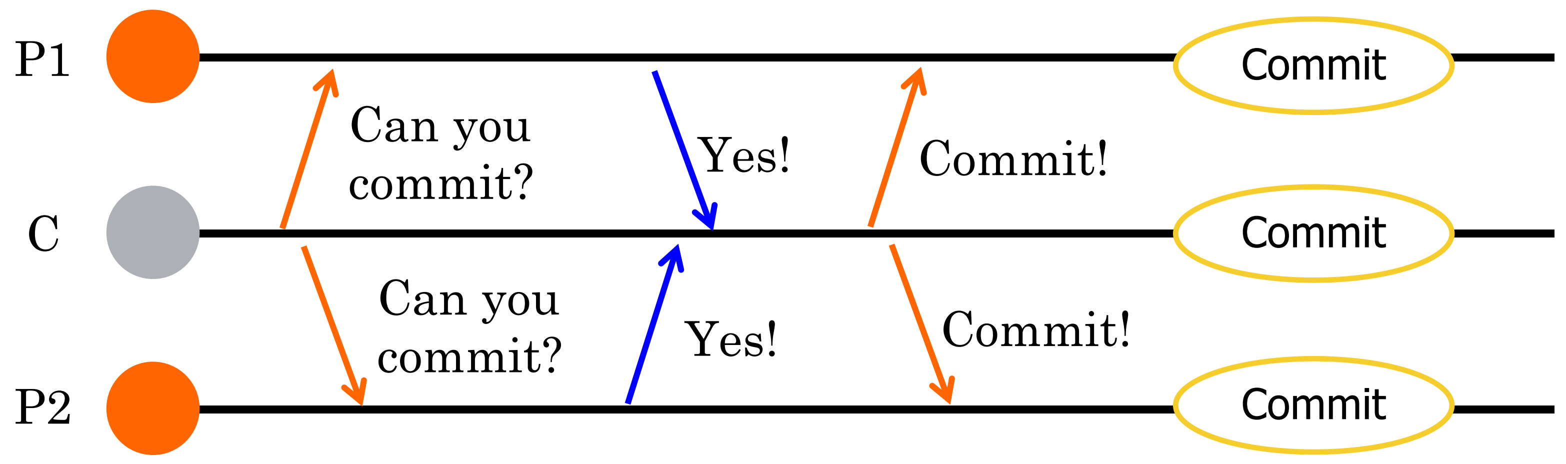
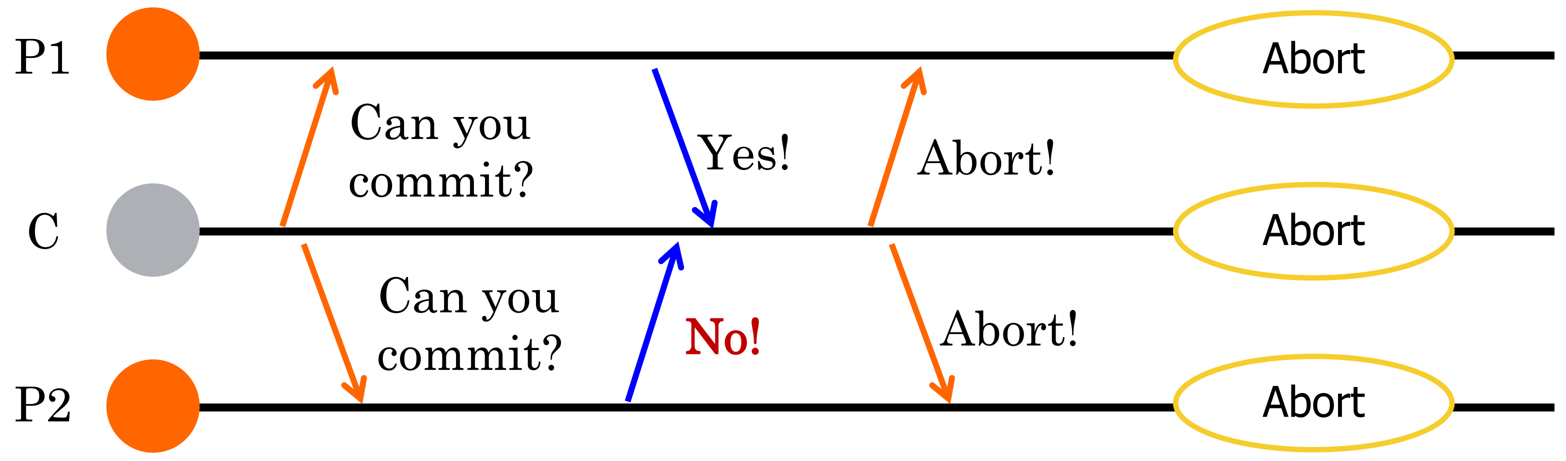
Distributed Systems 1: Lab 4, Two-Phase Commit

davide.vecchia@unitn.it

Two-phase commit (2PC)

- Coordinator C collects votes to decide if a transaction should be committed or aborted for all nodes (all-or-nothing atomic commit).





Exercise template

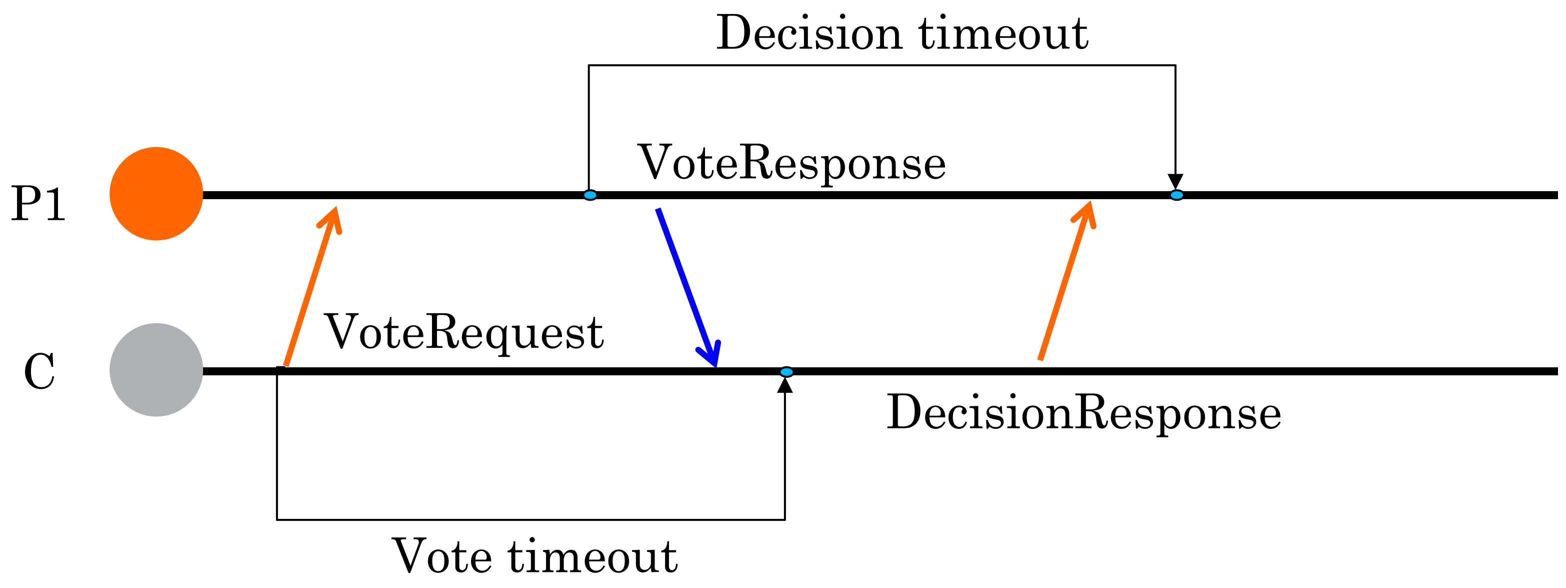
- `TwoPhaseCommit.java` defines Coordinator & Participant actors
  - Both inherit from an abstract actor Node with common functionality
- On start:
  - The coordinator and the participants get informed about the group
- Constants:
  - number of participants
  - the timeout durations
  - how the participants should vote (fixed yes/no)

Exercise template

- The program executes only one round of 2PC.
- The coordinator starts by multicasting a vote request. Participants reply with their predefined vote.

- The incomplete implementation will not work in case of crashes and delays!

Messages and timeouts



- What should coordinator C do upon vote timeout?

  Broadcast ABORT decision.

- What should a participant do upon decision timeout?

  If it voted no, **nothing (it already decided ABORT).**
  If it voted yes, **ask other participants for the decision.**

Akka messages

- **VoteRequest**
- **VoteResponse**
- **DecisionRequest** – only on timeout
- **DecisionResponse**

- Messages sent to self
  - **Timeout**
  - **Recovery**

Useful methods

- **multicast(m)** – send a message to all the participants
    - except for self and the coordinator
- **setTimeout(t)** – send a timeout message to self in specified time (ms)
- **fixDecision(d)** – accept the final decision
- **hasDecided()** – tells whether the current node has accepted the decision or not yet
- **allVotedYes()** – tells whether ”yes” replies were received from all participants (coordinator only)
- **print(string)** – simple debug logging

Simulating faults

- **crash(time)** – simulate a crash and recovery after the specified time interval
- **multicastAndCrash(m, time)** – crash in the middle of the multicast and later recover
- **delay(time)** – sleep for the specified time

```
// simulate a crash of node 2
if (id==2) {crash(5000); return;}

// simulate a performance failure of node 2
if (id==2) delay(4000);
```