Foundations of Distributed Systems

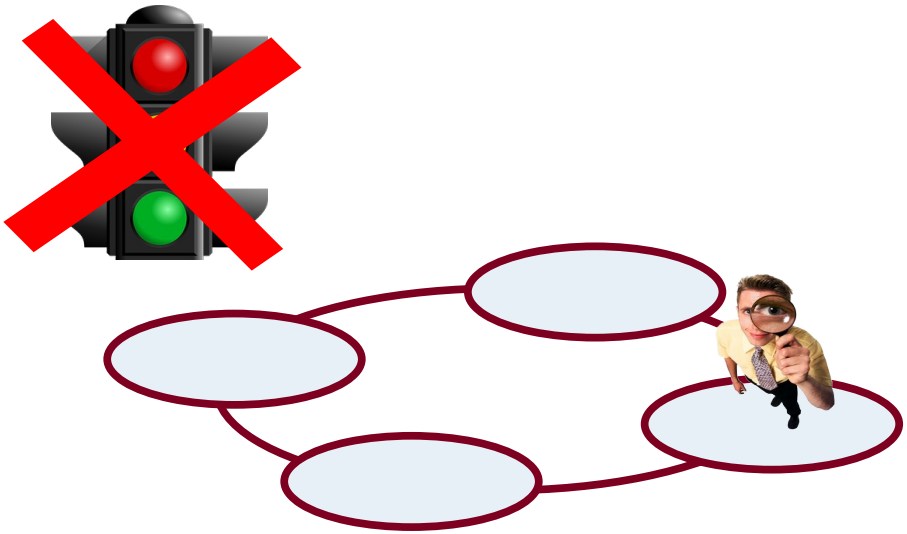
José Orlando Pereira

HASLab / Departamento de Informática
Universidade do Minho

2020/2021

Goals

- No global arbitration
- How to change the system from the inside?

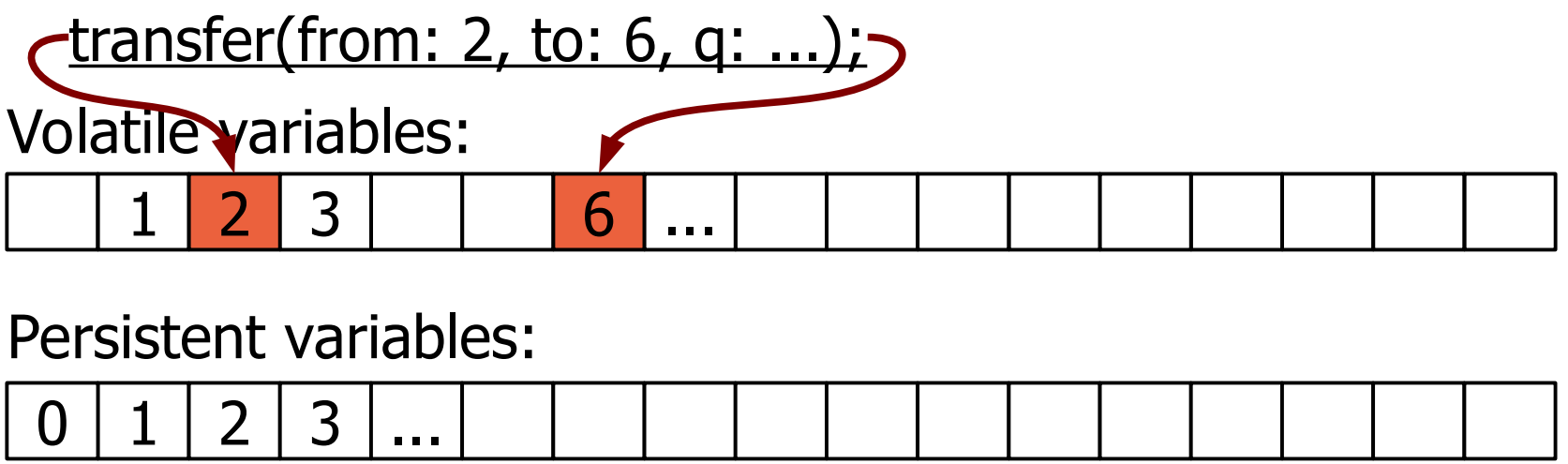


Crash and Recovery

- Asynchronous system model
- Crash and recovery events (a.k.a reboot)
  - Assume that there is always recovery
- Two types of variables:
  - Volatile, lost on crash
  - Persistent, kept after recovery

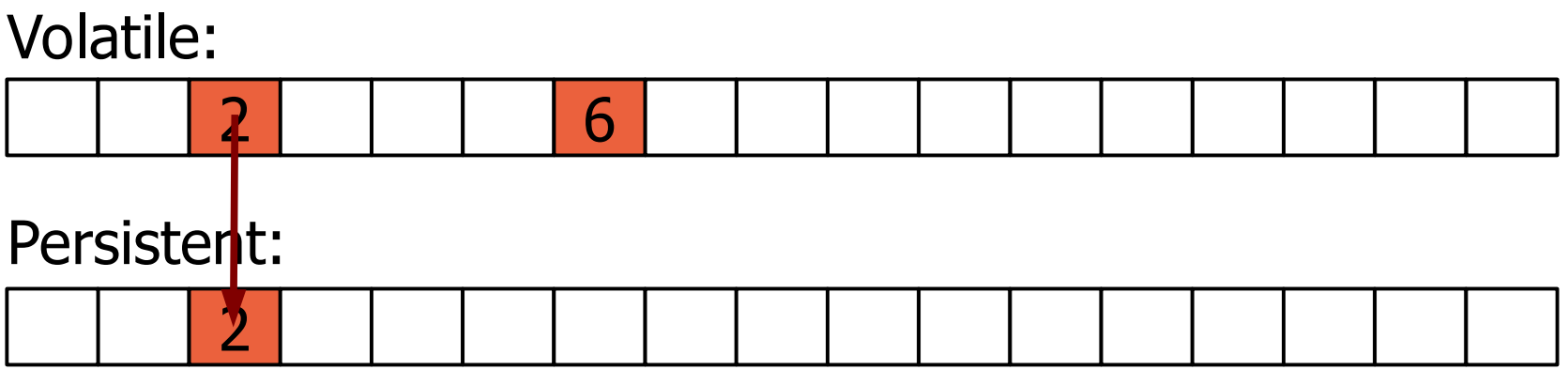
Operations

- The application changes volatile variables:



Operations

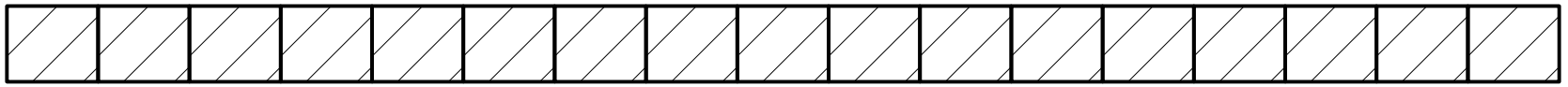
- Changes can now be copied to persistent variables:



Failure and restart

- Restart erases volatile variables:

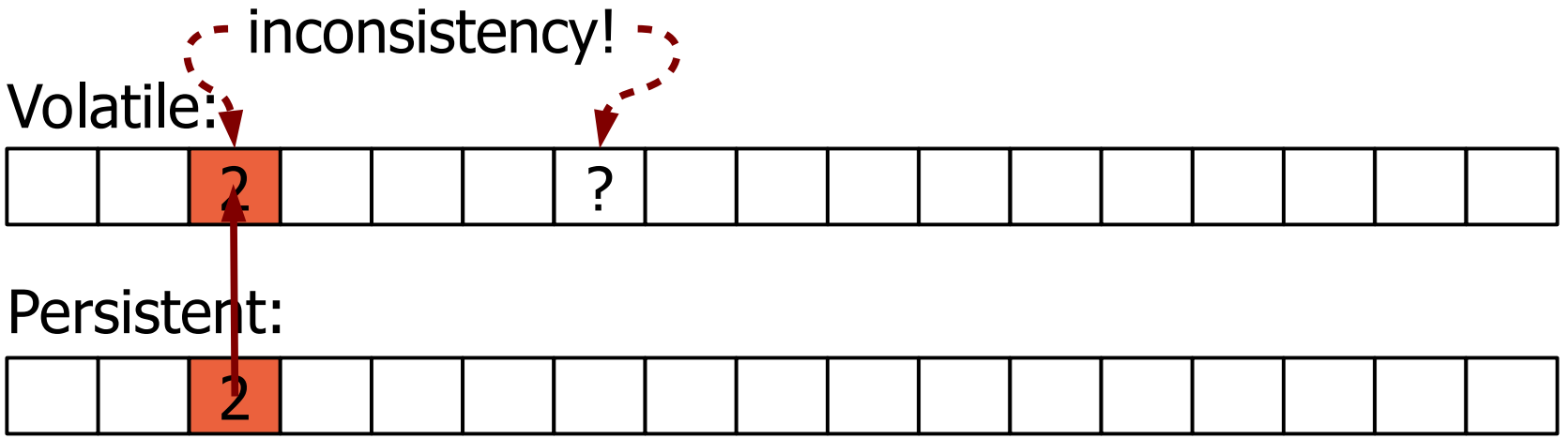
Volatile:



Persistent:

Failure and restart

- Volatile variables are restored from the persistent copy:



Challenges

- Single process failure and recovery
- Partial system failure and recovery

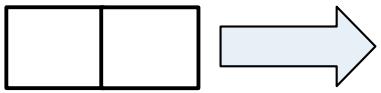
Redo log

- Assume a persistent sequential log:

Volatile variables:

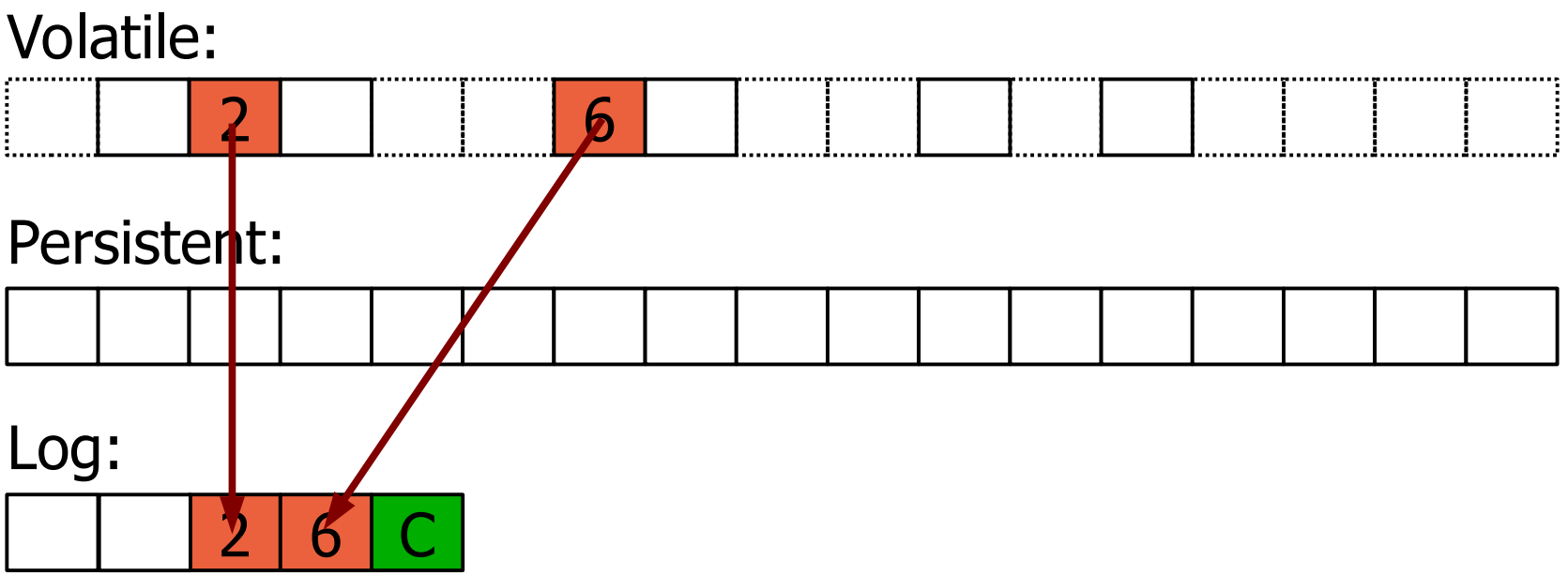
Persistent variables:

Sequential log:



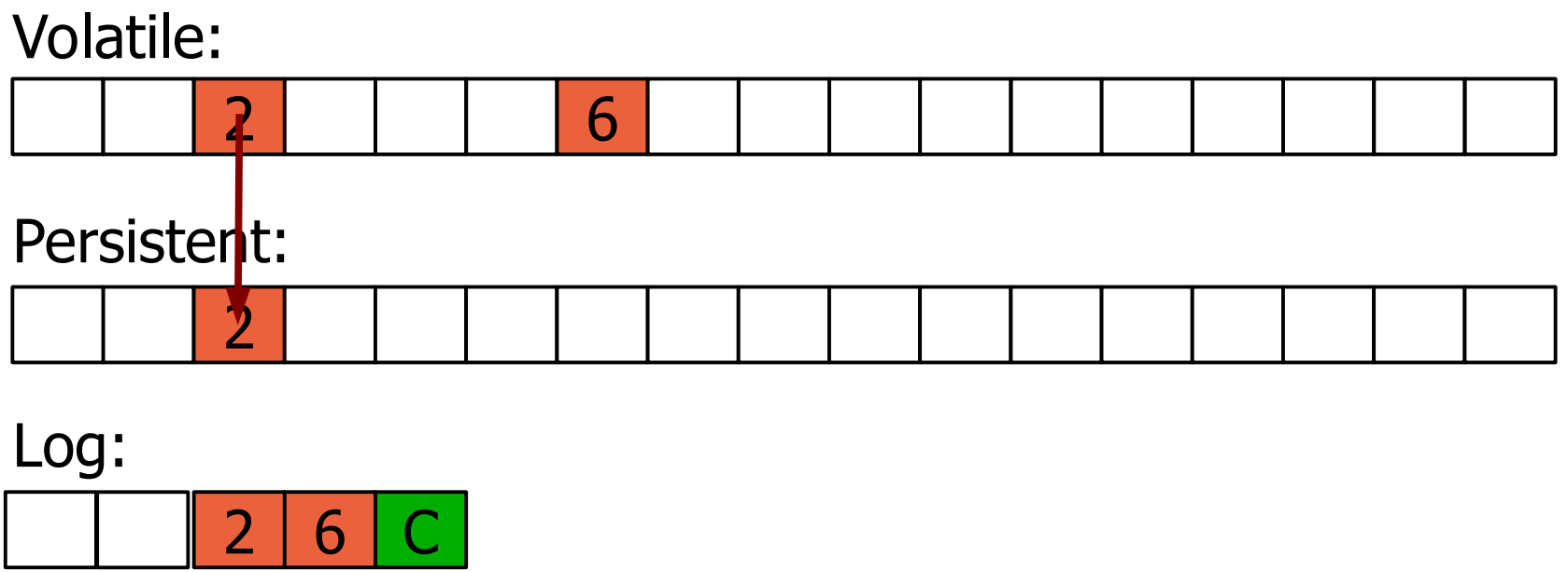
Redo log

- Changed variables are copied to the log followed by a commit marker:



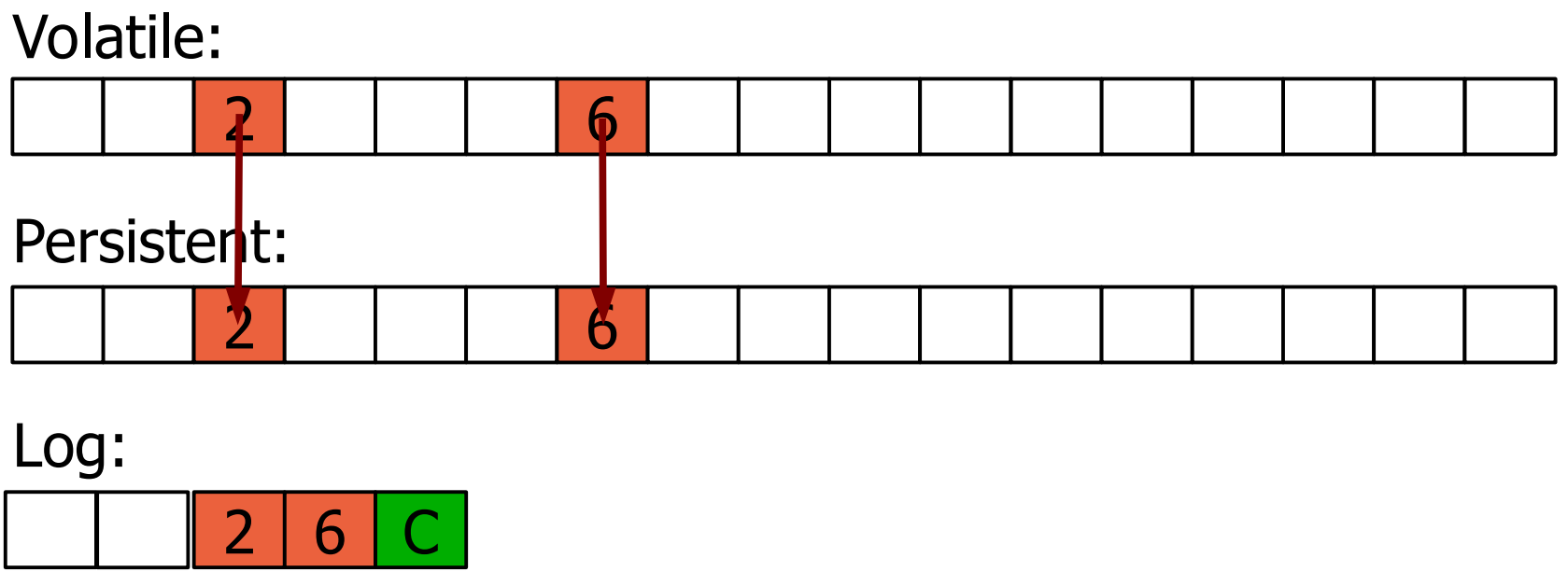
Redo log

- Changes can now be copied to persistent variables:



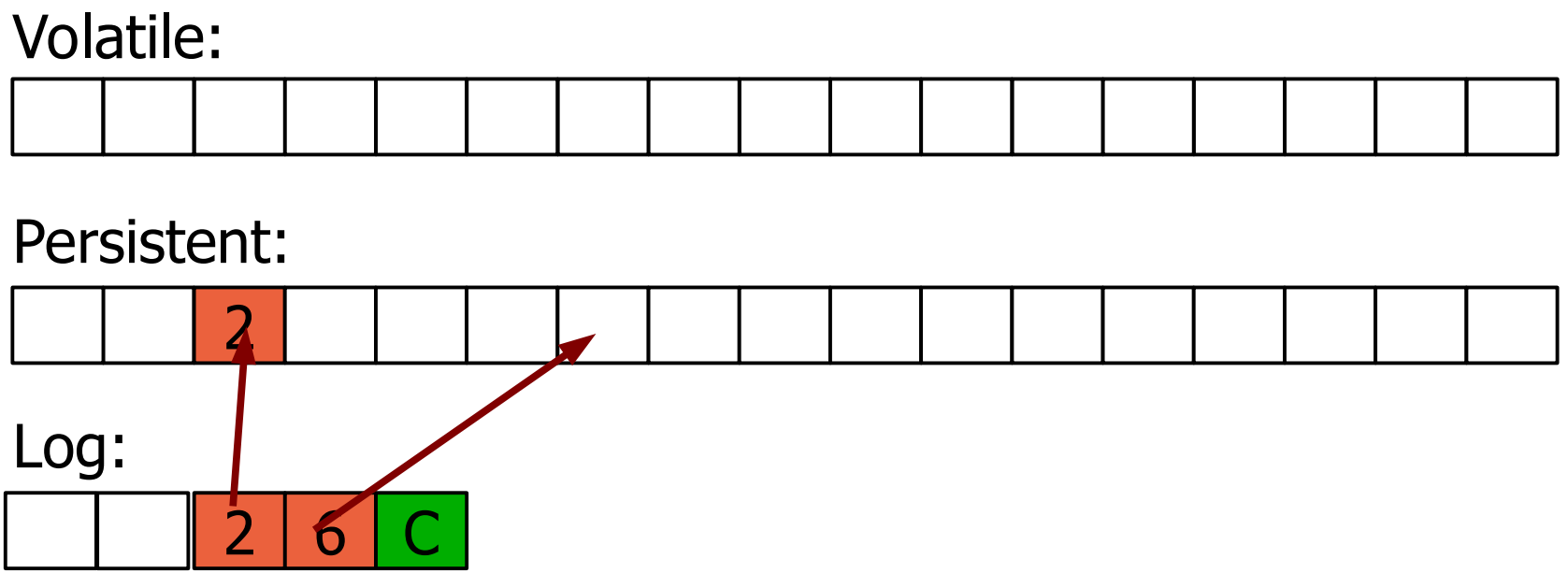
Redo log

- Eventually all persistent variables are updated:



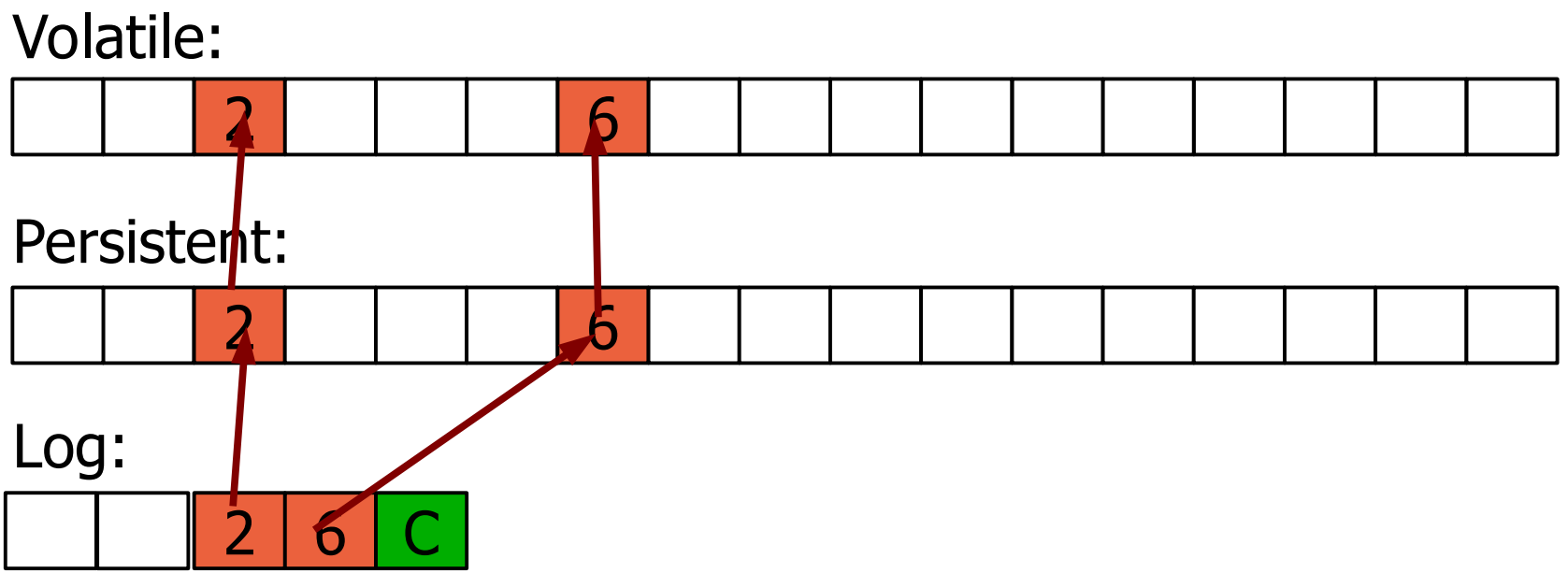
Redo log

- In case of restart, upon recovery, redo all changes from the log:



Redo log

- And restore volatile variables to become operational again:

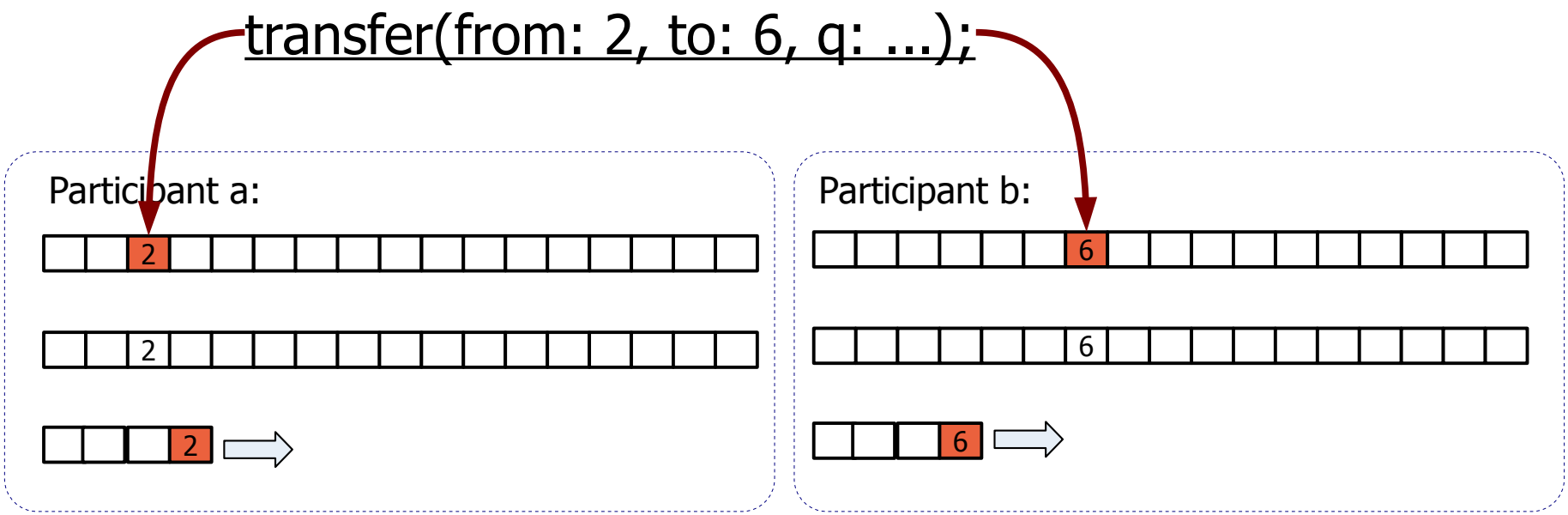


Challenges

- ~~Single process failure and recovery~~
- Partial system failure and recovery

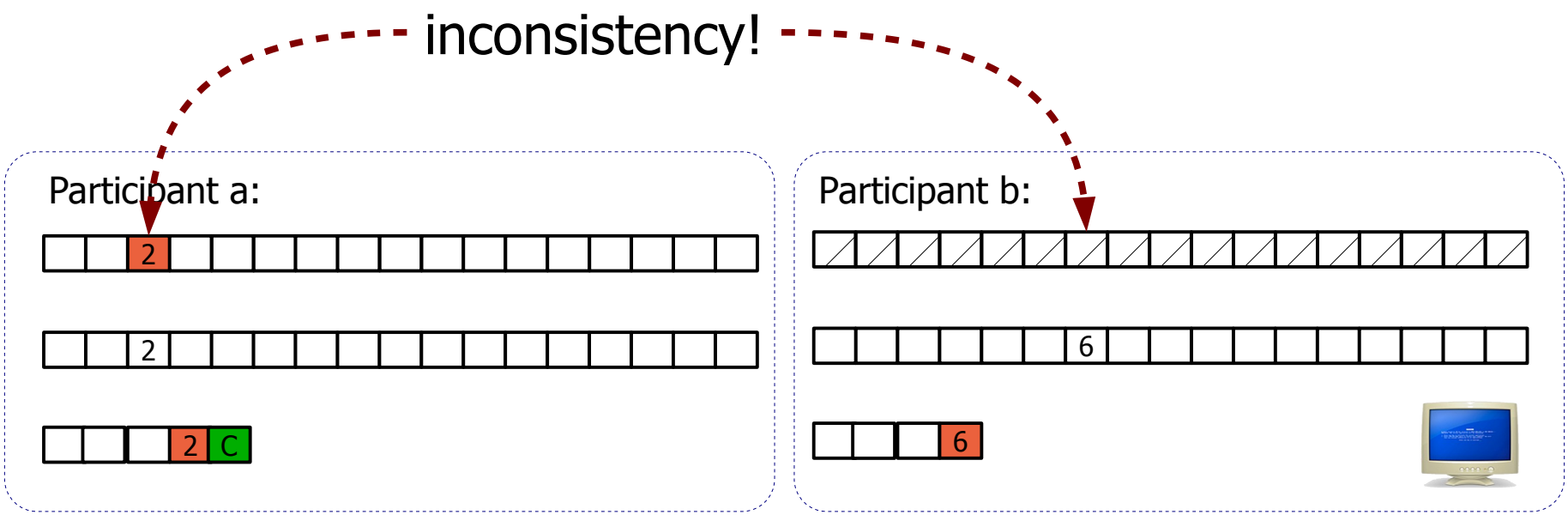
Distributed operations

- Distributed operation updates variables in multiple processes:



Partial failure and restart

- If each participant commits independently, then...

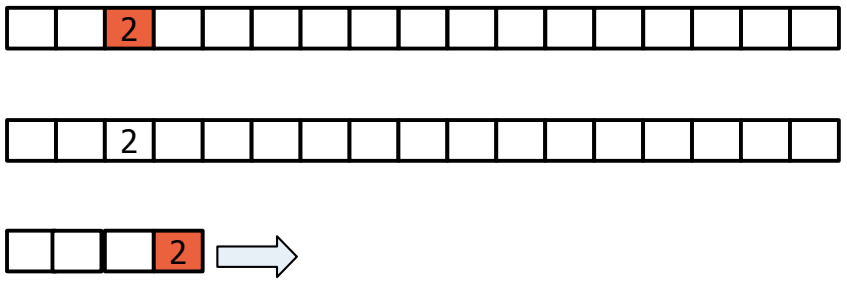


Two Phase Commit (2PC)

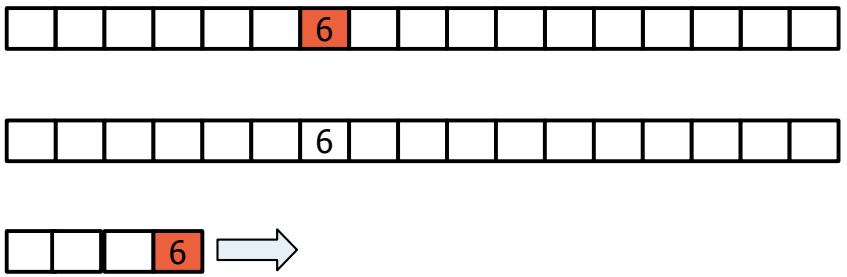
- Add a coordinator with an additional persistent log:

Coordinator log:

Participant a:

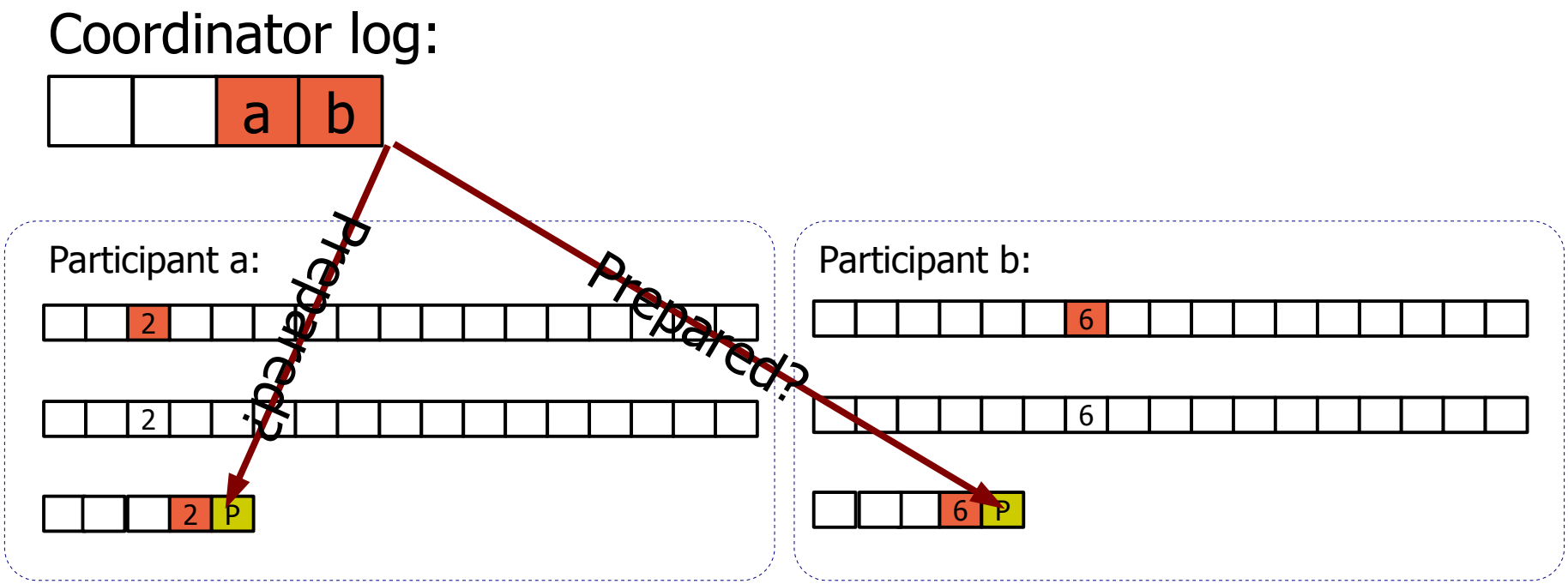


Participant b:



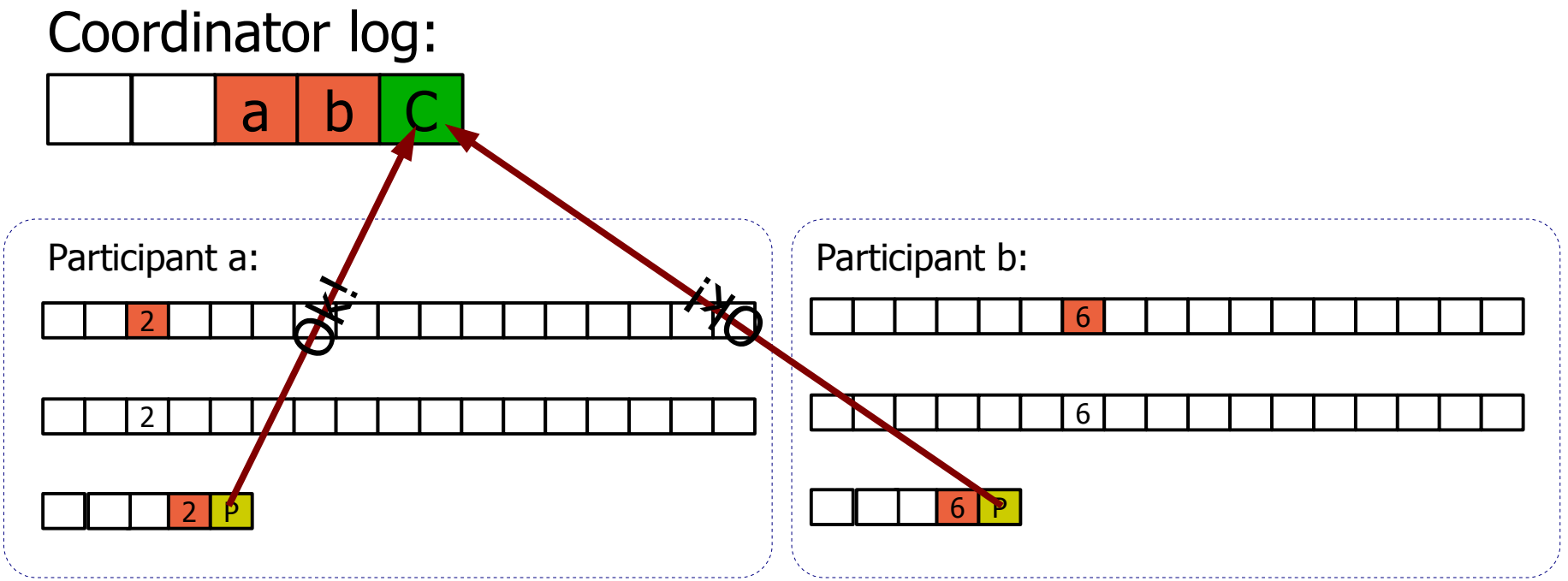
2PC: Phase 1

- The coordinator asks all participants to prepare, inserting prepared markers in logs:



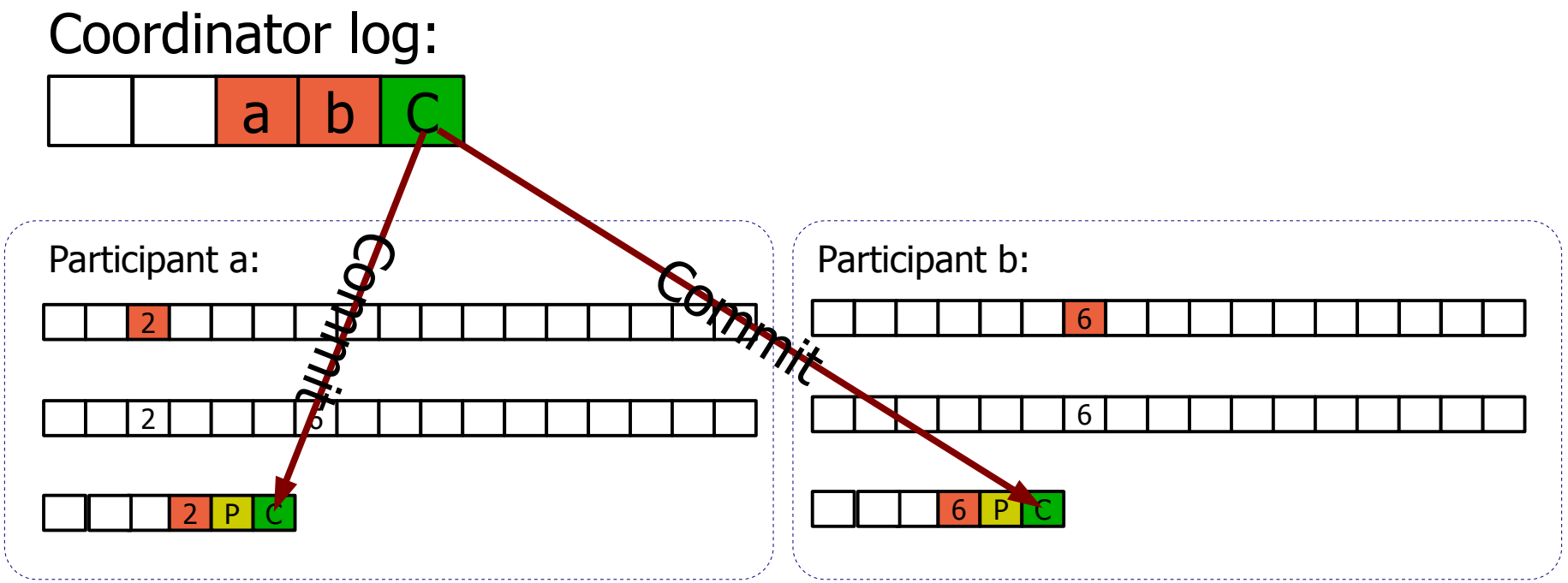
2PC: Phase 1 Completed

- If all participants are able to prepare, the coordinator inserts a commit marker:



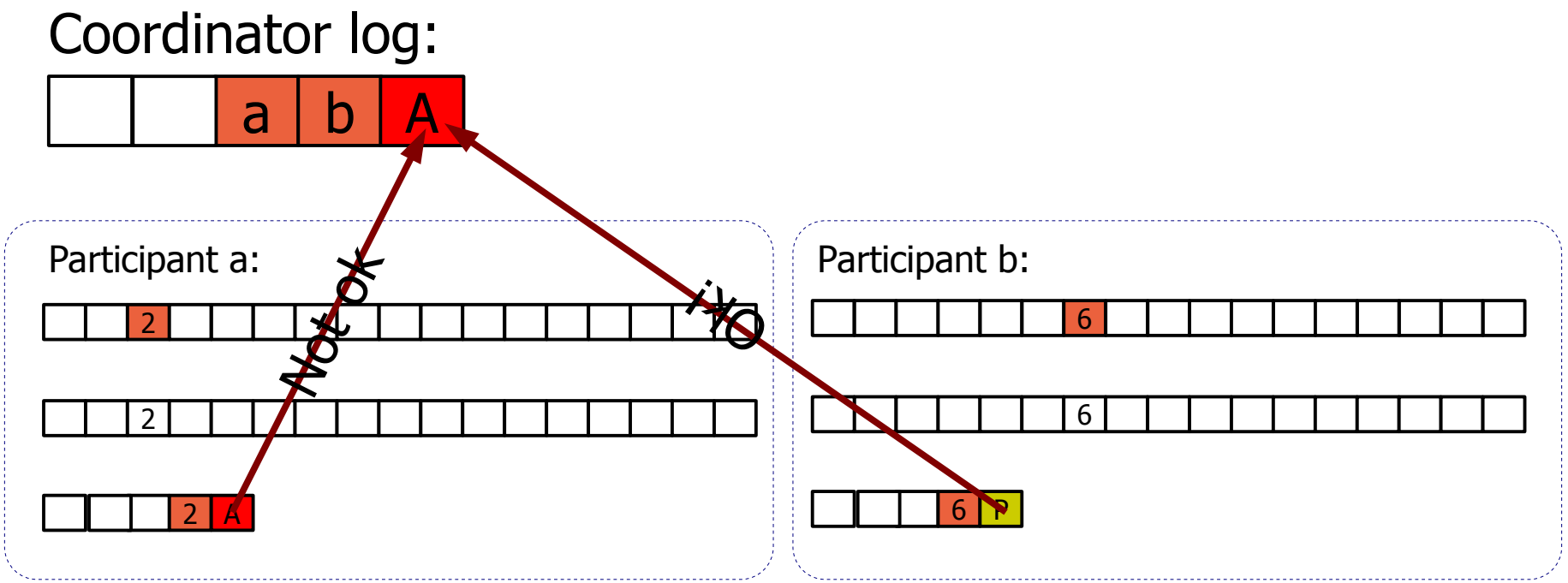
2PC: Phase 2

- Participants are informed of outcome inserting commit markers in their logs:



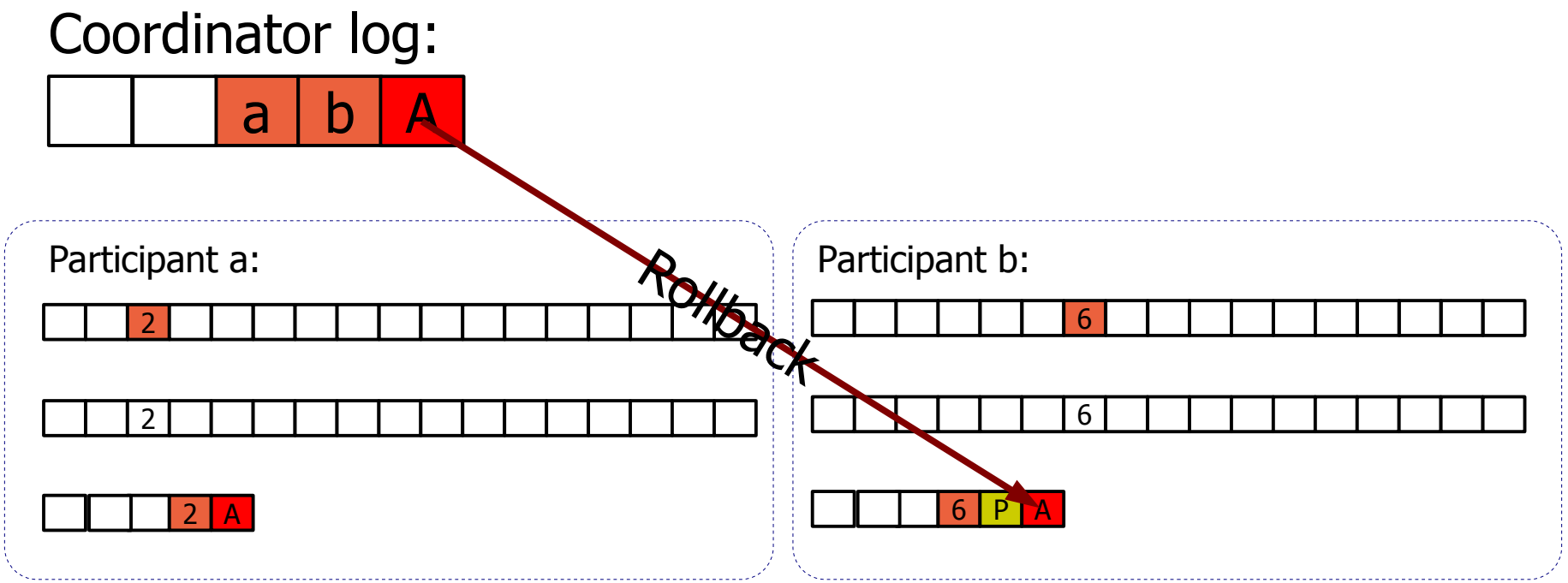
2PC: Phase 1 Aborted

- If a single participant calls for abort or fails the entire transaction is marked for rollback:



2PC: Phase 2 Aborted

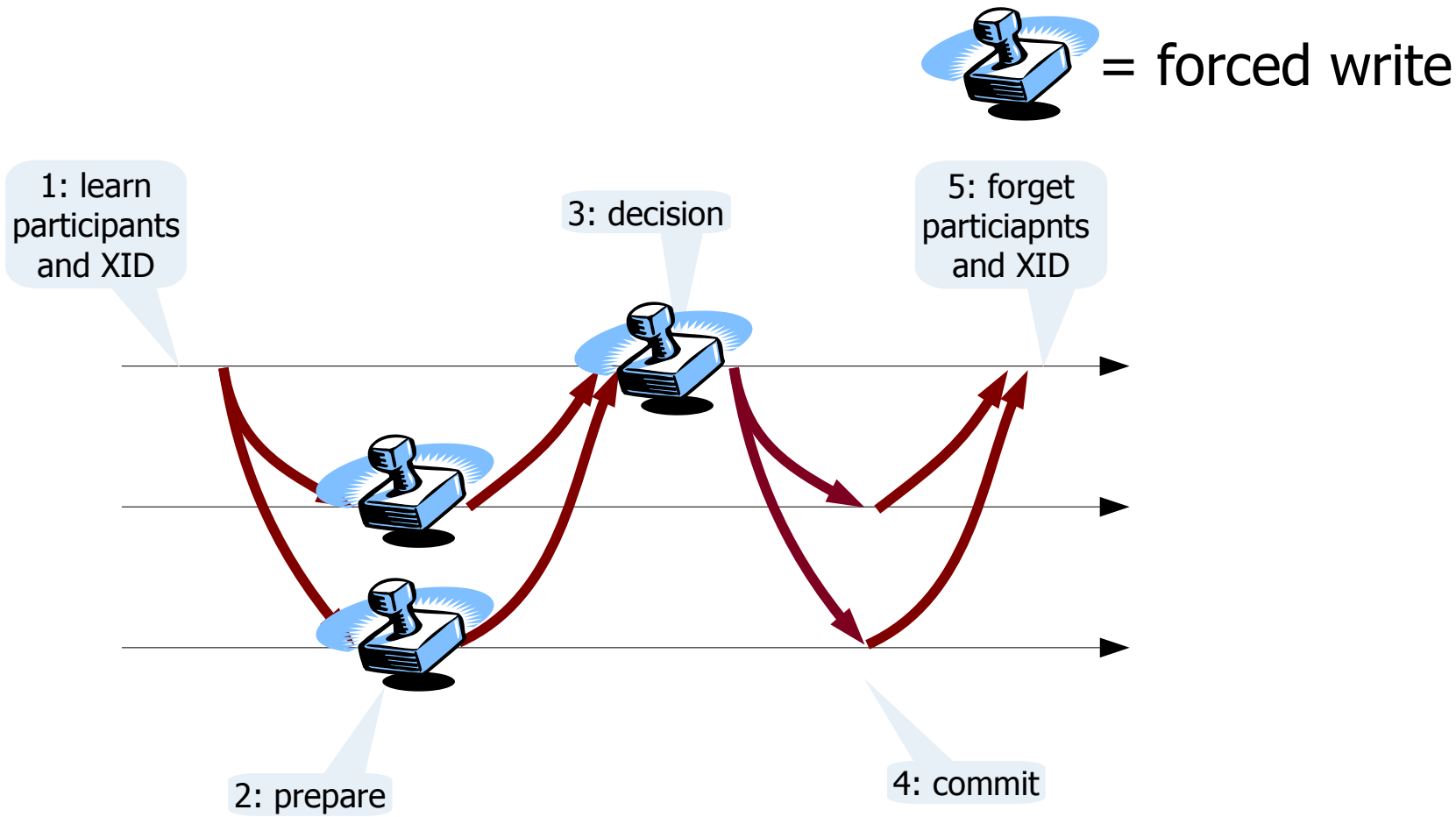
- If a single participant calls for abort or fails the entire transaction is marked for rollback:



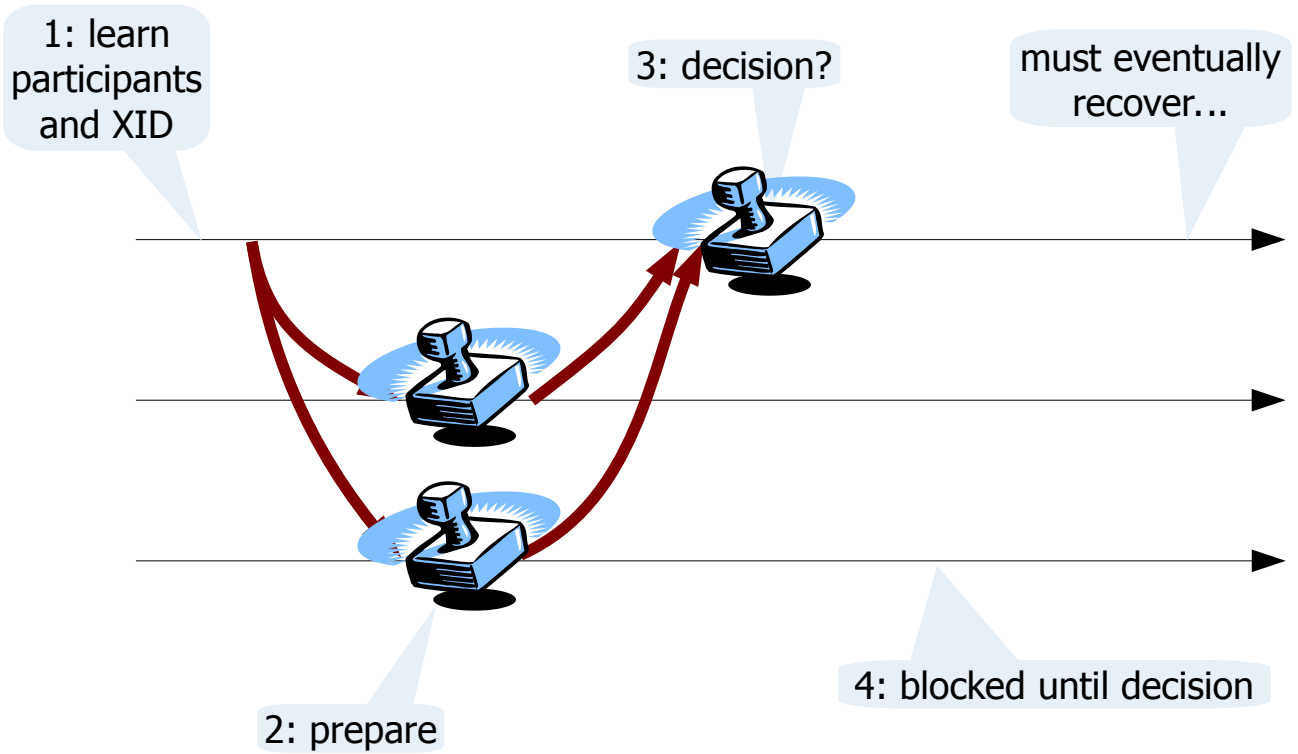
2PC: Recovery

- Restart of the coordinator:
  - Before 2PC started: Abort, as it might have missed a resource
  - During 2PC: Restart current phase by repeating the request
- Restart of a participant:
  - Has not voted: Local rollback, will abort the entire transaction
  - Has voted: Wait for the decision from the coordinator

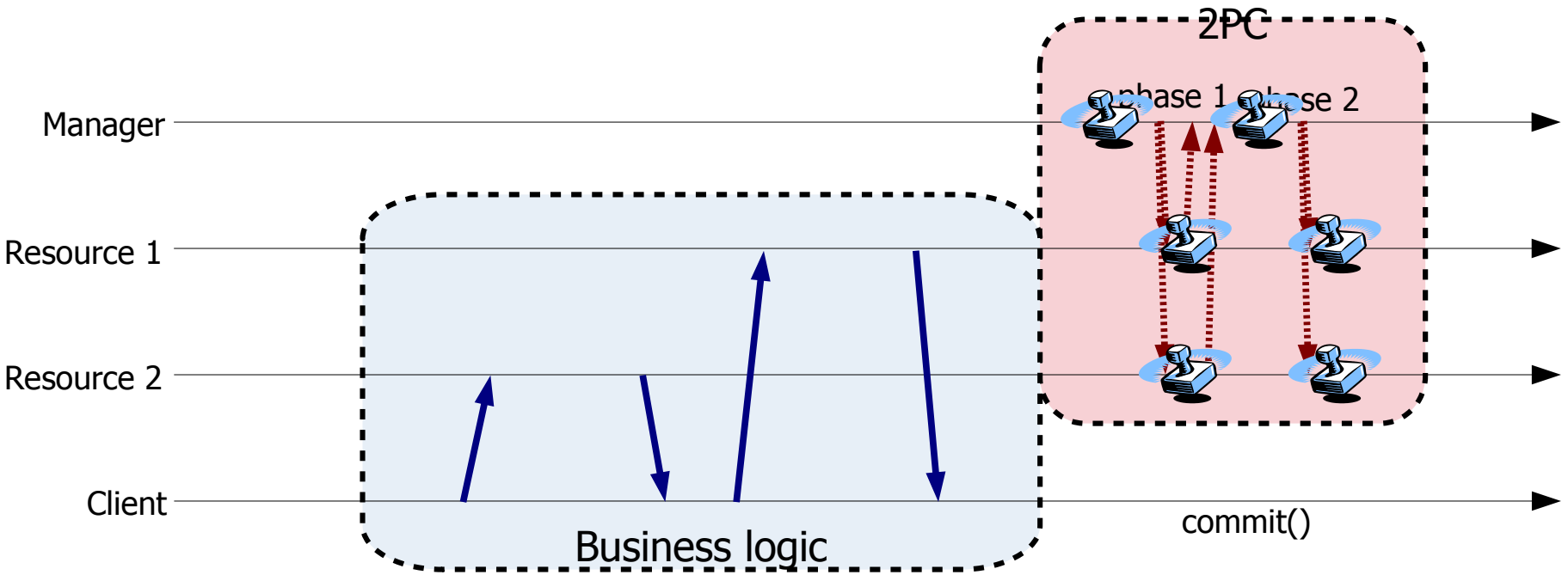
Summary



2PC: Blocking

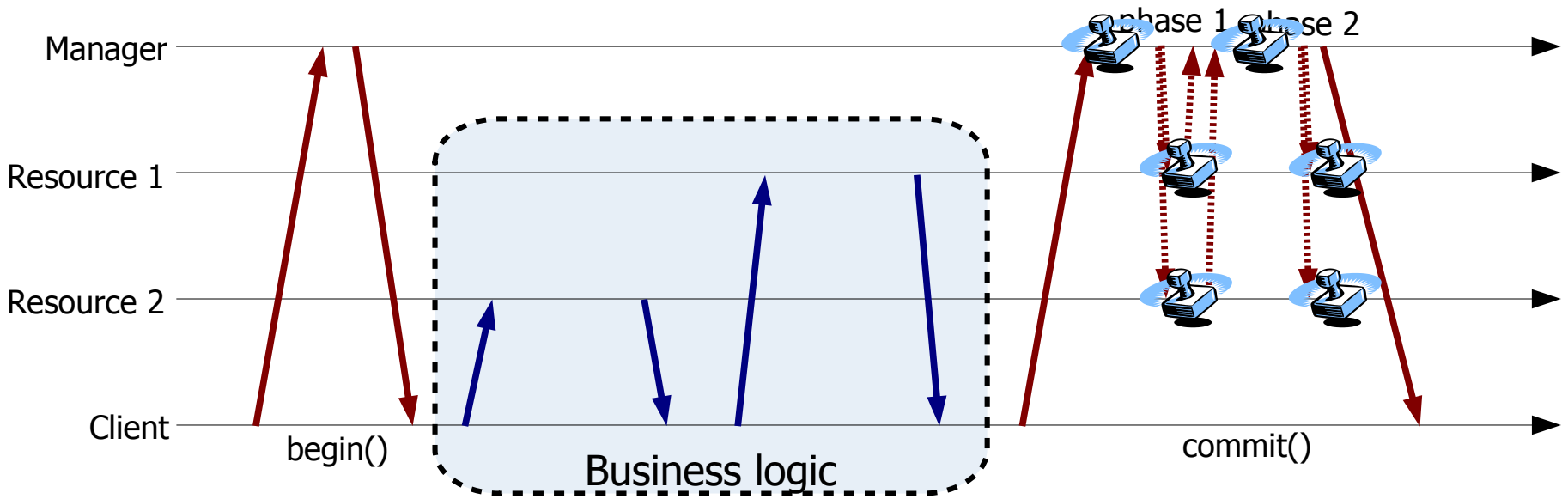


Application vs Transactions



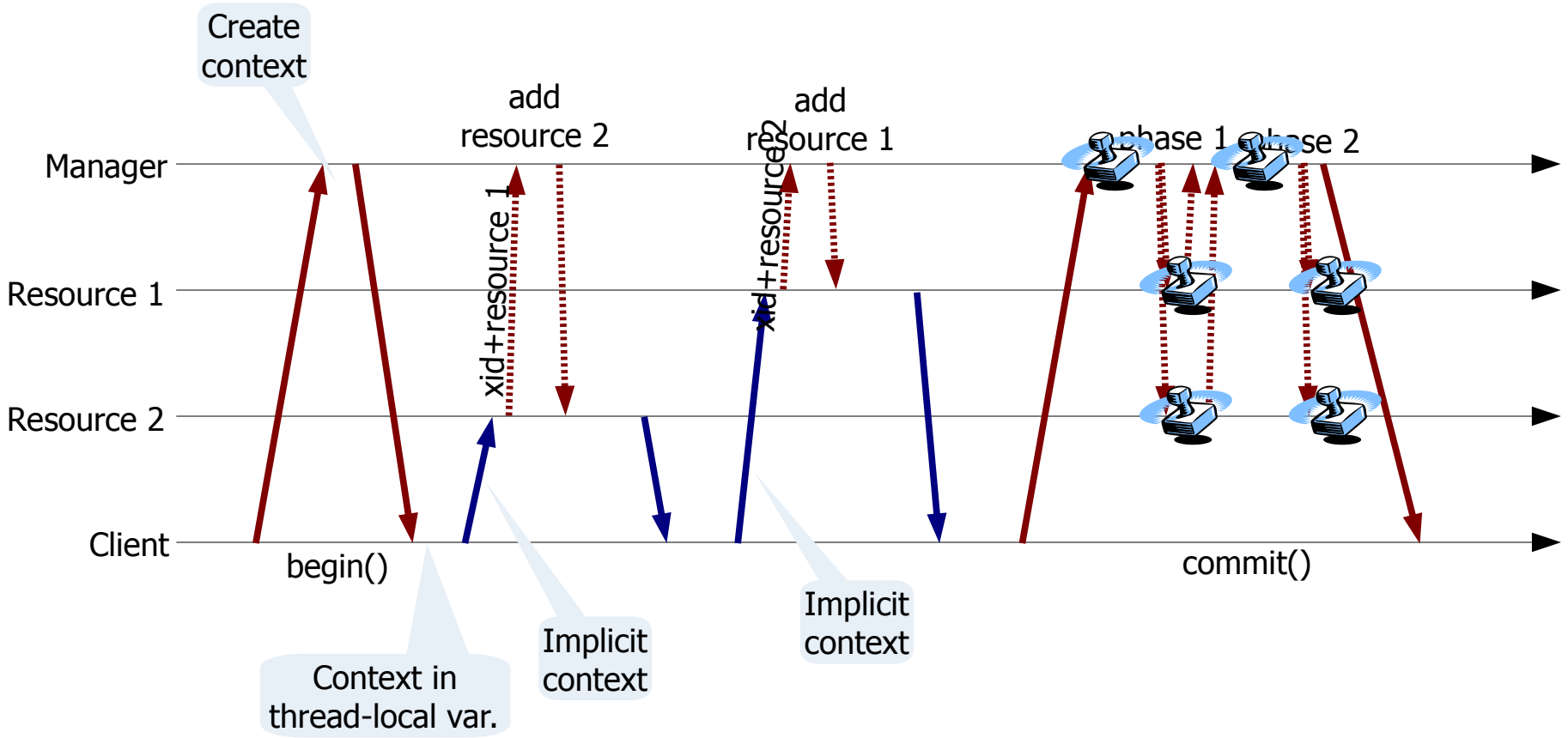
- How to connect them?
  - Don’t change business logic

Transaction demarcation

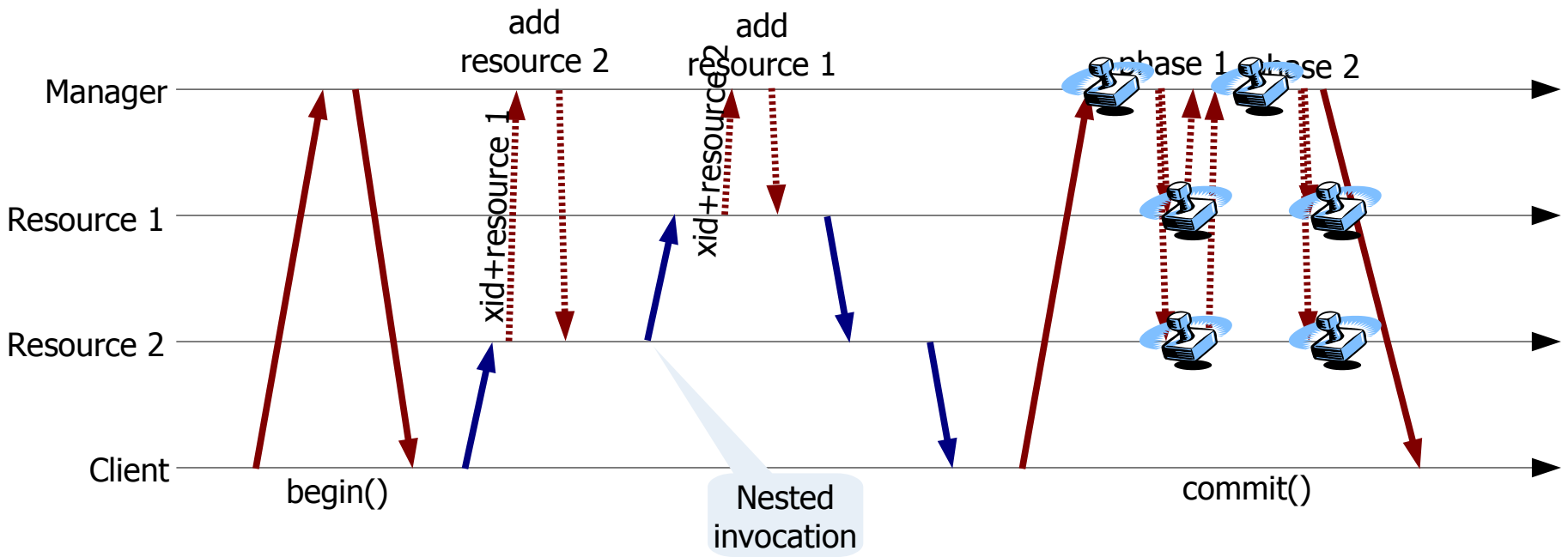


- Call transaction manager before/after
- How does manager discover participants?

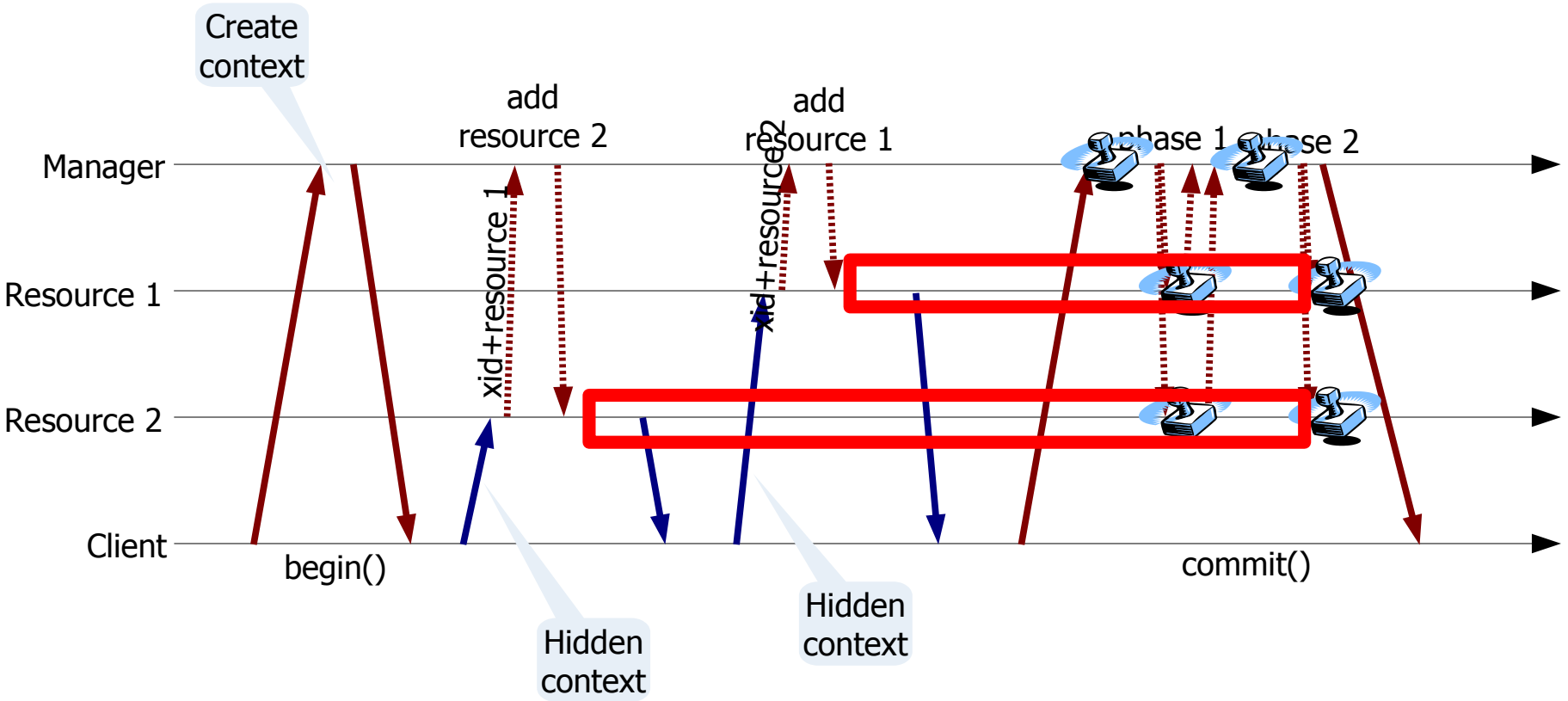
Transactional RPC + 2PC



Transactional RPC + 2PC



Transactional RPC + 2PC + 2PL



Summary

- Atomicity with faults: 2PC
- Atomicity with concurrent clients: locking
- With locking + 2PC:
  - Rollback on deadlock / client failure
  - Implicitly release locks on commit / rollback