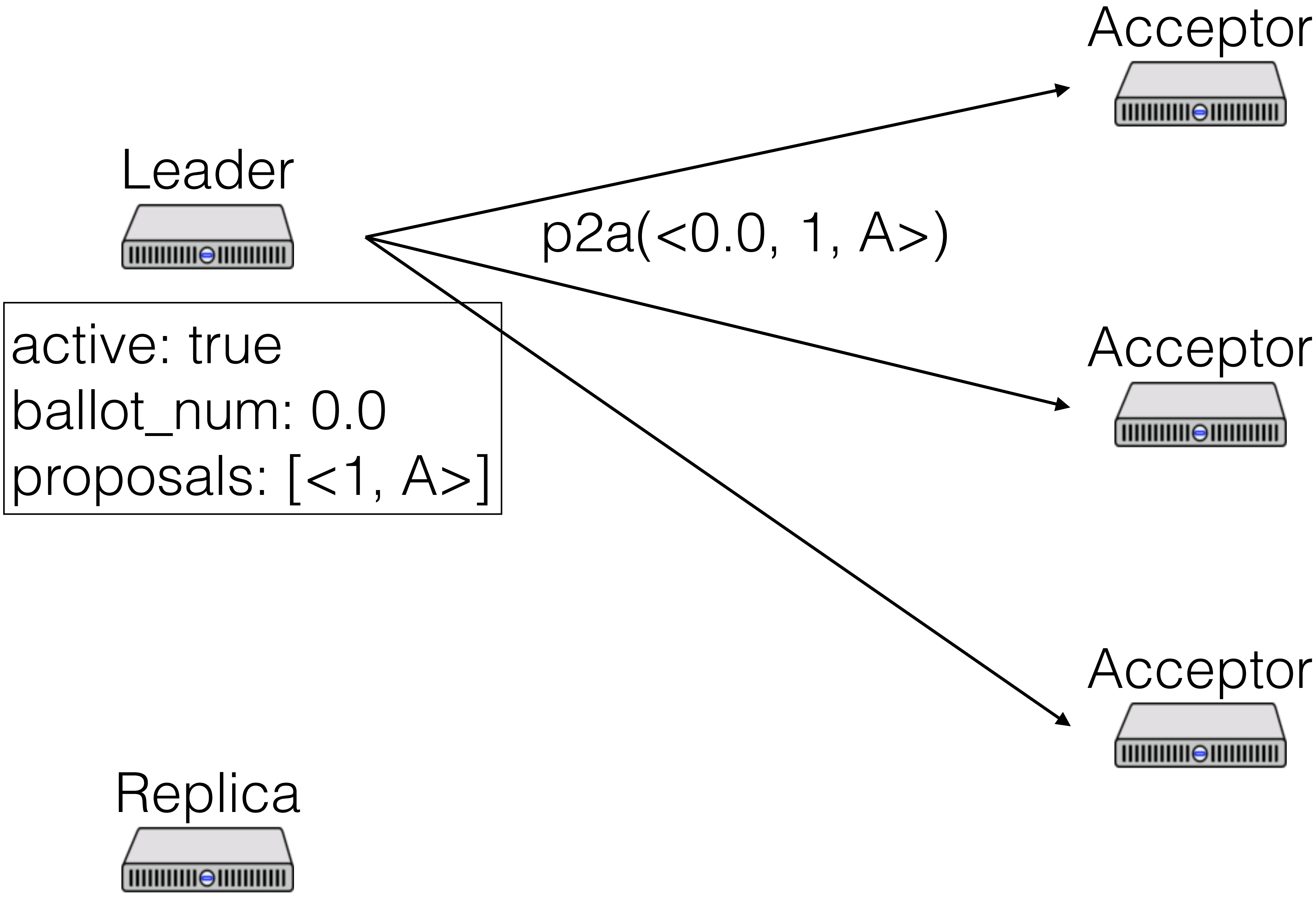
active: true
ballot_num: 0.0
proposals: [<1, A>]

Acceptor

Acceptor

Replica

Leader: Handling proposals



Leader

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

p2a(<0.0, 1, A>)

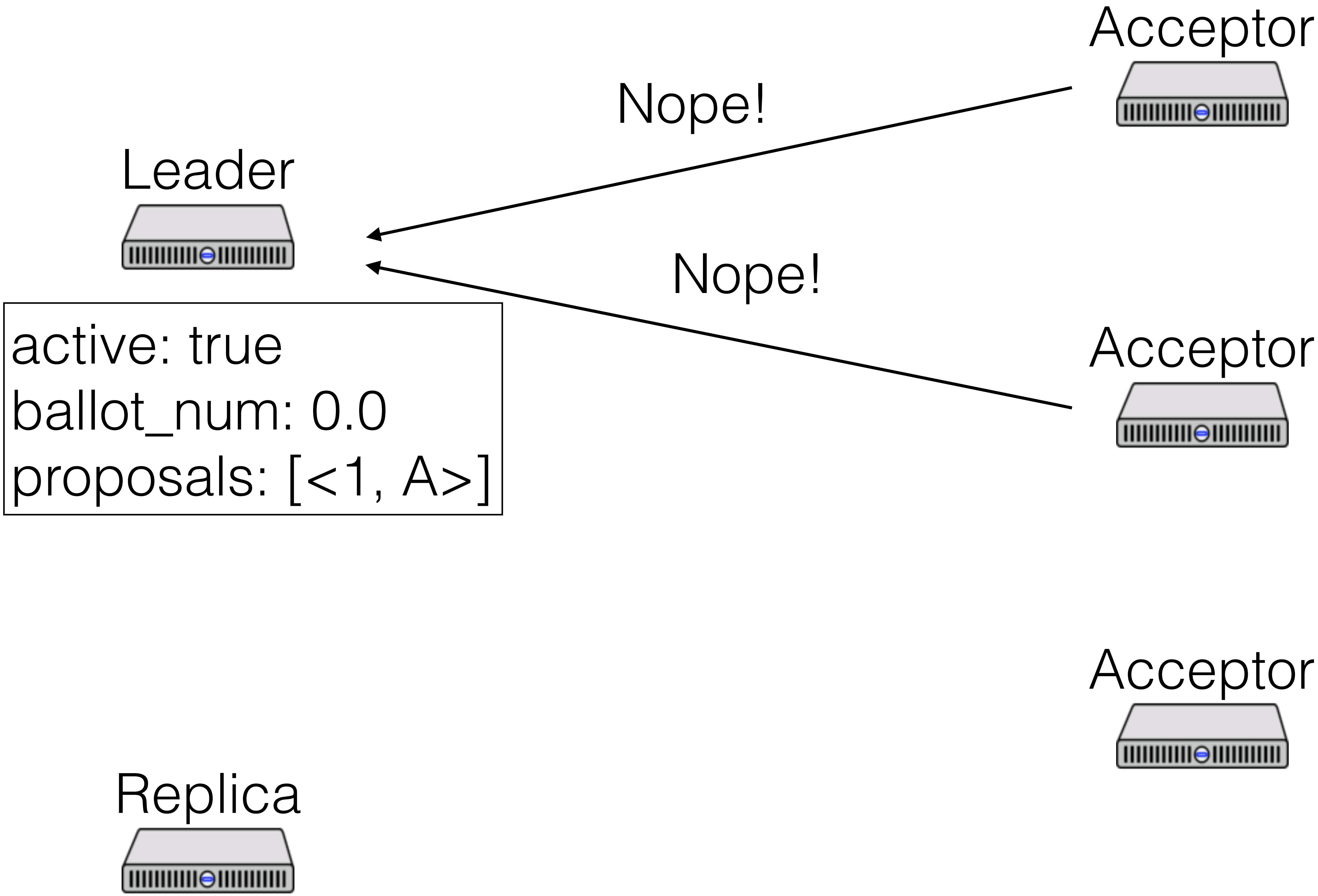
Acceptor

Acceptor

Acceptor

Replica

Leader: Handling proposals



Acceptor

Nope!

Leader

Nope!

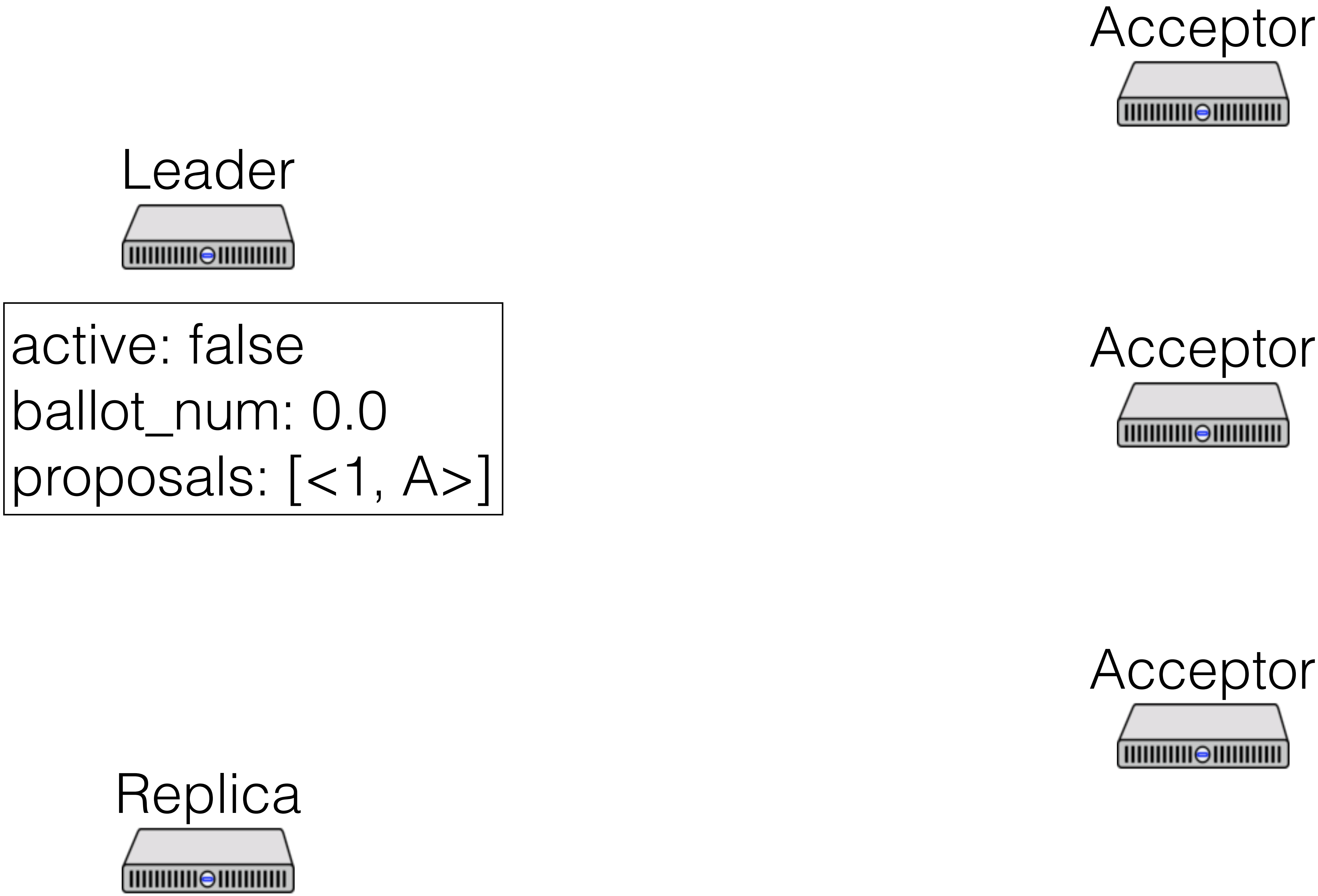
Acceptor

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

Acceptor

Replica

Leader: Handling proposals



Leader

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

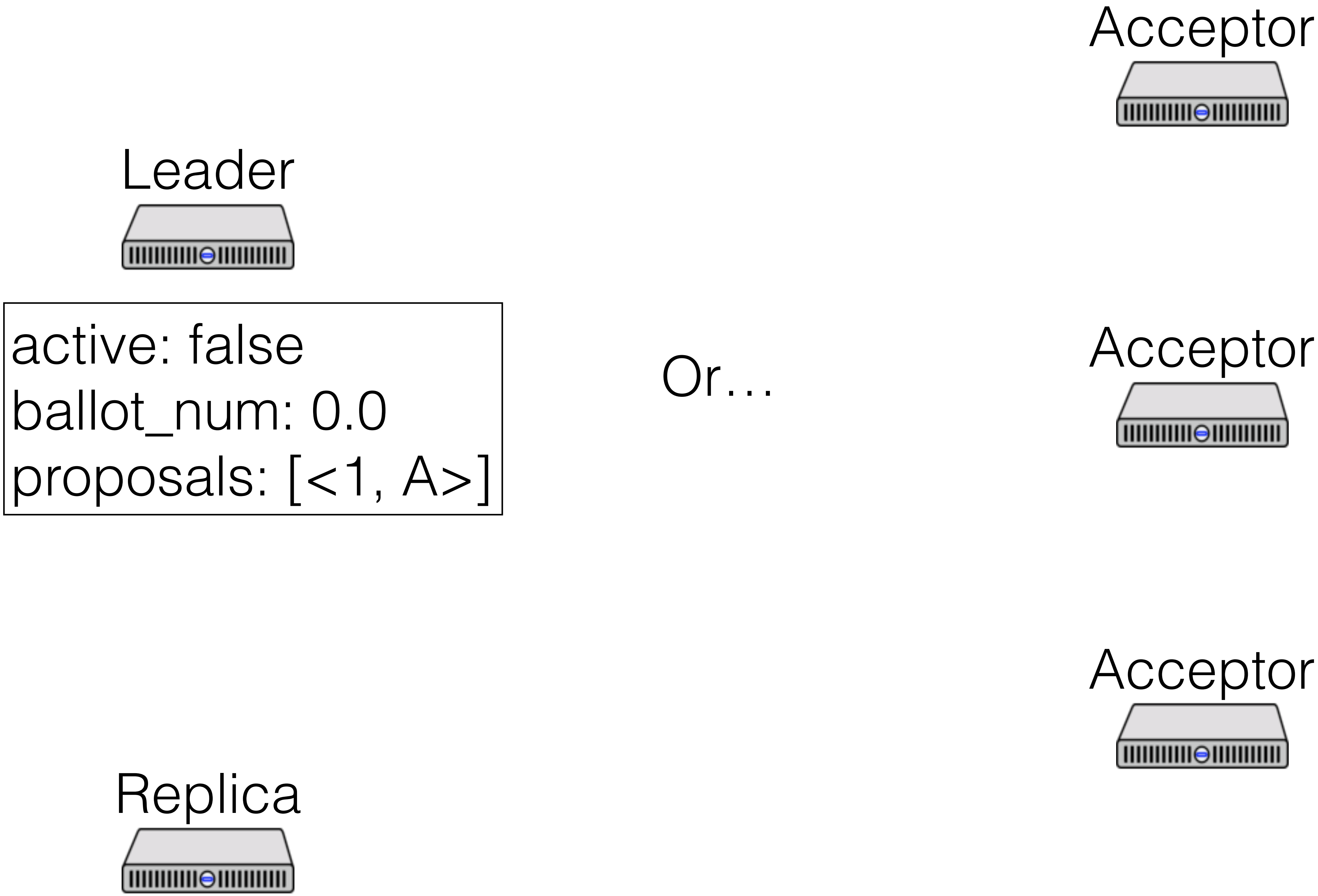
Replica

Acceptor

Acceptor

Acceptor

Leader: Handling proposals



Acceptor

Leader

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

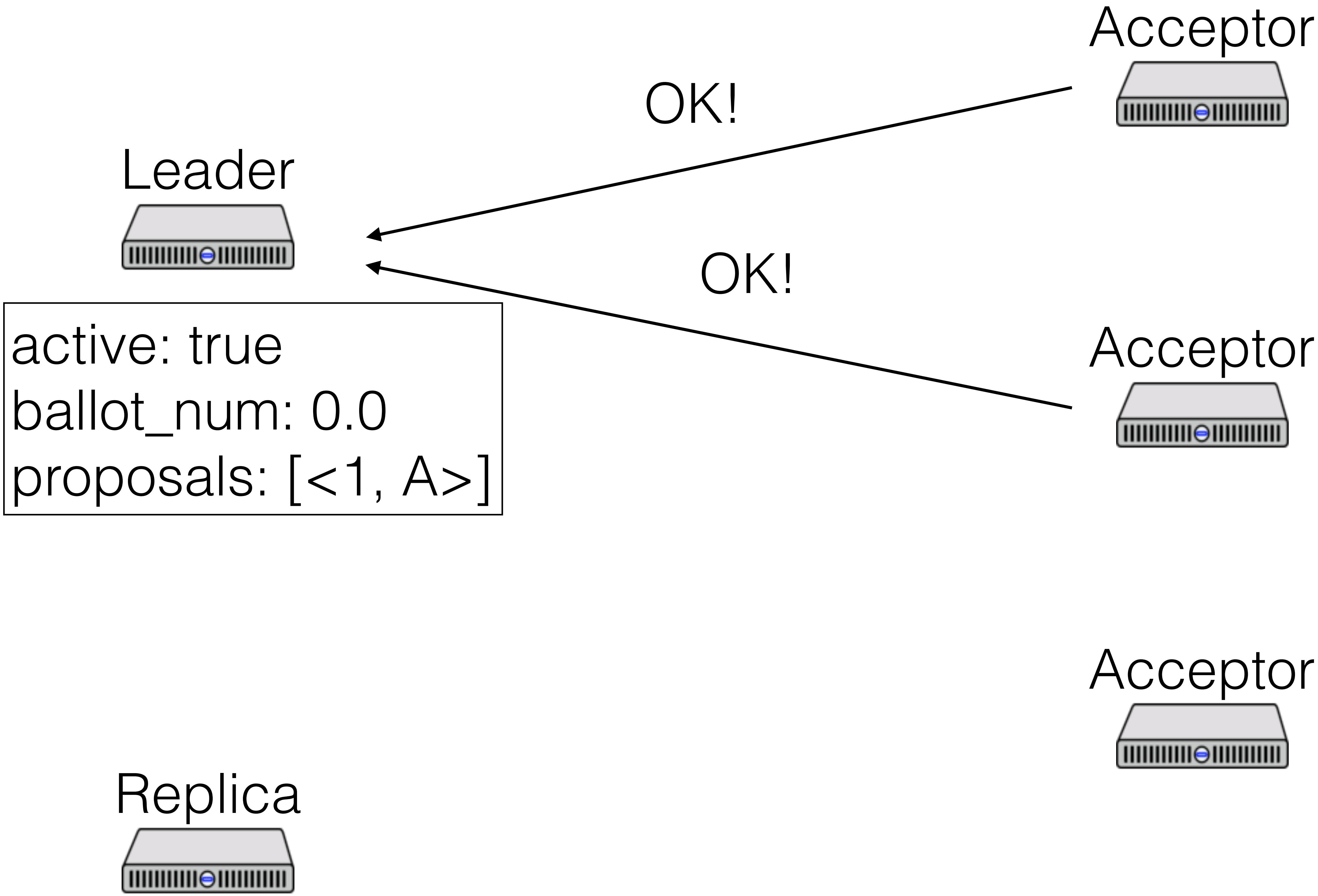
Or…

Acceptor

Acceptor

Replica

Leader: Handling proposals



Acceptor

OK!

Leader

OK!

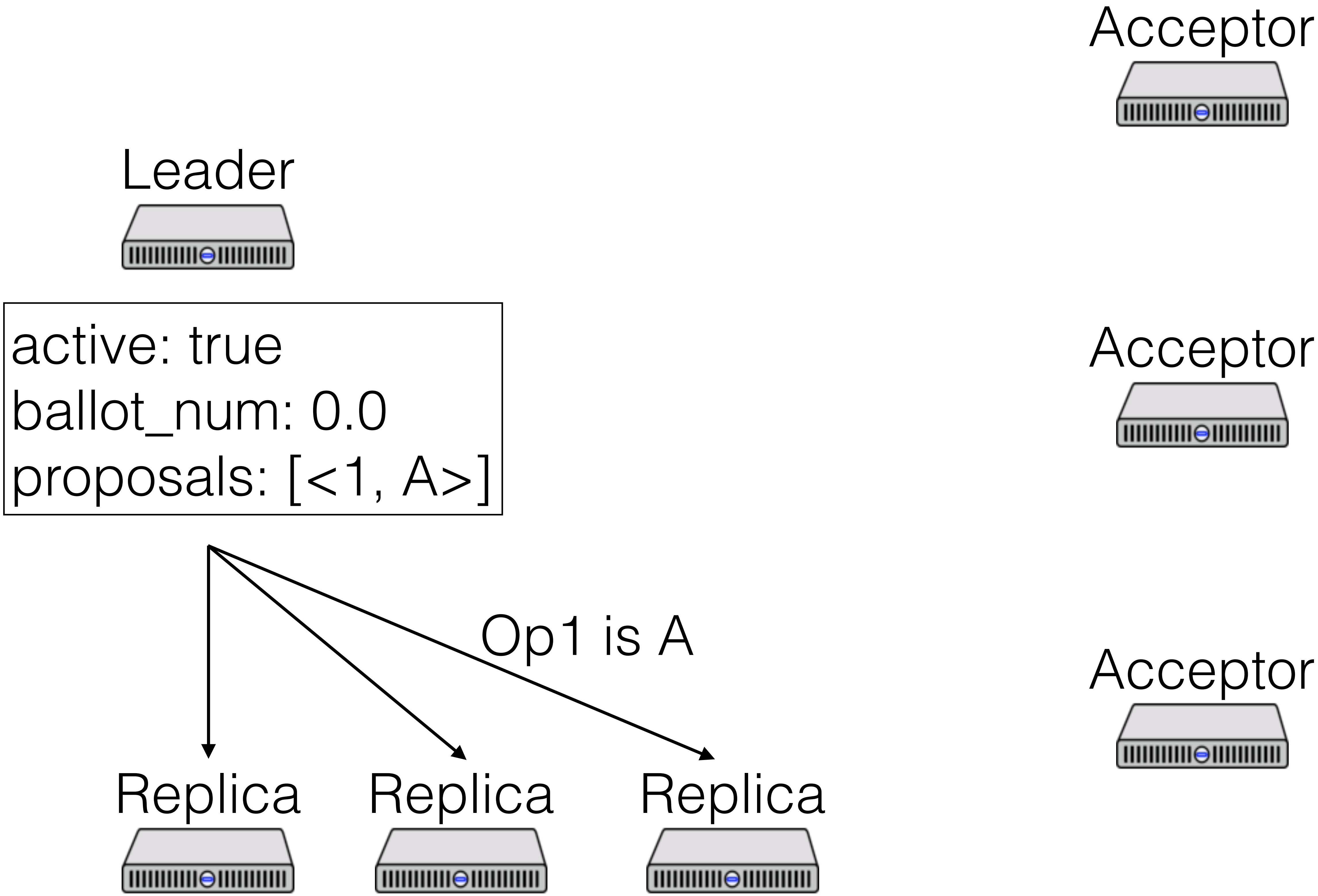
active: true
ballot_num: 0.0
proposals: [<1, A>]

Acceptor

Acceptor

Replica

Leader: Handling proposals



Acceptor

Leader

Acceptor

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

Op1 is A

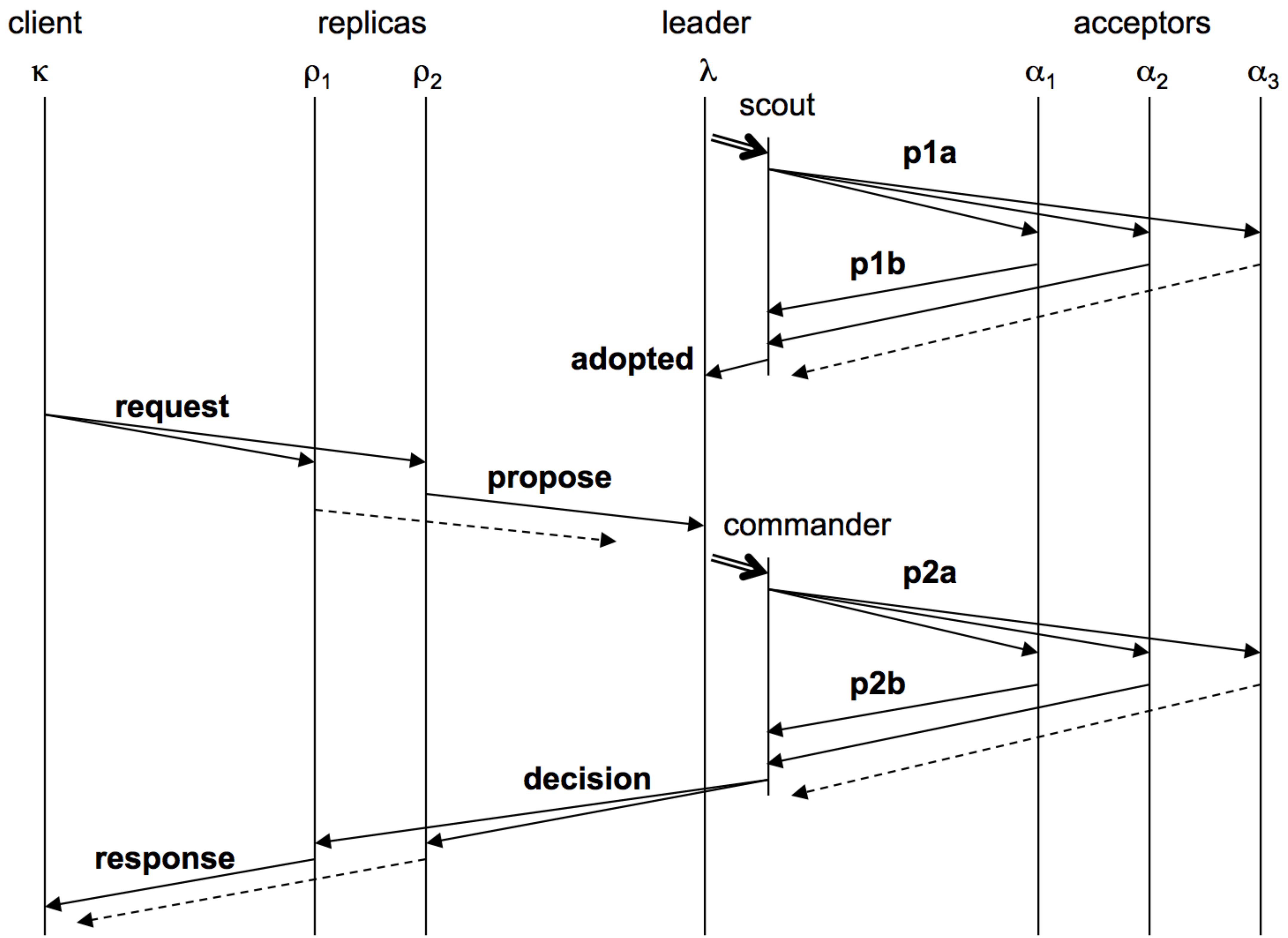
Acceptor

Replica

Replica

Replica

Paxos Made Moderately Complex Made Simple



Election revisited

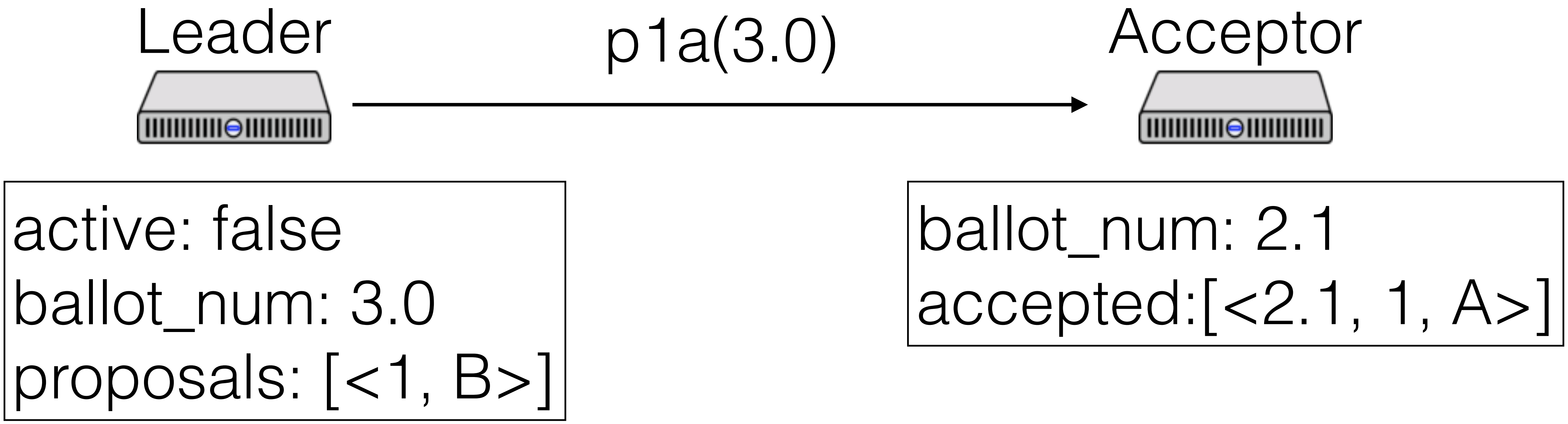
Leader

active: false
ballot_num: 3.0
proposals: [<1, B>]

Acceptor

ballot_num: 2.1
accepted:[<2.1, 1, A>]

Election revisited



Leader

p1a(3.0)

Acceptor

active: false
ballot_num: 3.0
proposals: [<1, B>]

ballot_num: 2.1
accepted:[<2.1, 1, A>]

Election revisited

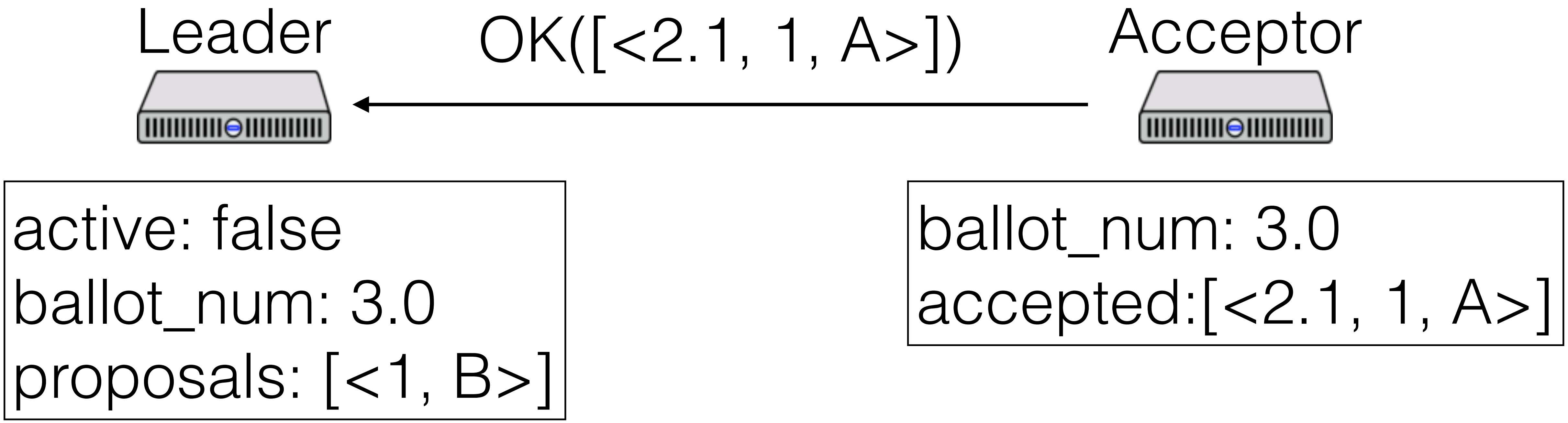
Leader

active: false
ballot_num: 3.0
proposals: [<1, B>]

Acceptor

ballot_num: 3.0
accepted:[<2.1, 1, A>]

Election revisited



Leader

OK([<2.1, 1, A>])

Acceptor

active: false
ballot_num: 3.0
proposals: [<1, B>]

ballot_num: 3.0
accepted:[<2.1, 1, A>]

Election revisited

Leader

active: true
ballot_num: 3.0
proposals: [<1, A>]

Acceptor

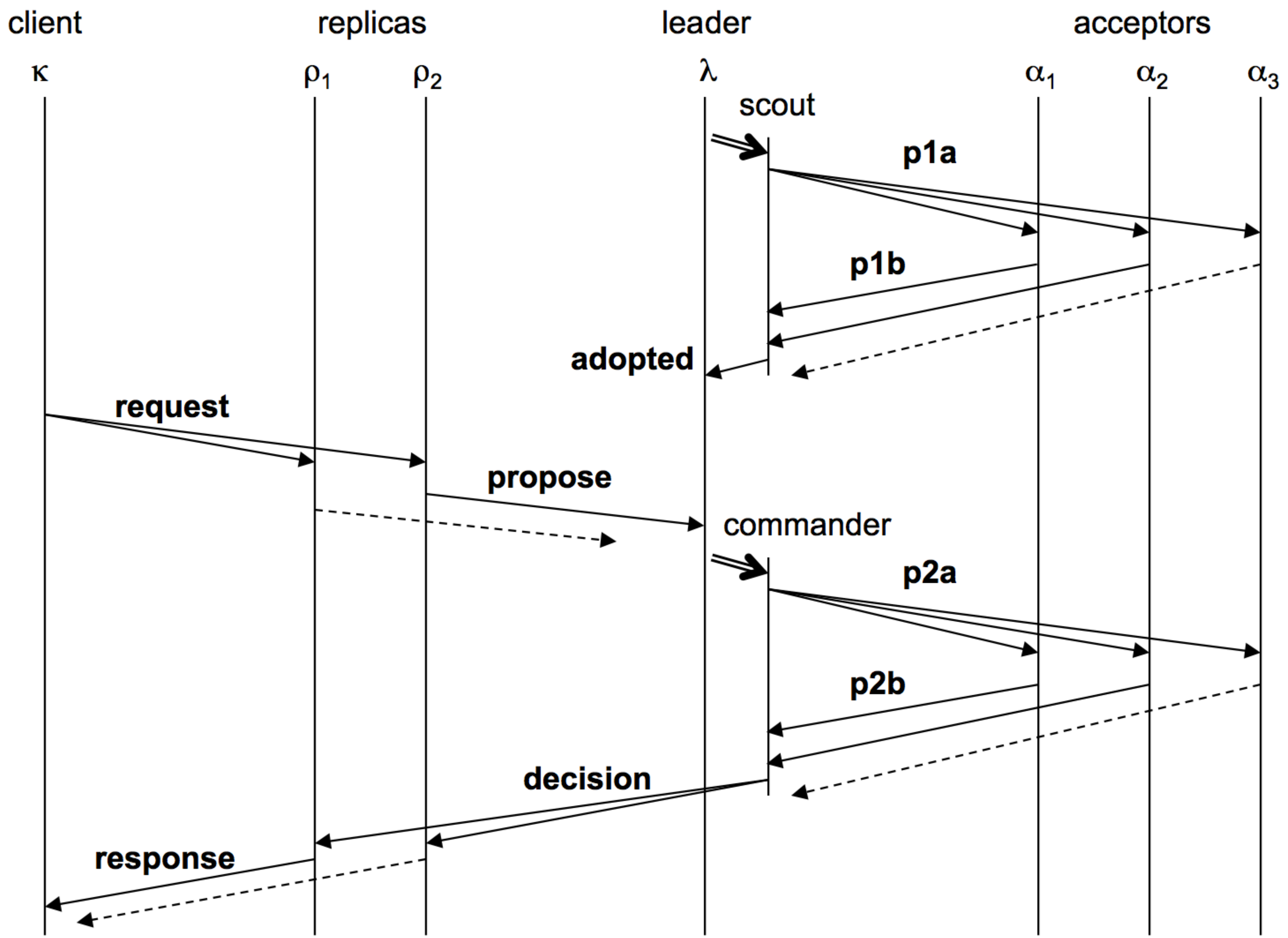
ballot_num: 3.0
accepted:[<2.1, 1, A>]

Leaders

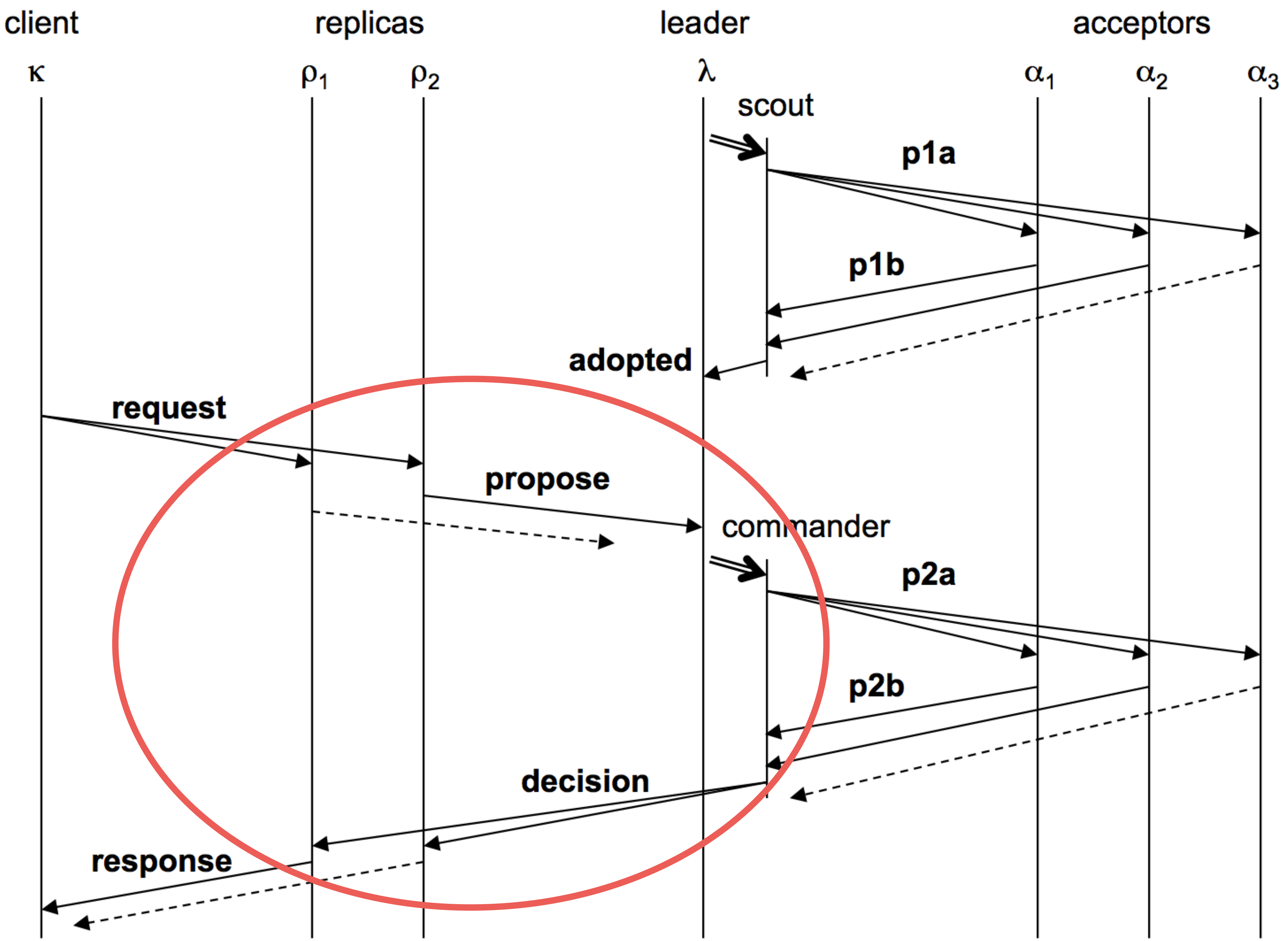
- Only propose one value per ballot and slot

- 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

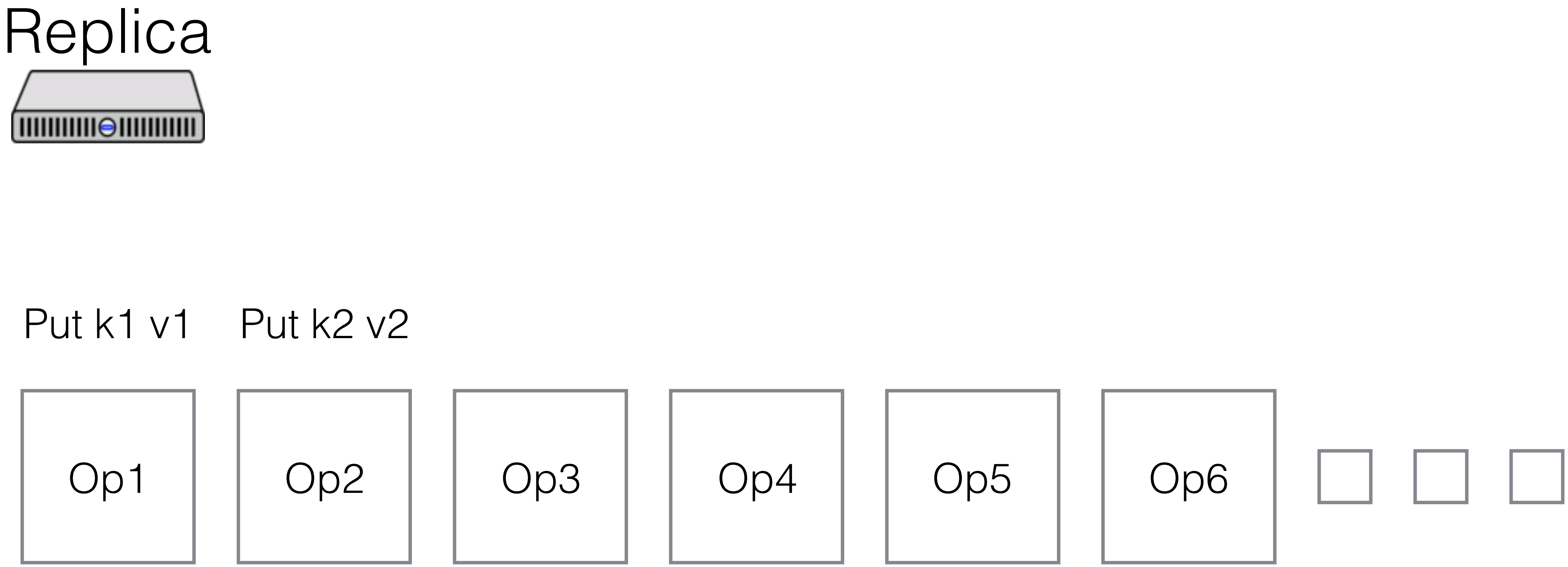
Paxos Made Moderately Complex Made Simple



Paxos Made Moderately Complex Made Simple



Replicas



Replica

Put k1 v1

Put k2 v2

Op1

Op2

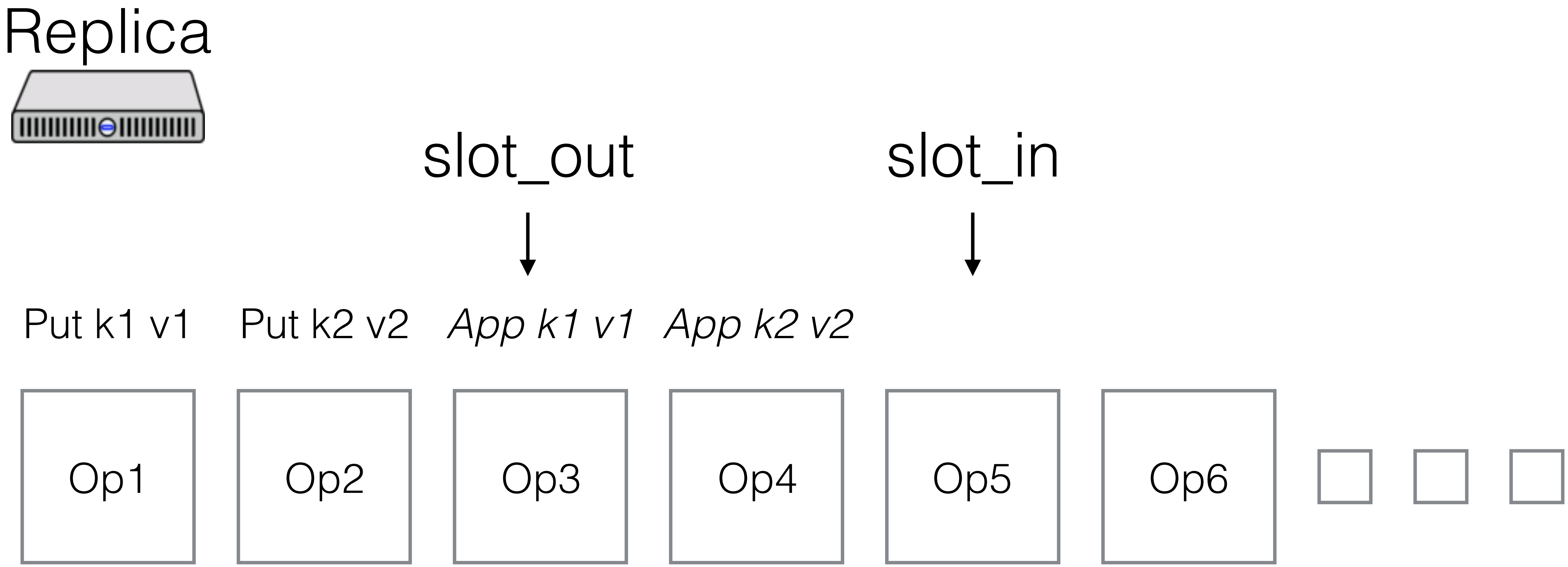
Op3

Op4

Op5

Op6

Replicas



Replica

slot_out

slot_in

Put k1 v1

Put k2 v2

App k1 v1

App k2 v2

Op1

Op2

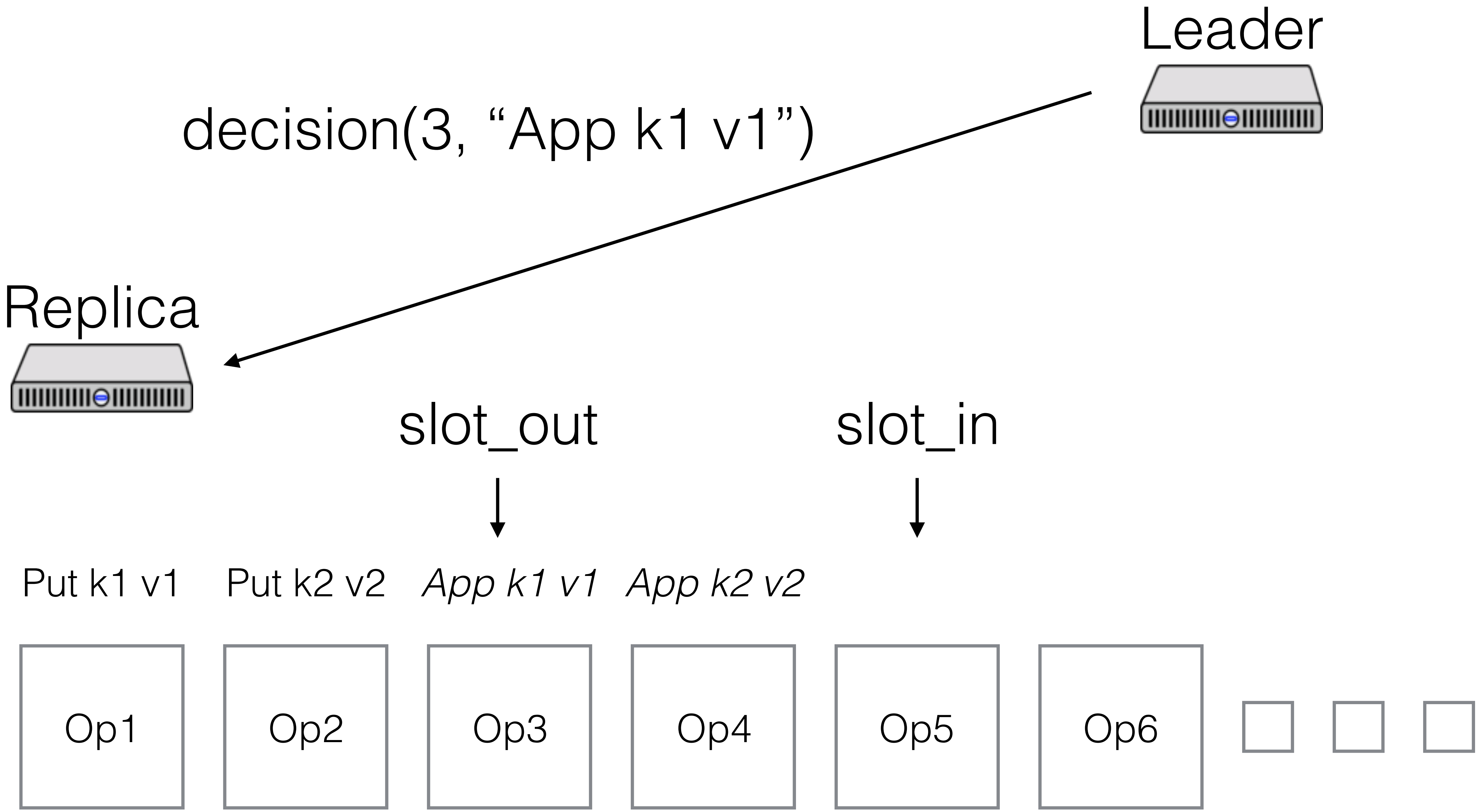
Op3

Op4

Op5

Op6

Replicas



Leader

decision(3, “App k1 v1”)

Replica

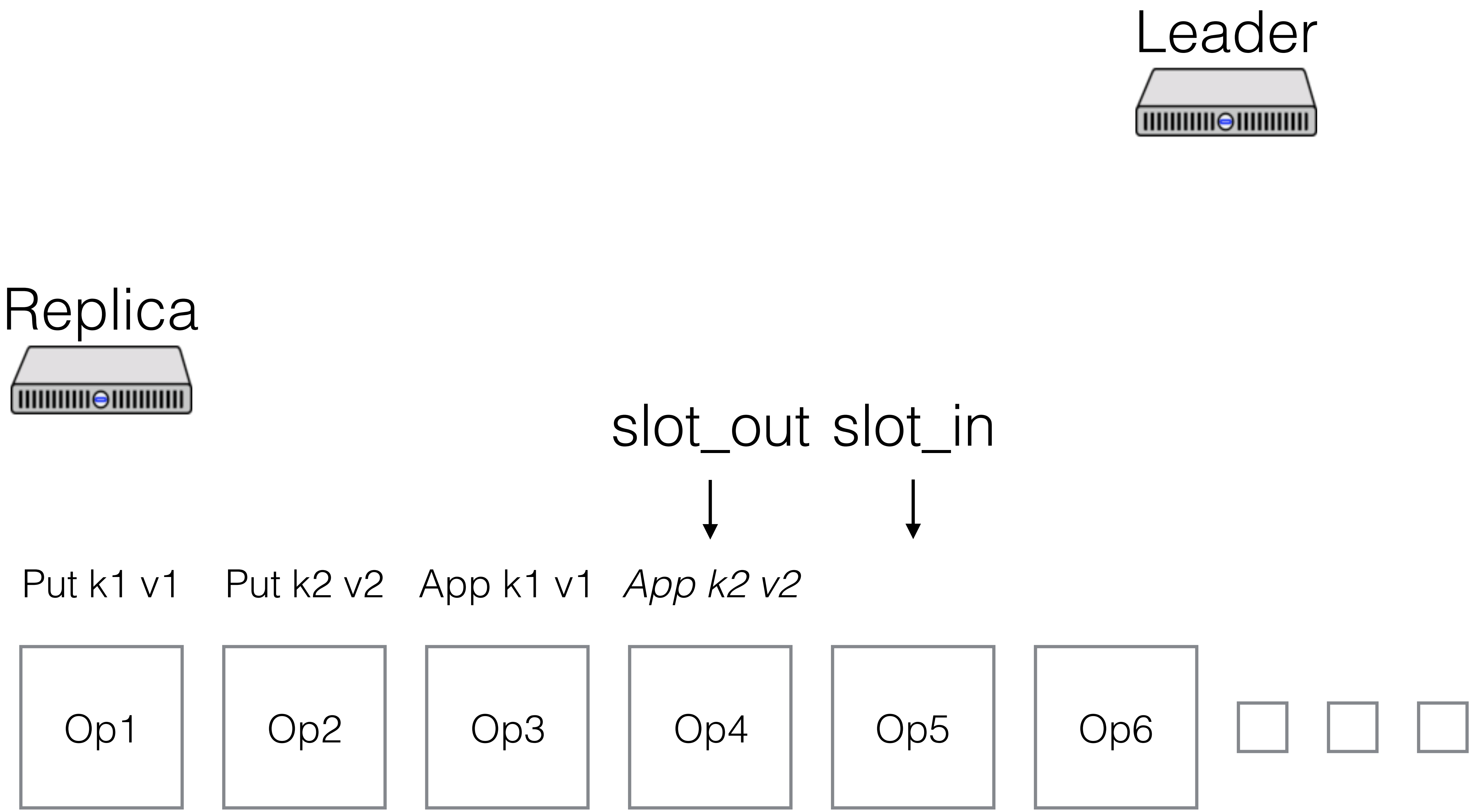
slot_out

slot_in

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

Op1 | Op2 | Op3 | Op4 | Op5 | Op6 | □ | □ | □

Replicas



Leader

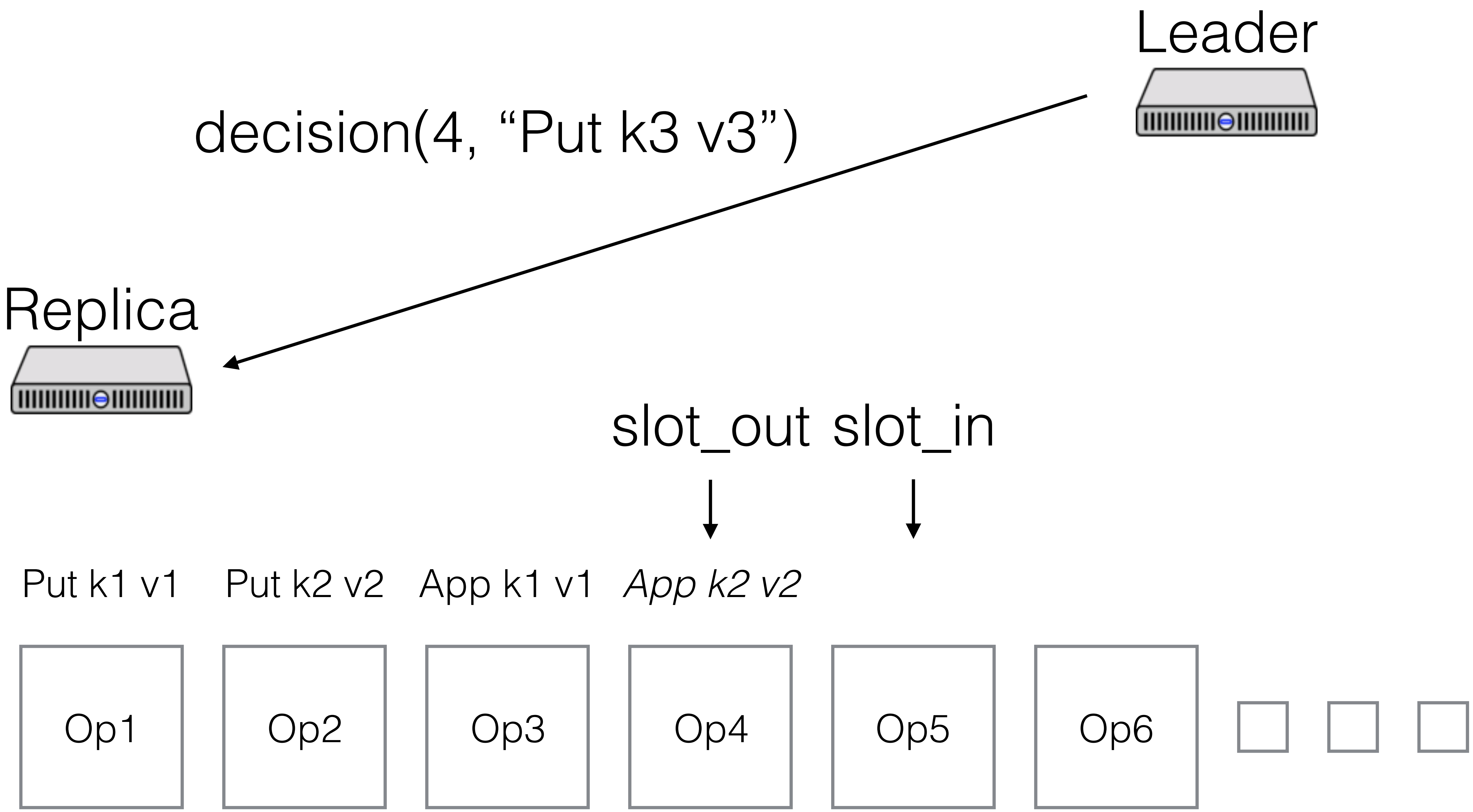
Replica

slot_out slot_in

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

Op1 Op2 Op3 Op4 Op5 Op6

Replicas



Leader

decision(4, “Put k3 v3”)

Replica

slot_out slot_in

Put k1 v1

Put k2 v2

App k1 v1

App k2 v2

Op1

Op2

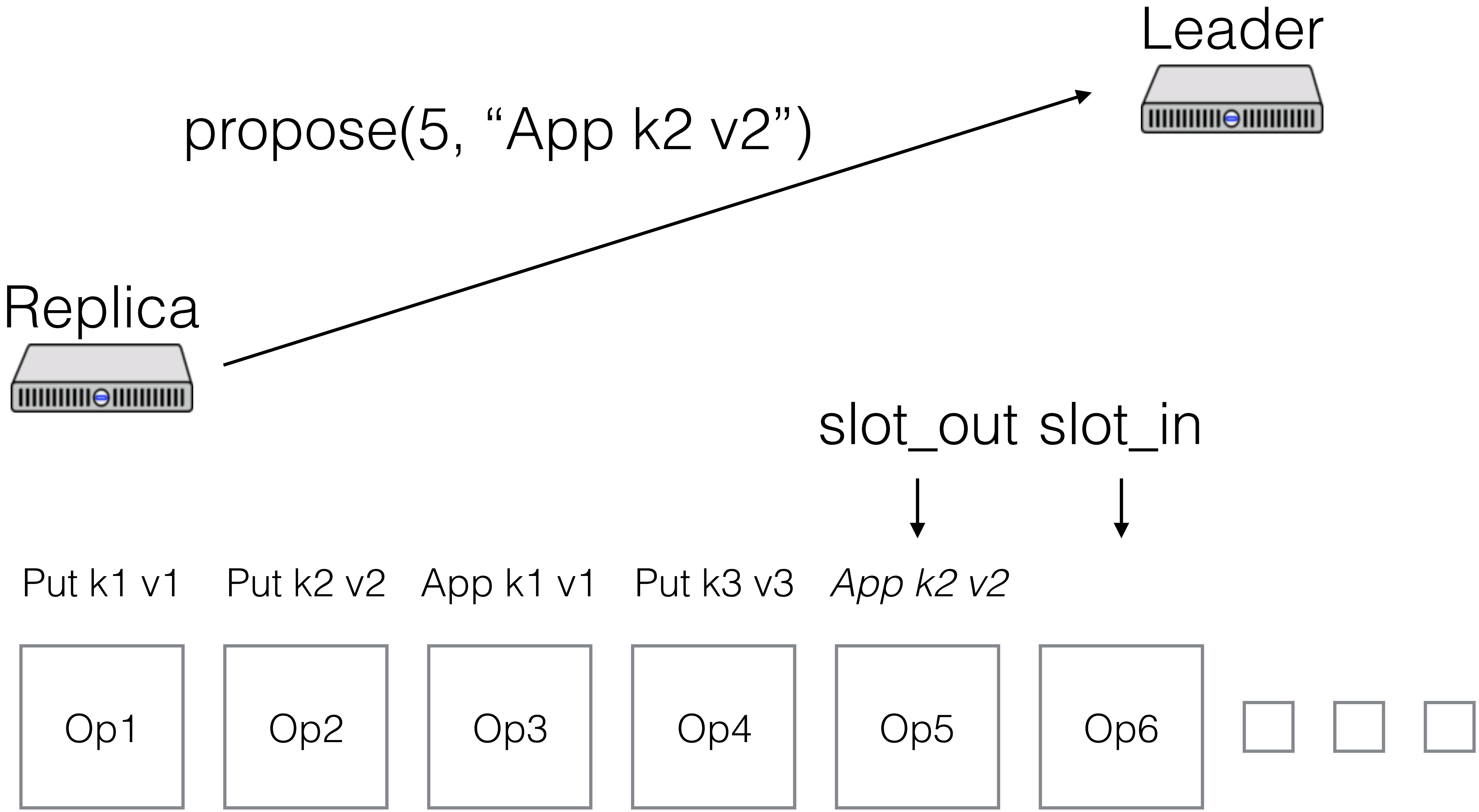
Op3

Op4

Op5

Op6

Replicas



Leader

propose(5, “App k2 v2”)

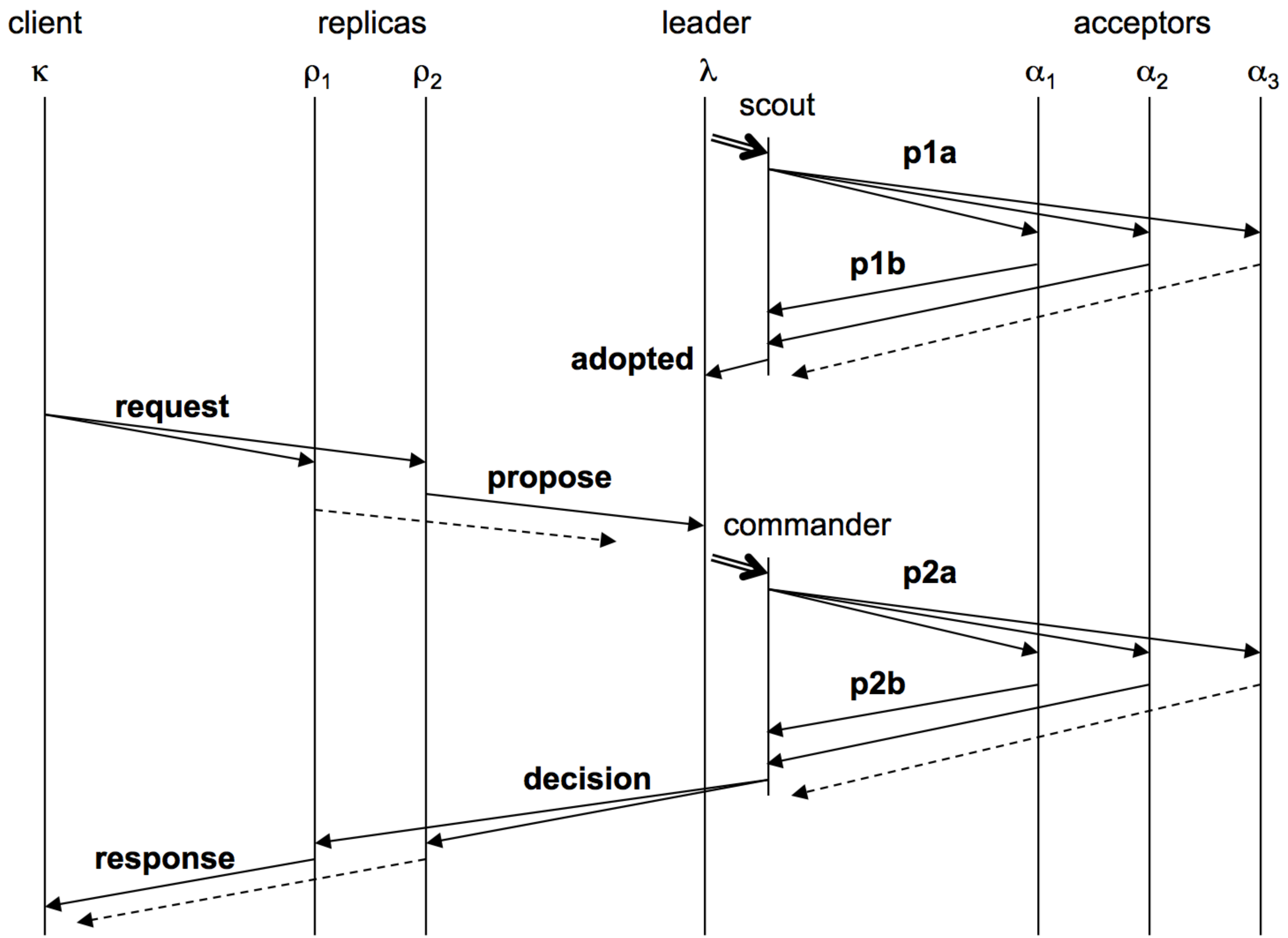
Replica

slot_out slot_in

Put k1 v1 Put k2 v2 App k1 v1 Put k3 v3 *App k2 v2*

Op1 Op2 Op3 Op4 Op5 Op6

Paxos Made Moderately Complex Made Simple



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!

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?

Reconfiguration

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

- Commit no-ops until WINDOW is cleared

Other complications

State simplifications

- Can track much less information, esp. on replicas

Garbage collection

- Unbounded memory growth is bad
- Lab 3: track finished slots across all instances, garbage collect when everyone has learned result

Read-only commands

- Can’t just read from replica (why?)
- But, don’t need their own slot