Paxos Made Simple

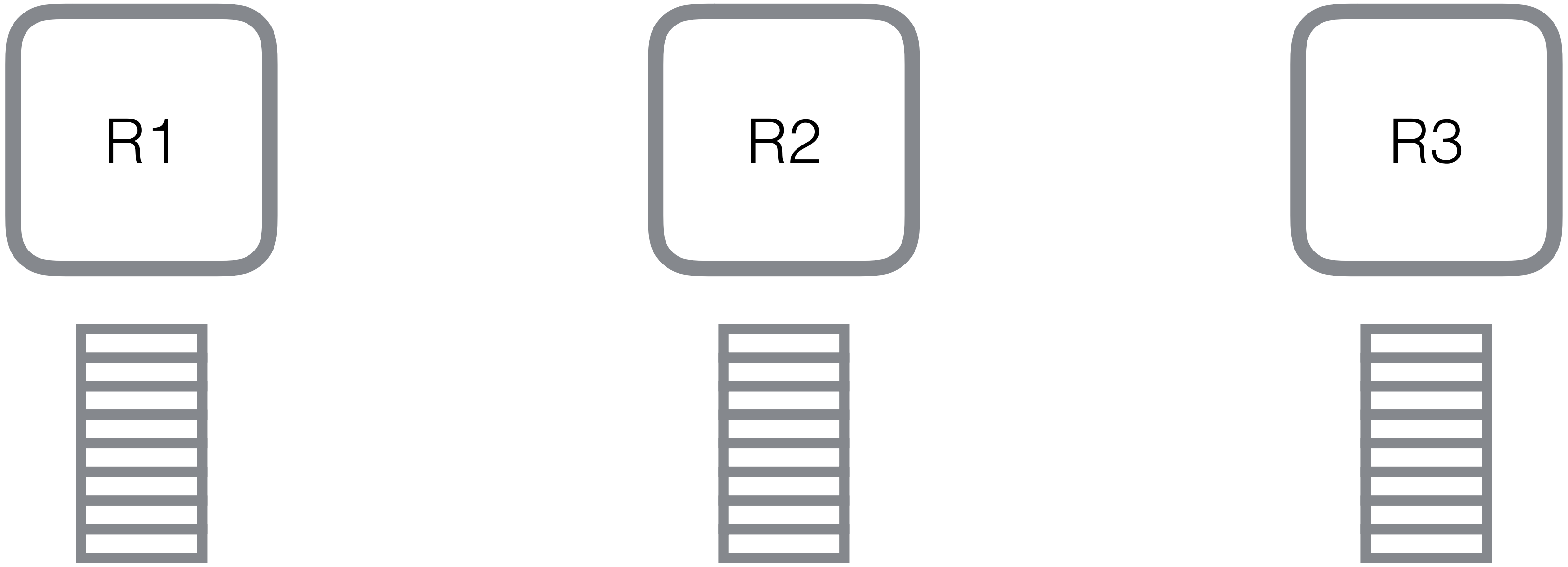
Leslie Lamport

Background

- Problem: fault tolerance in distributed systems
- Impact: needed everywhere in distributed systems!
    - Distributed databases
    - SDN controllers
    - …

Replicated State Machines

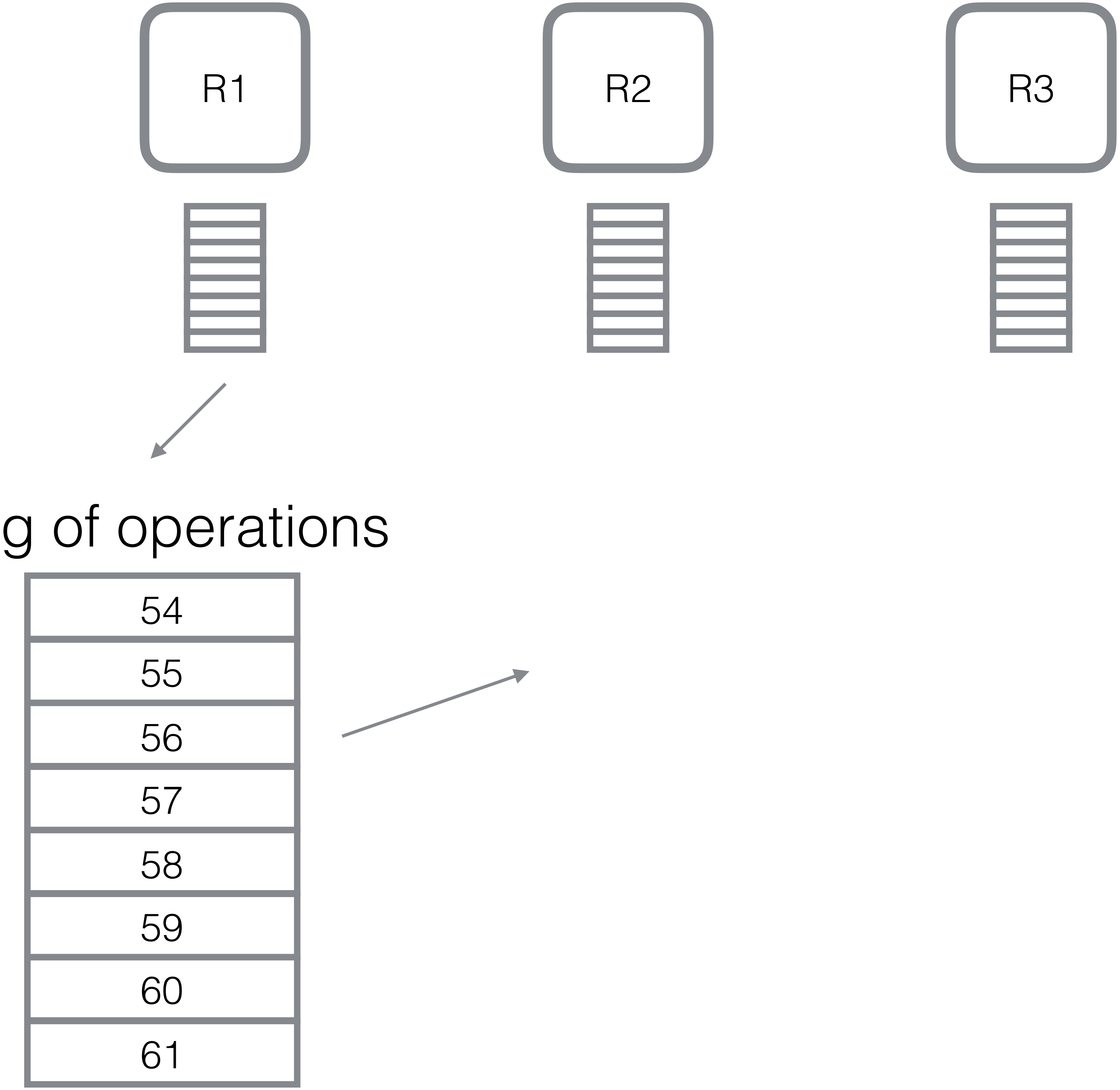
- Several servers, each is a SM
- Current state + command —> new state + output
- Deterministic
- Execute the same set of commands



R1

R2

R3



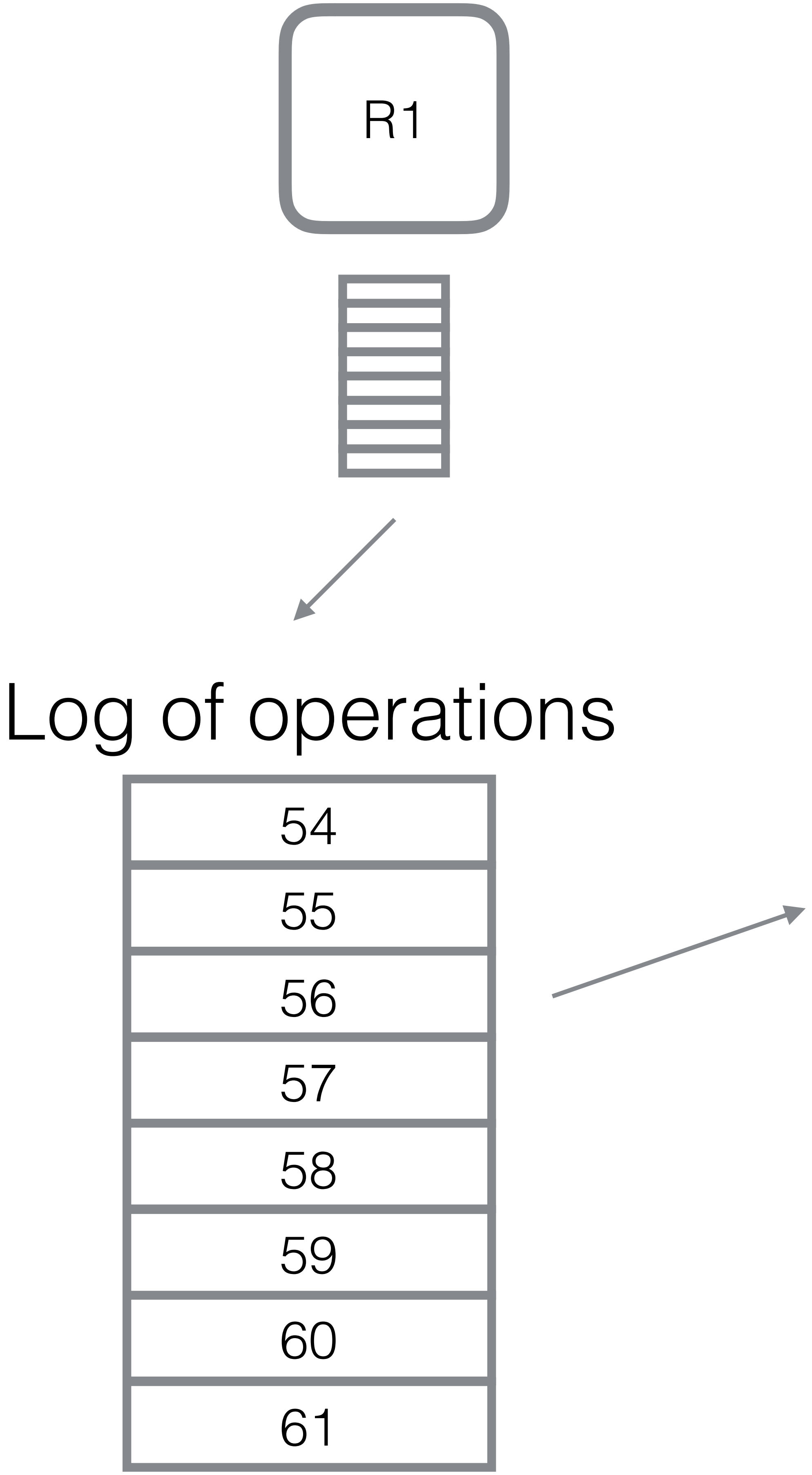
R1

R2

R3

Log of operations

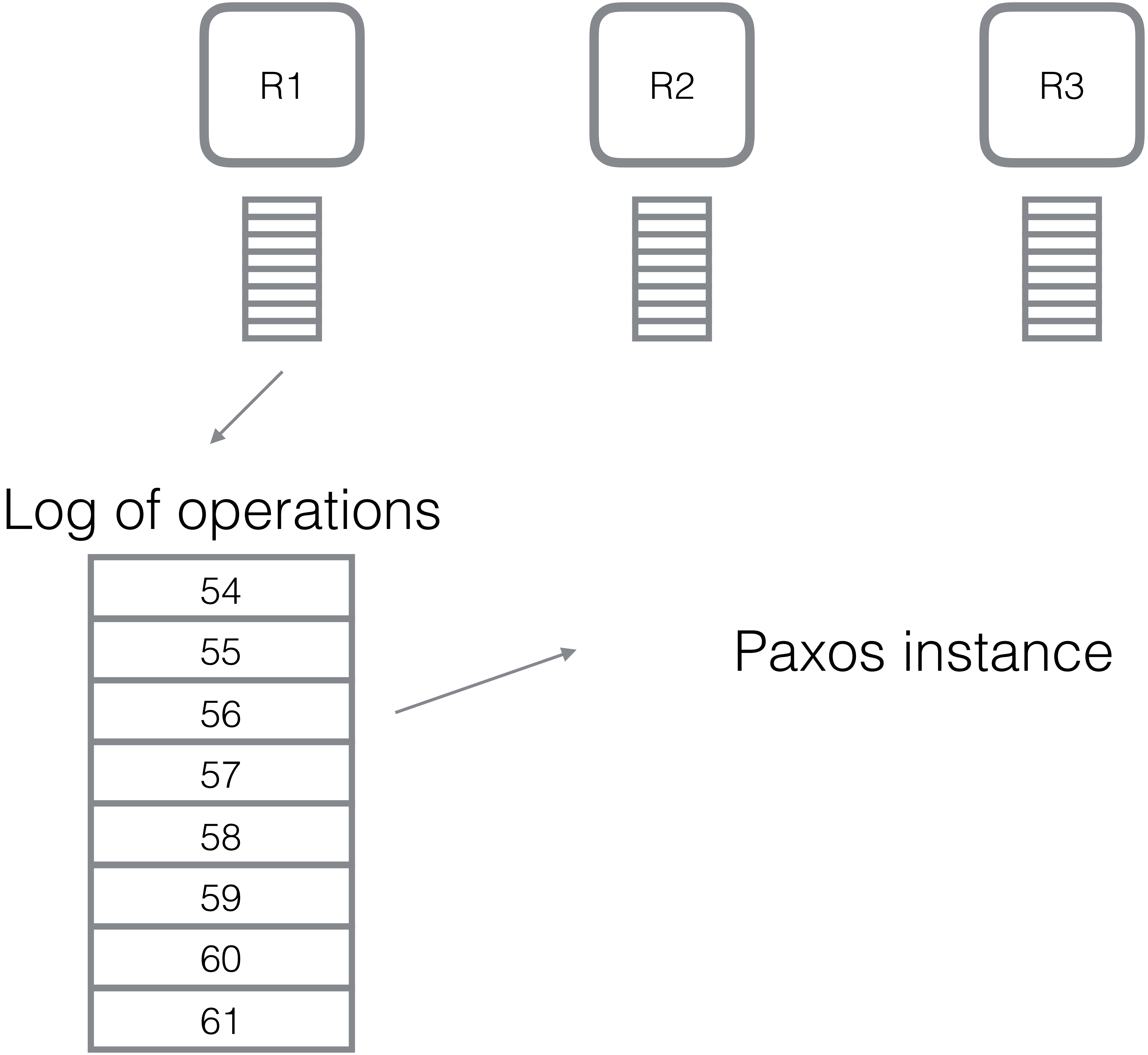
54
55
56
57
58
59
60
61



Log of operations

54
55
56
57
58
59
60
61

Problem:
distributed consensus



R1

R2

R3

Log of operations

54

55

56

57

58

59

60

61

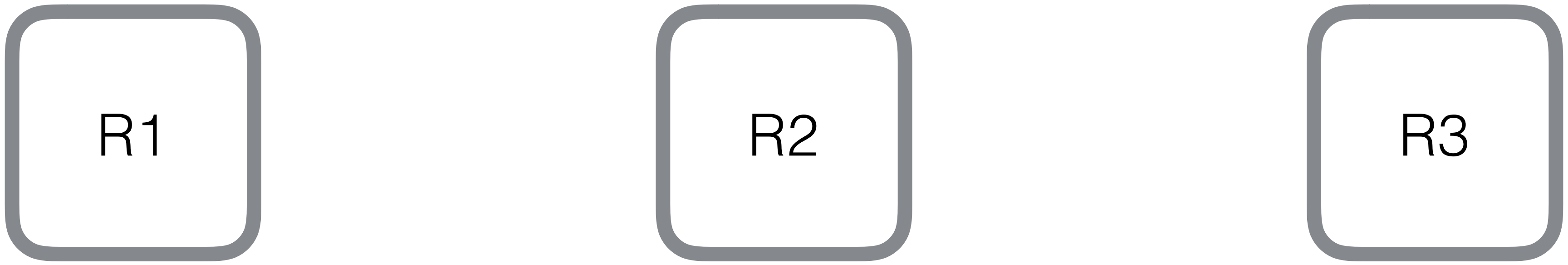
Paxos instance

3 roles: Proposers, Acceptors, Learners

Proposers: propose (m, v)
Acceptors: accept proposals and choose values
Learners: learn the chosen values

3 roles: Proposers, Acceptors, Learners

Proposers: propose (m, v)
Acceptors: accept proposals and choose values
Learners: learn the chosen values



R1

Proposer
Acceptor
Learner

R2

Proposer
Acceptor
Learner

R3

Proposer
Acceptor
Learner

- Safety
  - Only a proposed value is chosen
  - Only one value is chosen
  - A process does not learn a value has been chosen unless it actually has been

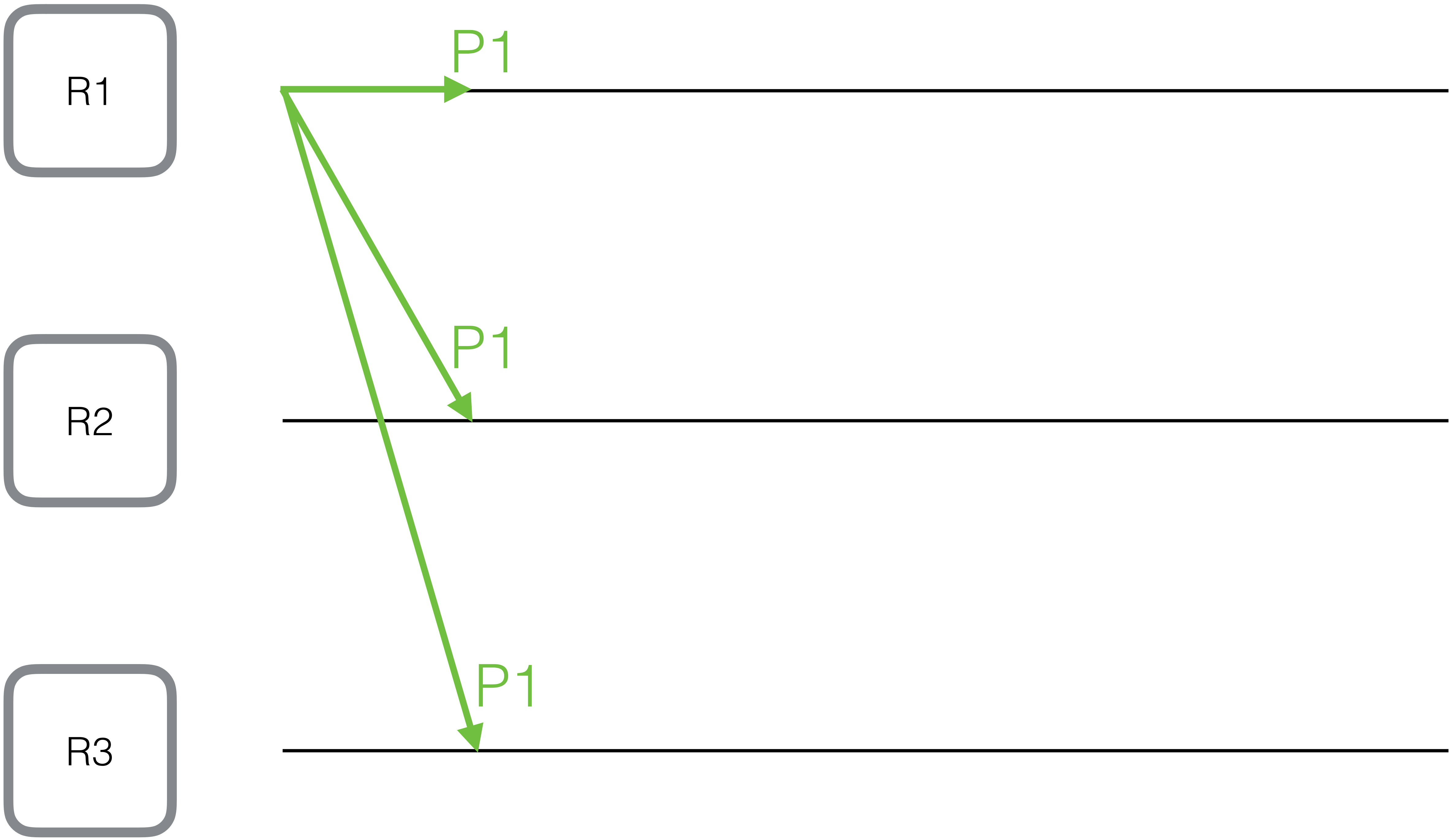
- Safety
    - Only a proposed value is chosen
    - Only one value is chosen
    - A process does not learn a value has been chosen unless it actually has been
- Assumptions:
    - Non-Byzantine failures: can fail by stopping, can restart
    - Messages take arbitrarily long to deliver, can duplicate, can be lost, but not corrupted

Invariants

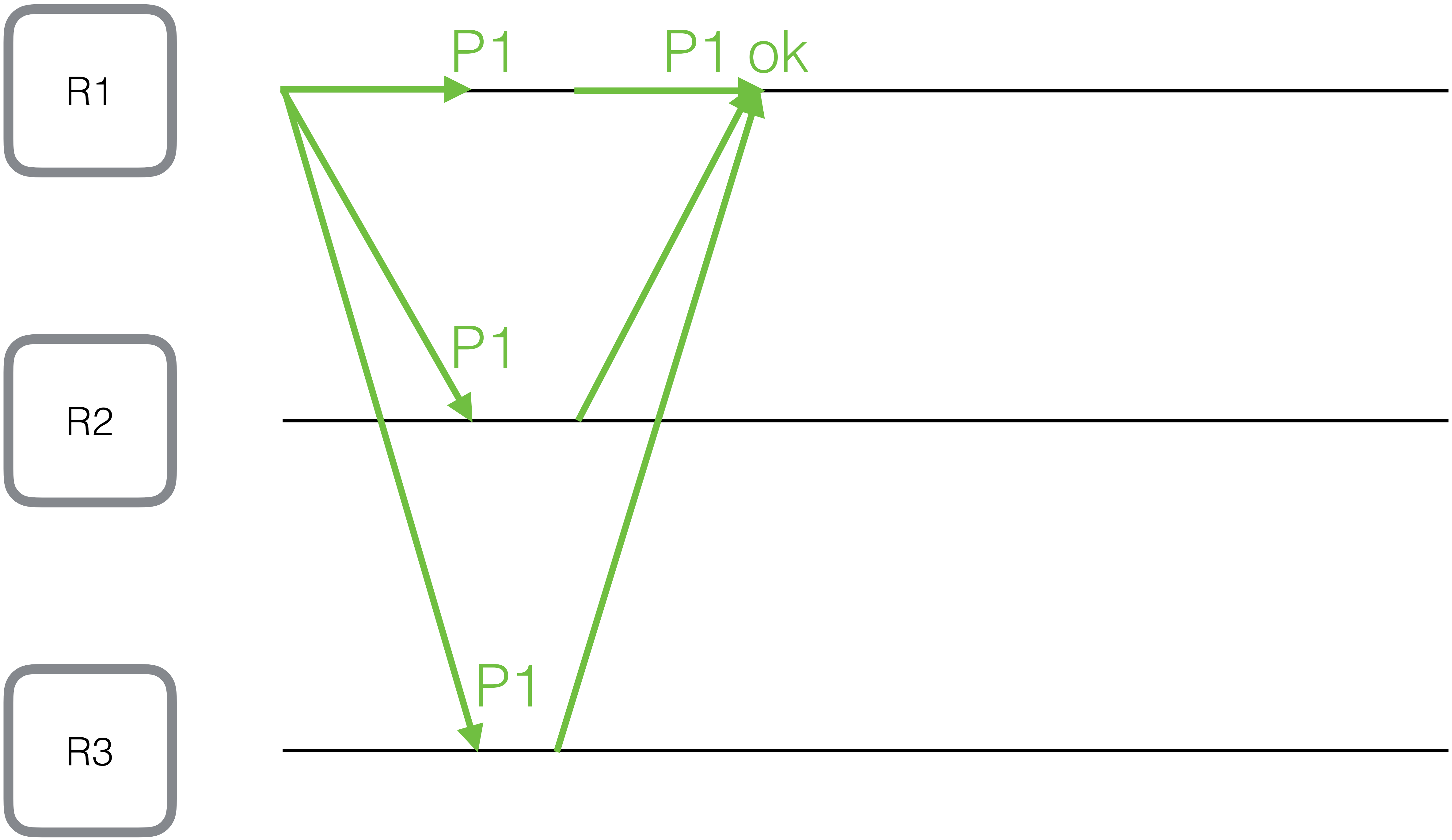
- An acceptor can accept a proposal numbered *n* iff it has not responded to a *prepare* request having a number greater than *n*
- For any *v* and *n*, if a proposal with value *v* and number *n* is issued, then there is a set *S* consisting of a majority of acceptors such that either
  - no acceptor in *S* has accepted any proposal numbered less than *n* or
  - *v* is the value of the highest-numbered proposal among all proposals numbered less than *n* accepted by the acceptors in *S*

- Phase 1: Prepare
  - Proposer proposes proposal number n
  - Acceptor responds if n > any prepare request to which it has responded
- Phase 2: Accept
  - Proposer proposes (n, v) such that v = value of highest number proposal among phase 1 responses, or any if no reported proposal
  - Acceptor can accept request for a proposal unless it has already responded to a prepare request having a number greater than n

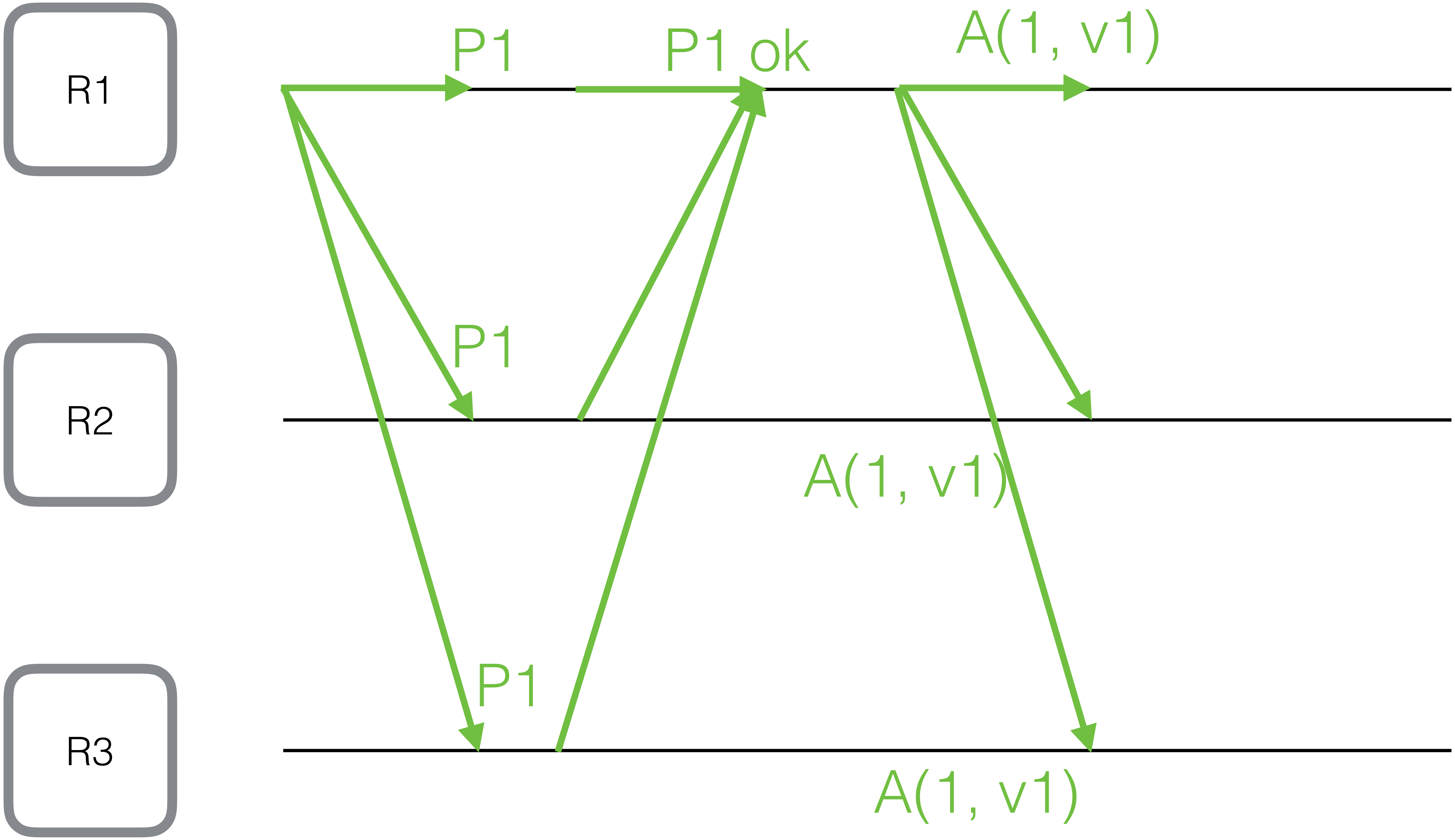
Example 1: Normal operation



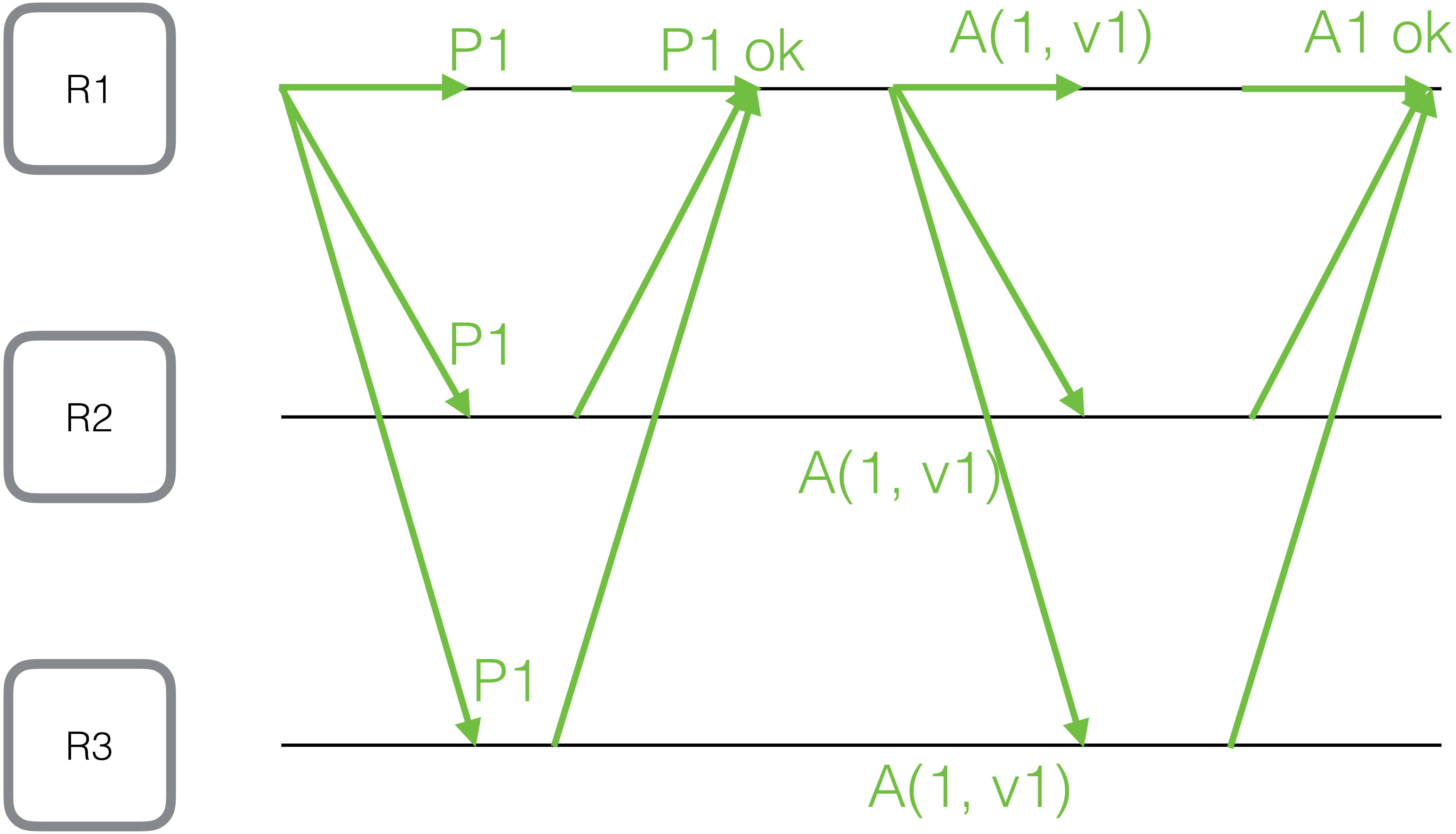
Example 1: Normal operation



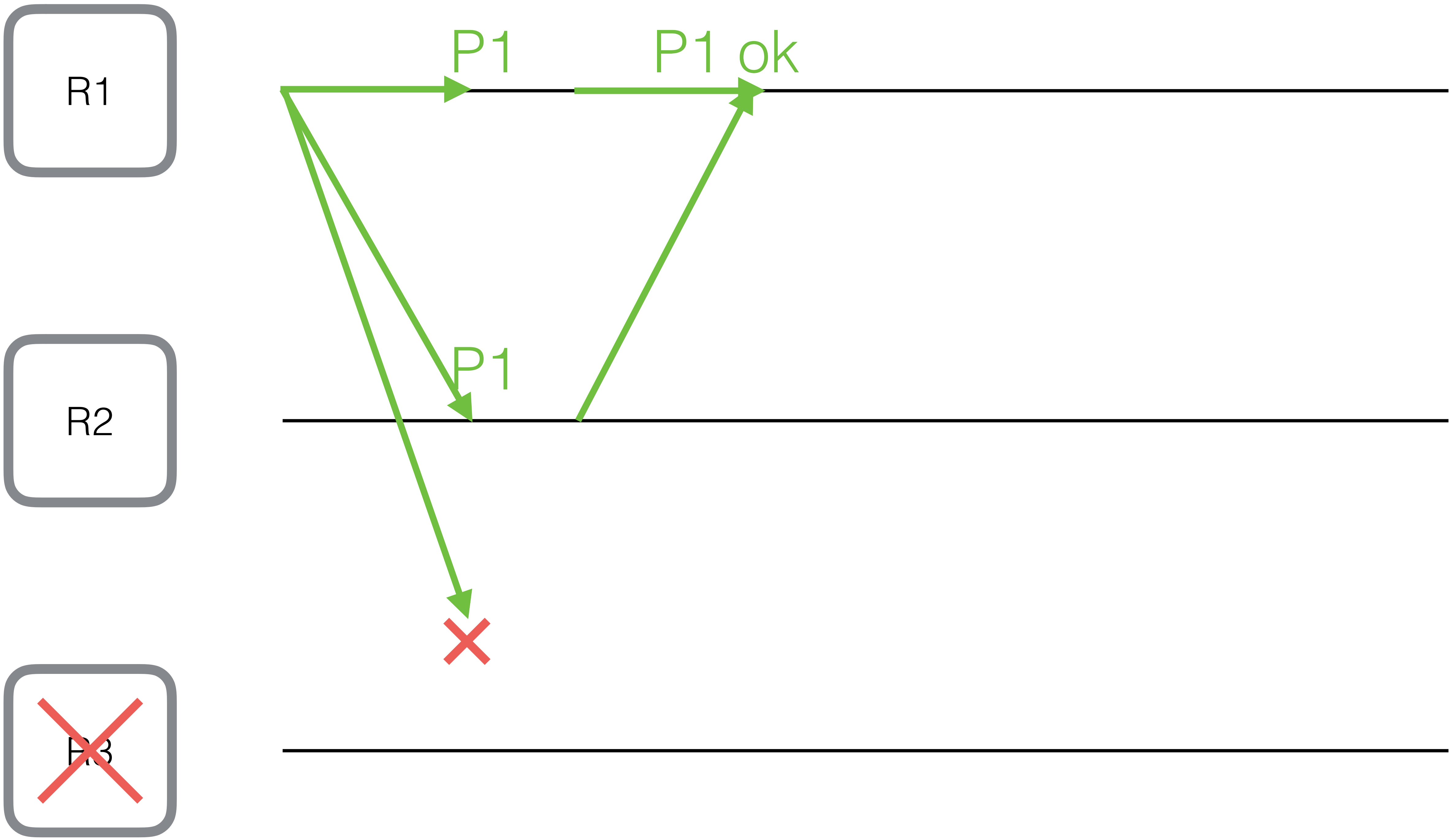
Example 1: Normal operation



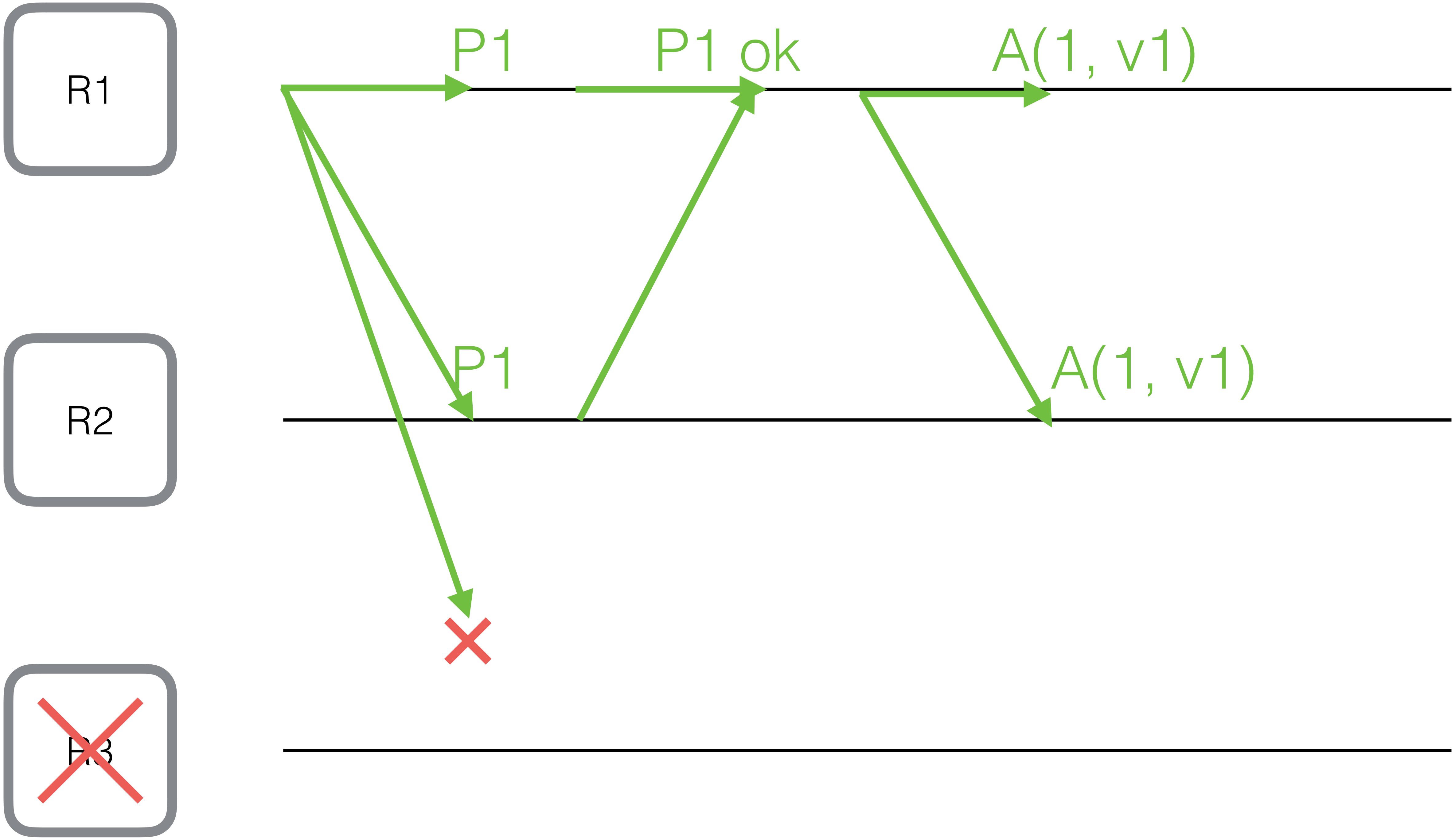
Example 1: Normal operation



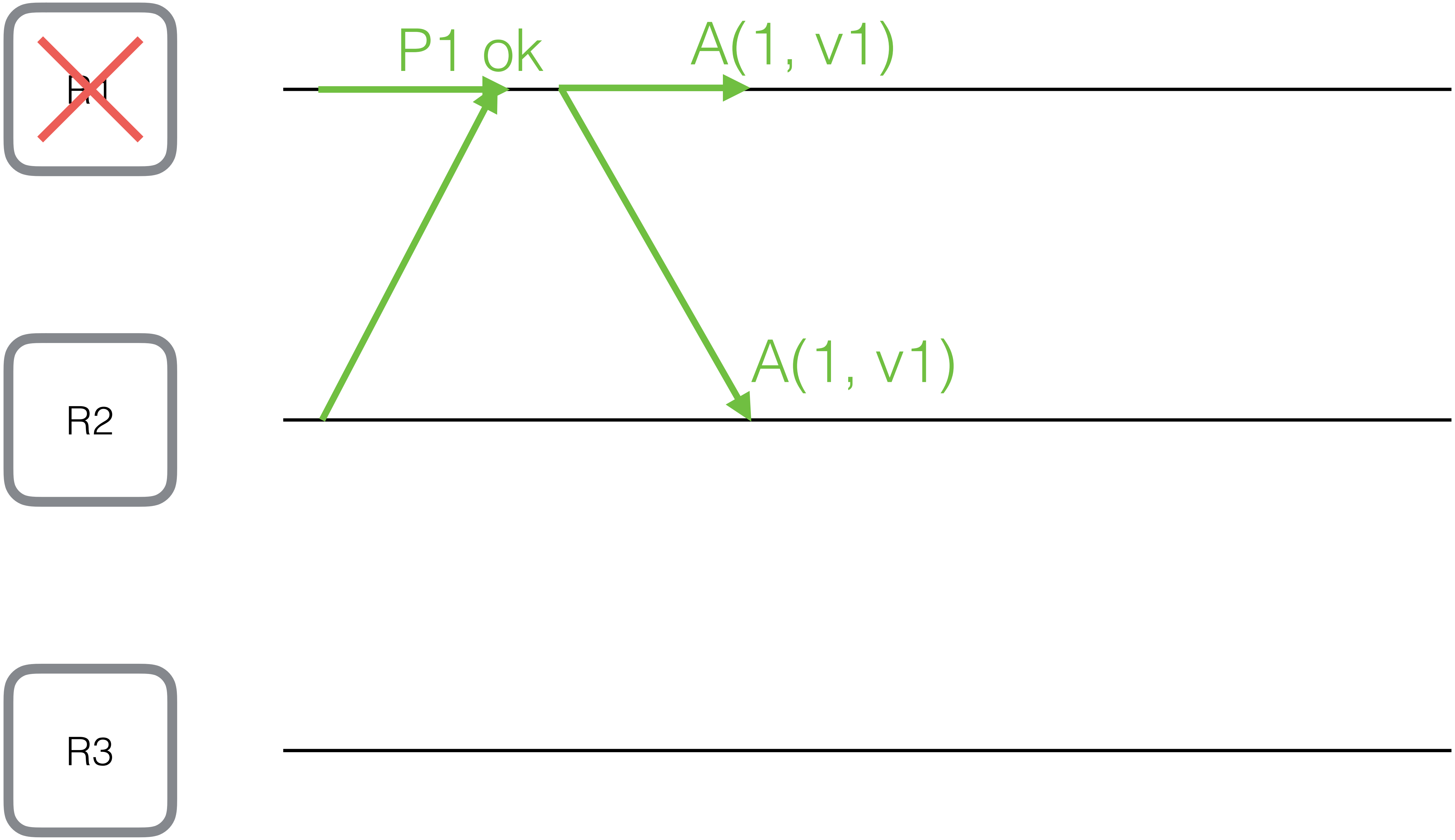
Example 2: Multiple proposers



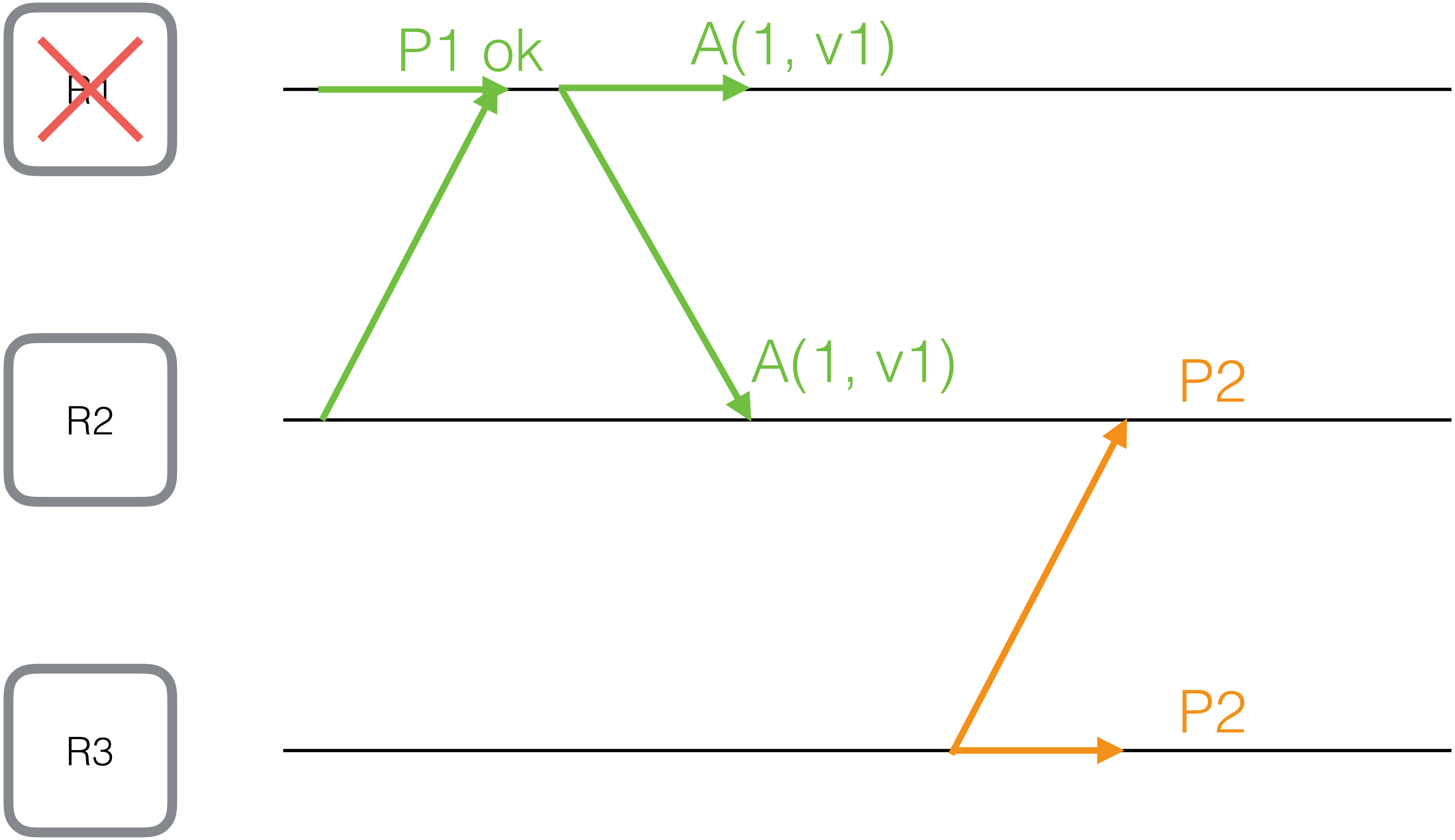
Example 2: Multiple proposers



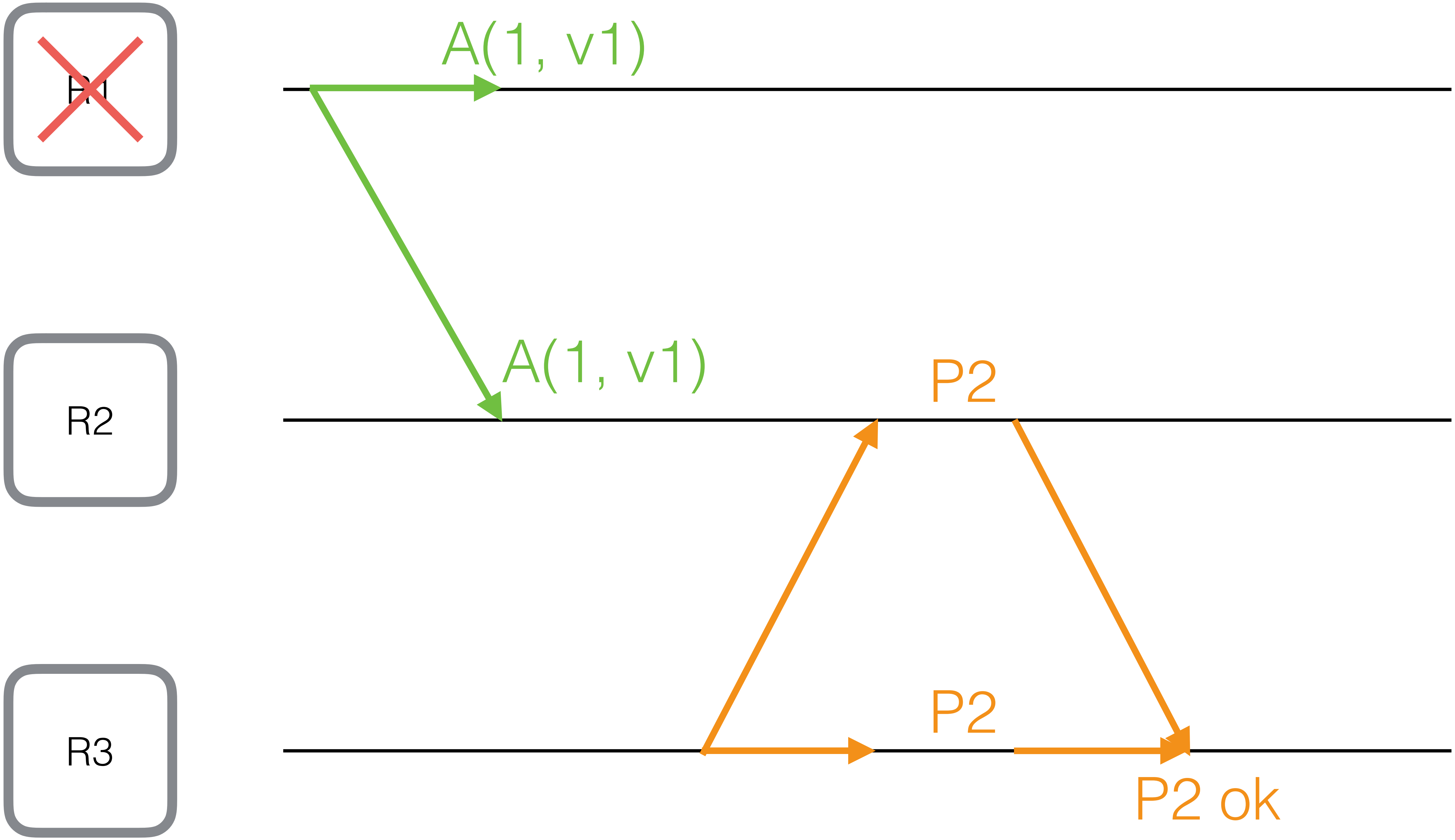
Example 2: Multiple proposers



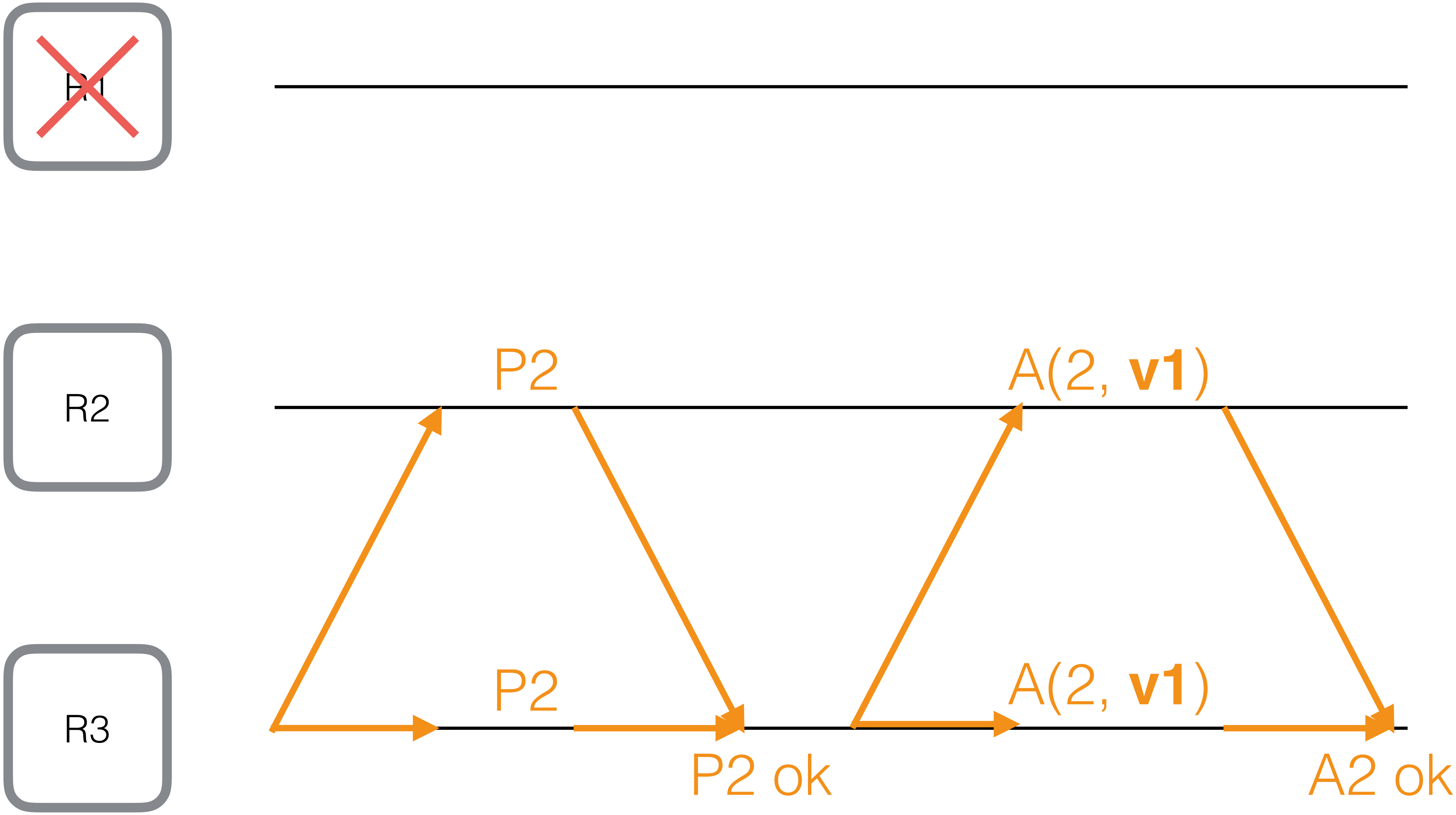
Example 2: Multiple proposers



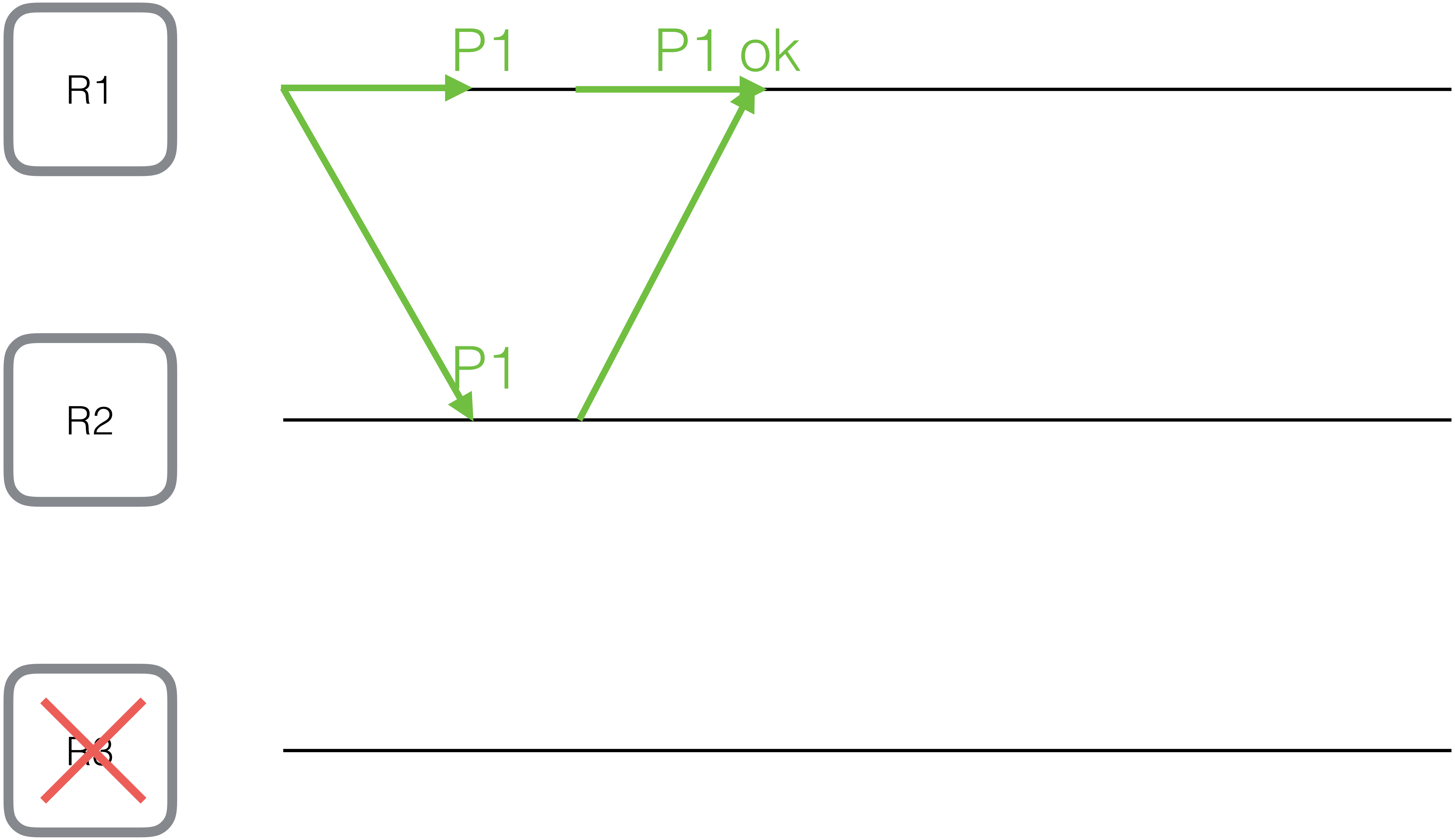
Example 2: Multiple proposers



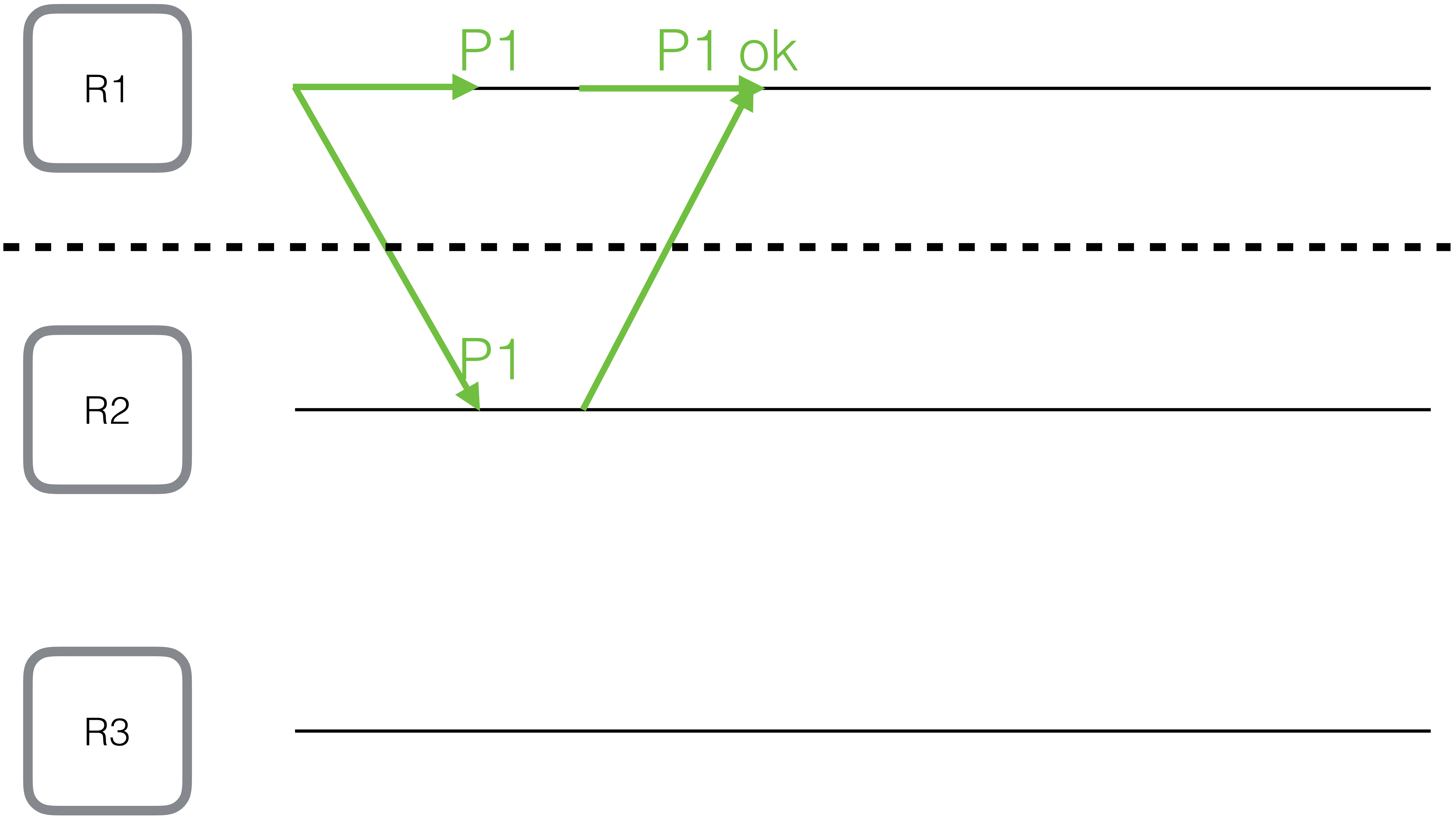
Example 2: Multiple proposers



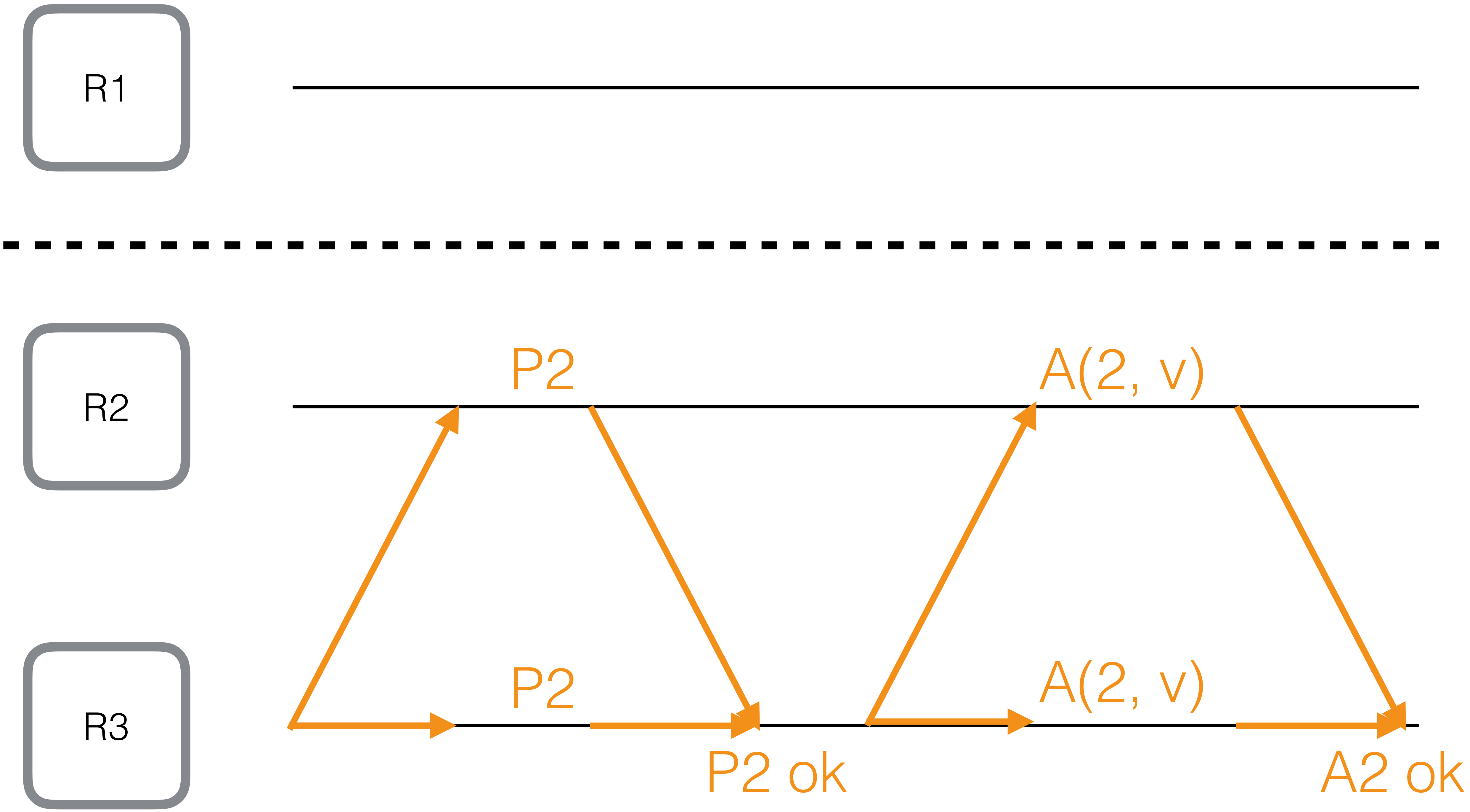
Example 3: Multiple proposers continued



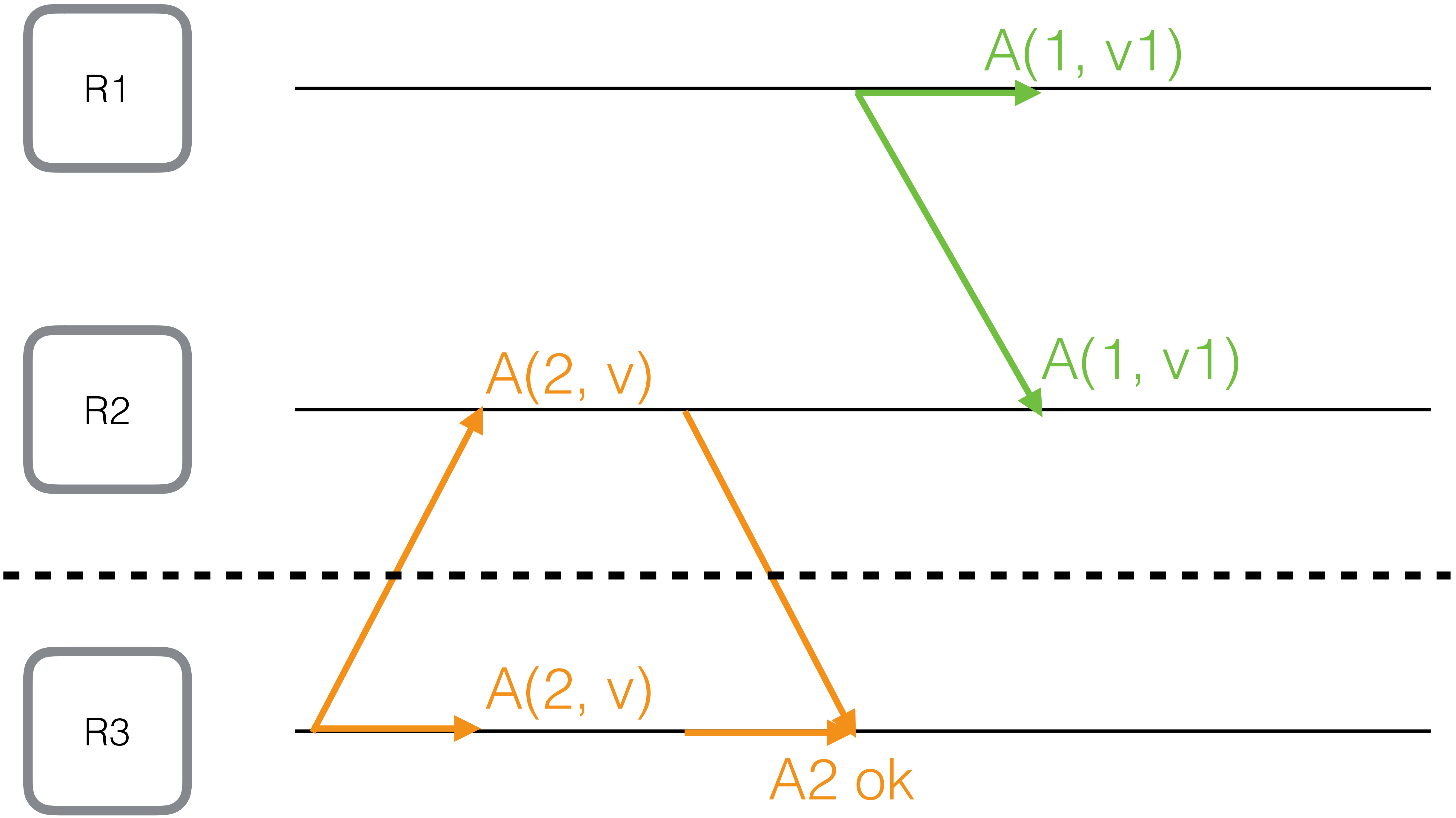
Example 3: Multiple proposers continued



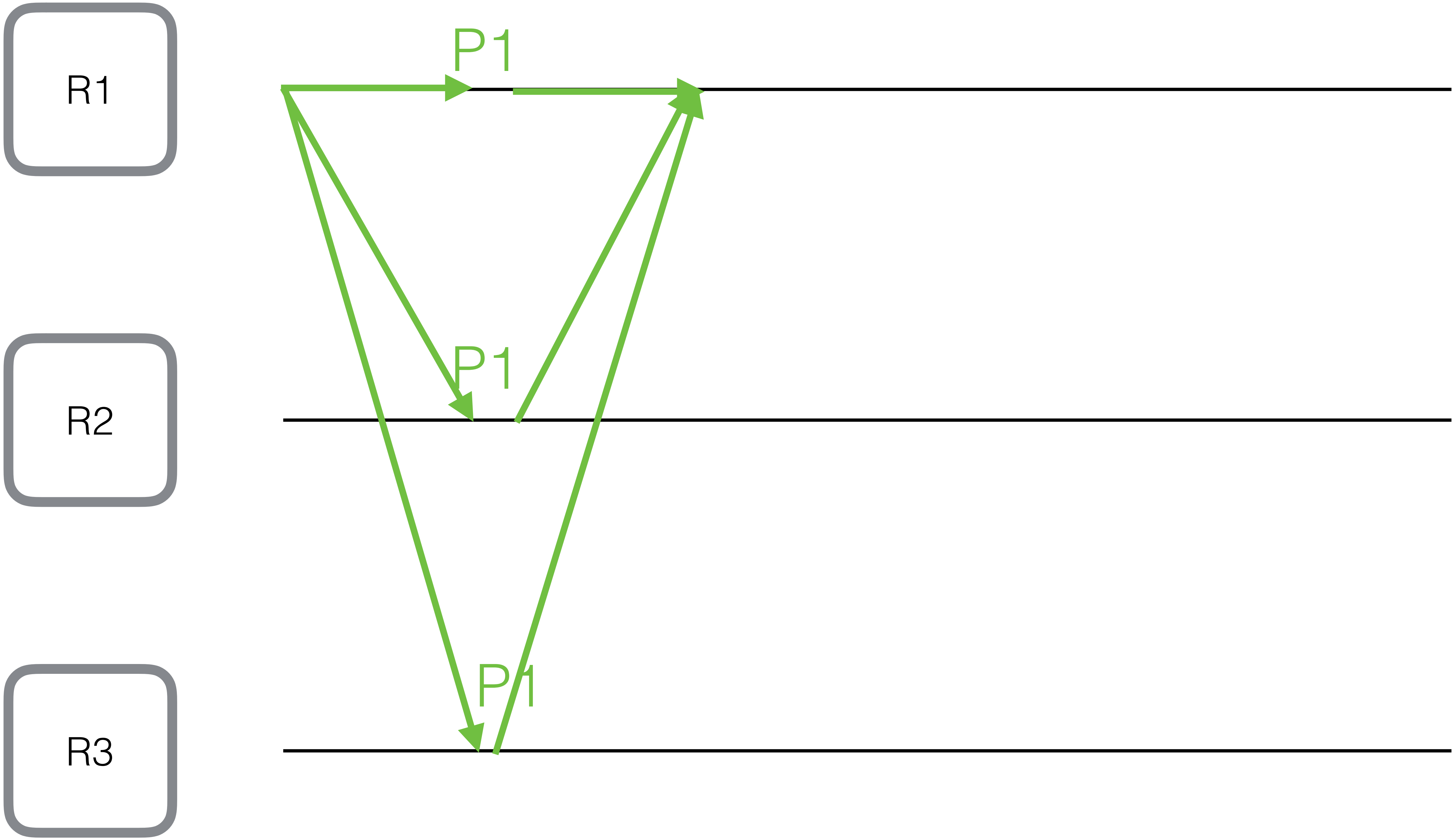
Example 3: Multiple proposers continued



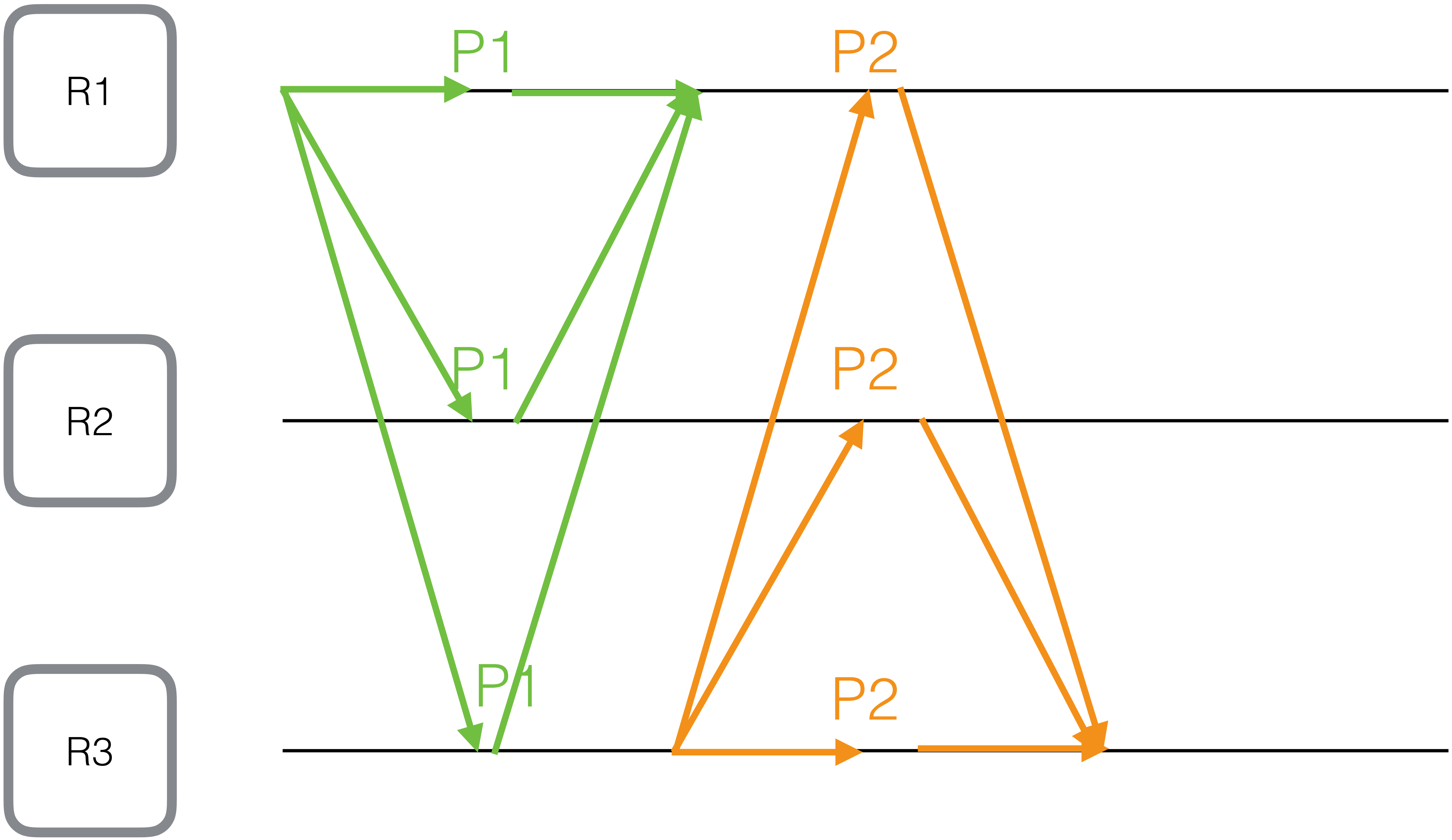
Example 3: Multiple proposers continued



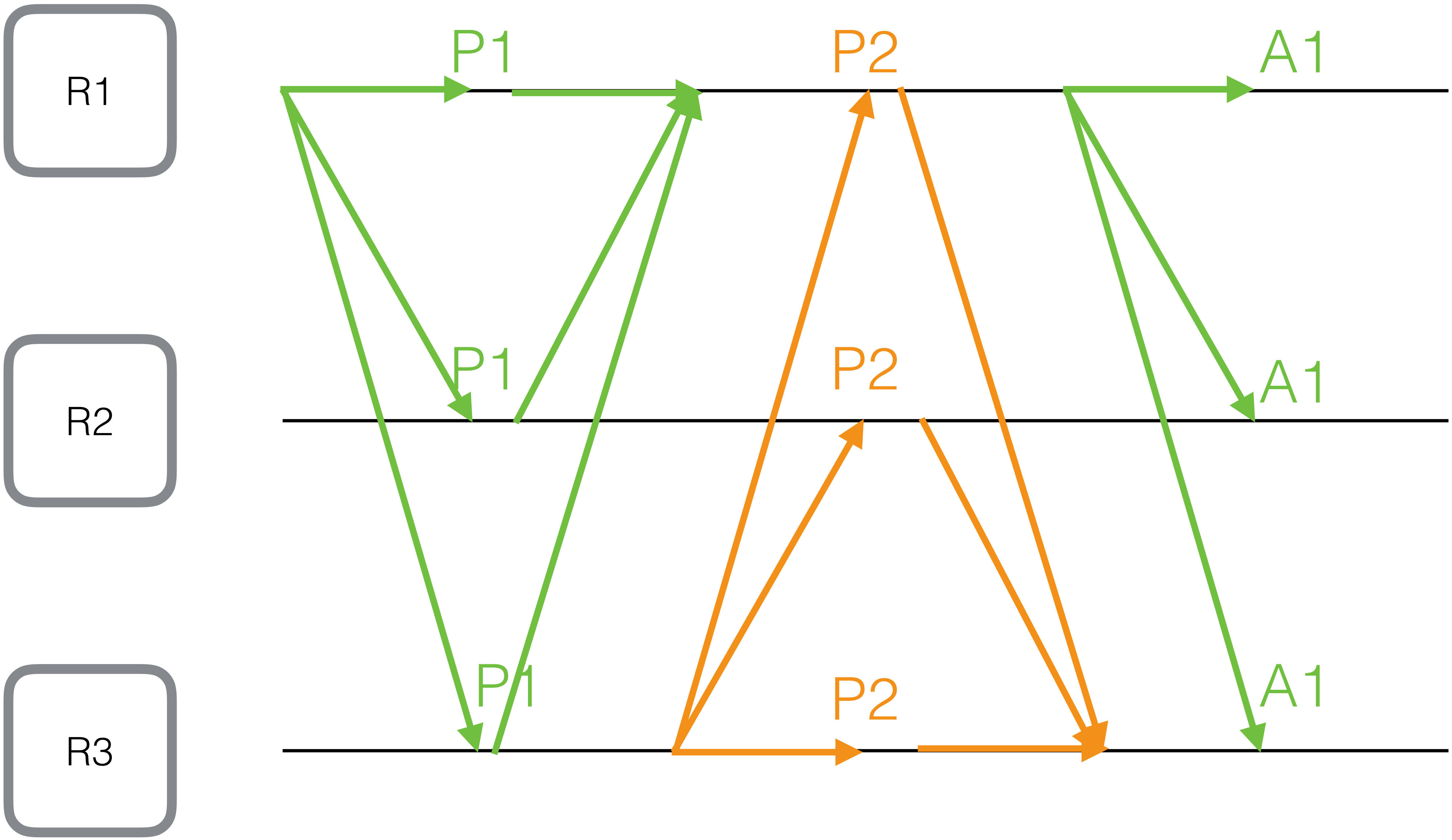
Example 4: Infinite proposals



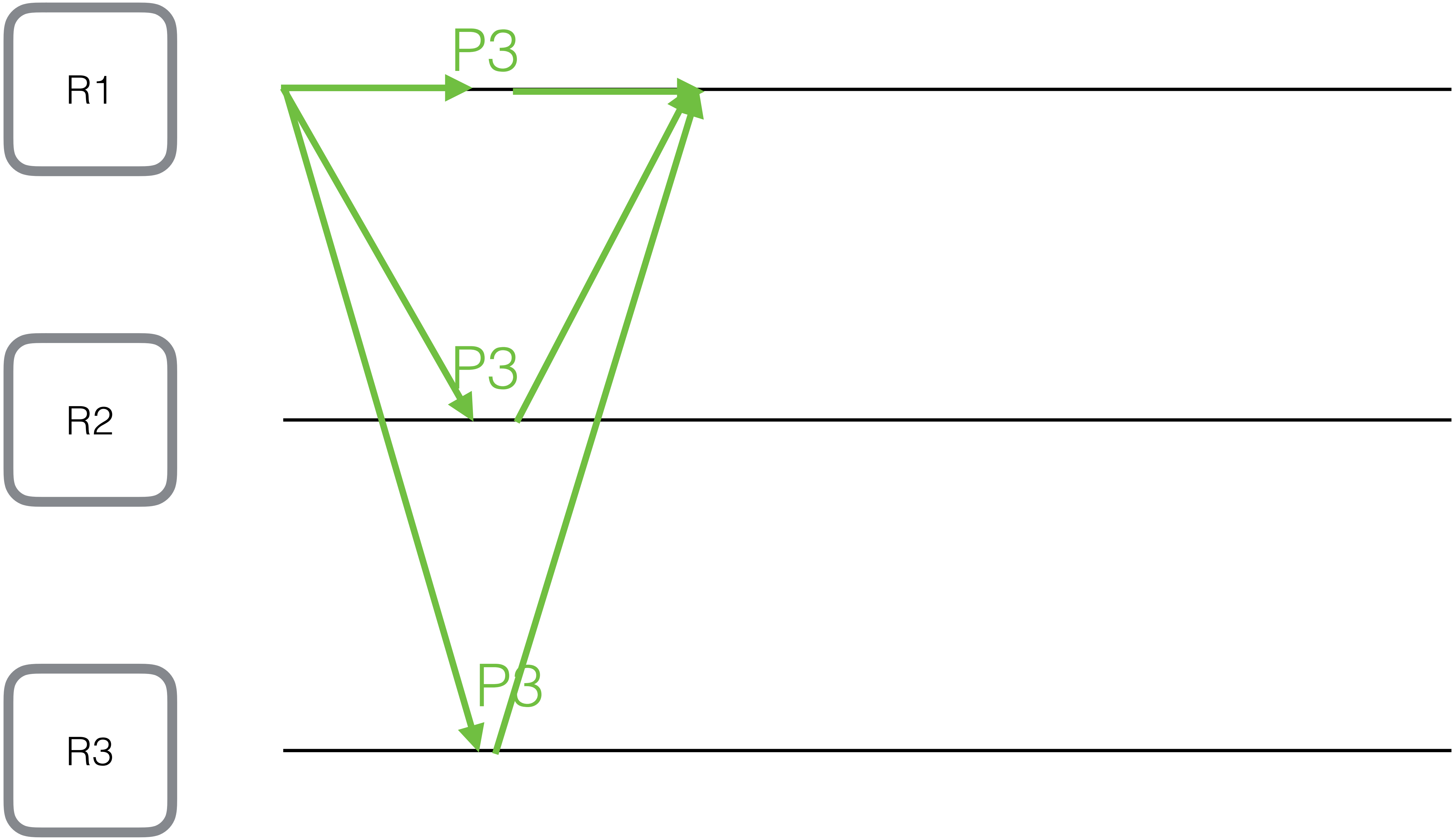
Example 4: Infinite proposals



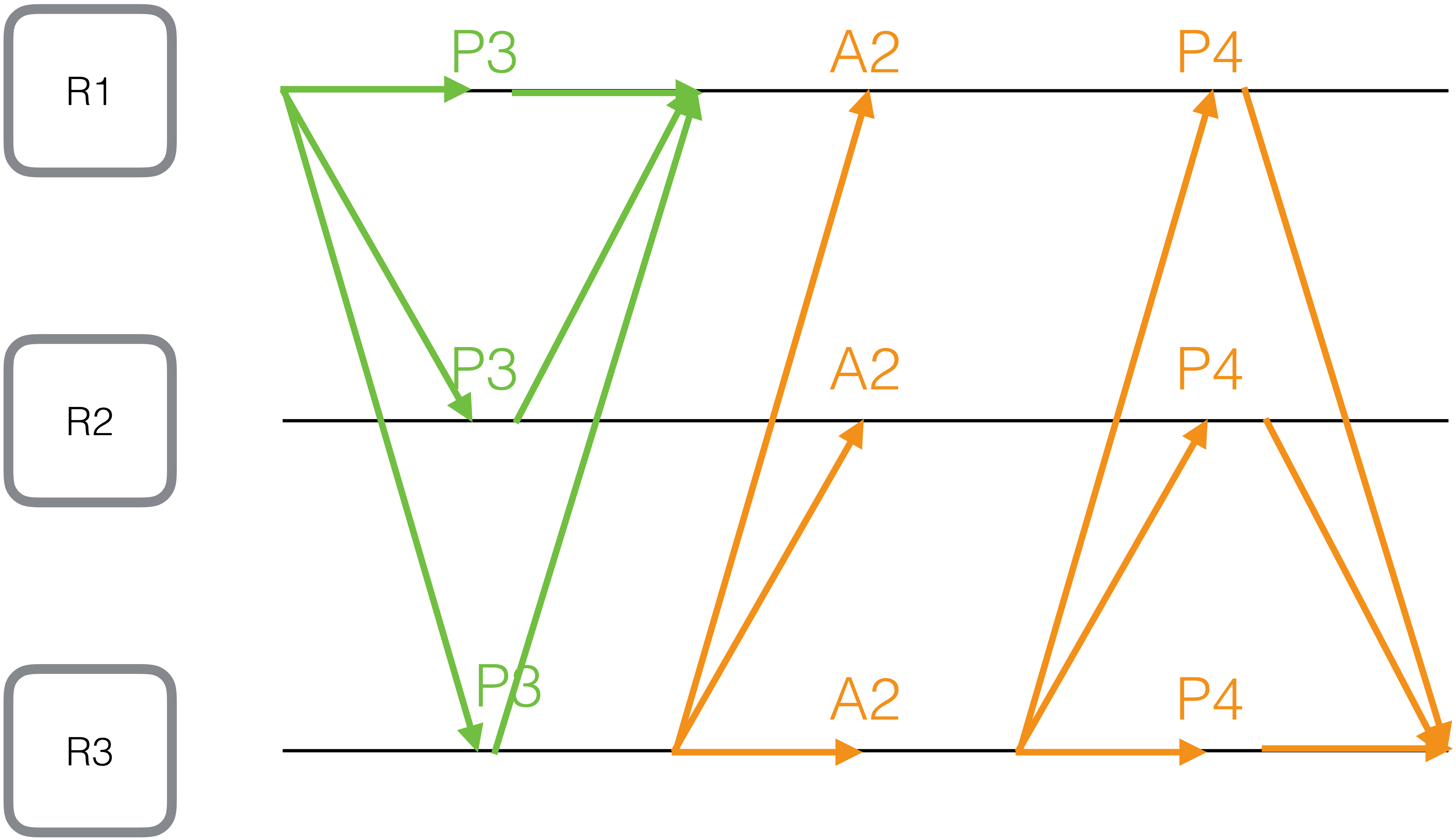
Example 4: Infinite proposals



Example 4: Infinite proposals



Example 4: Infinite proposals



Variations

- Multi-Paxos
    - Stable leader —> skip phase 1
    - Proposer —> Leader —> Acceptors —> Learner
- Fast Paxos
    - Proposer goes straight to acceptors
- Speculative Paxos, Egalitarian Paxos, …

Discussion

- A lot of variations of Paxos — where does the research lead?
    - Coordination cannot be better than 1 RTT
- Do you always need consensus? When do you need consensus?