Sistemas Distribuídos

José Orlando Pereira

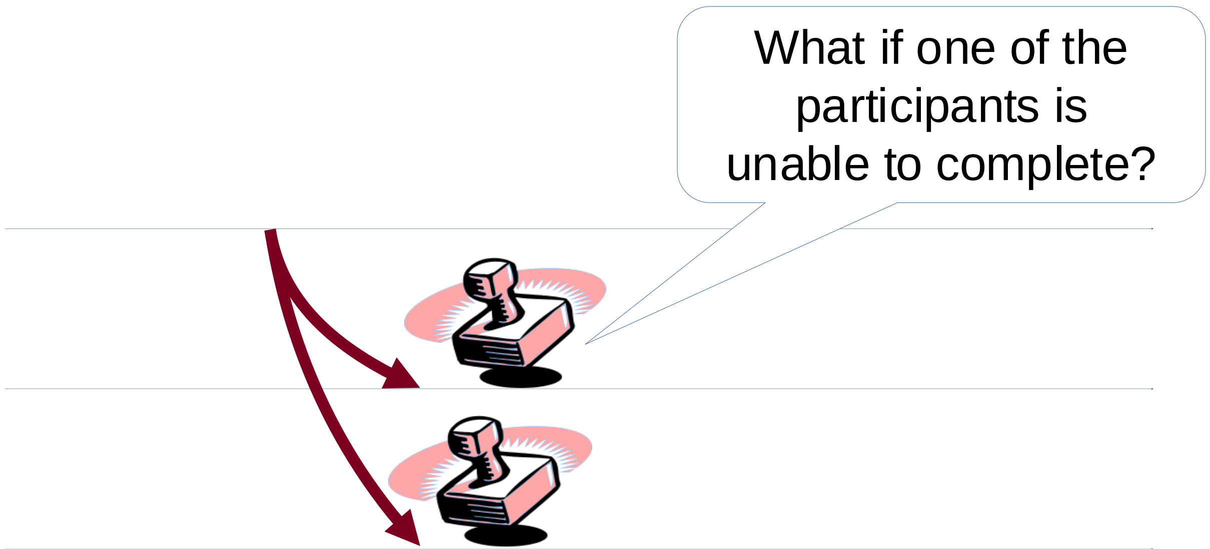
Departamento de Informática
Universidade do Minho

Fault tolerance

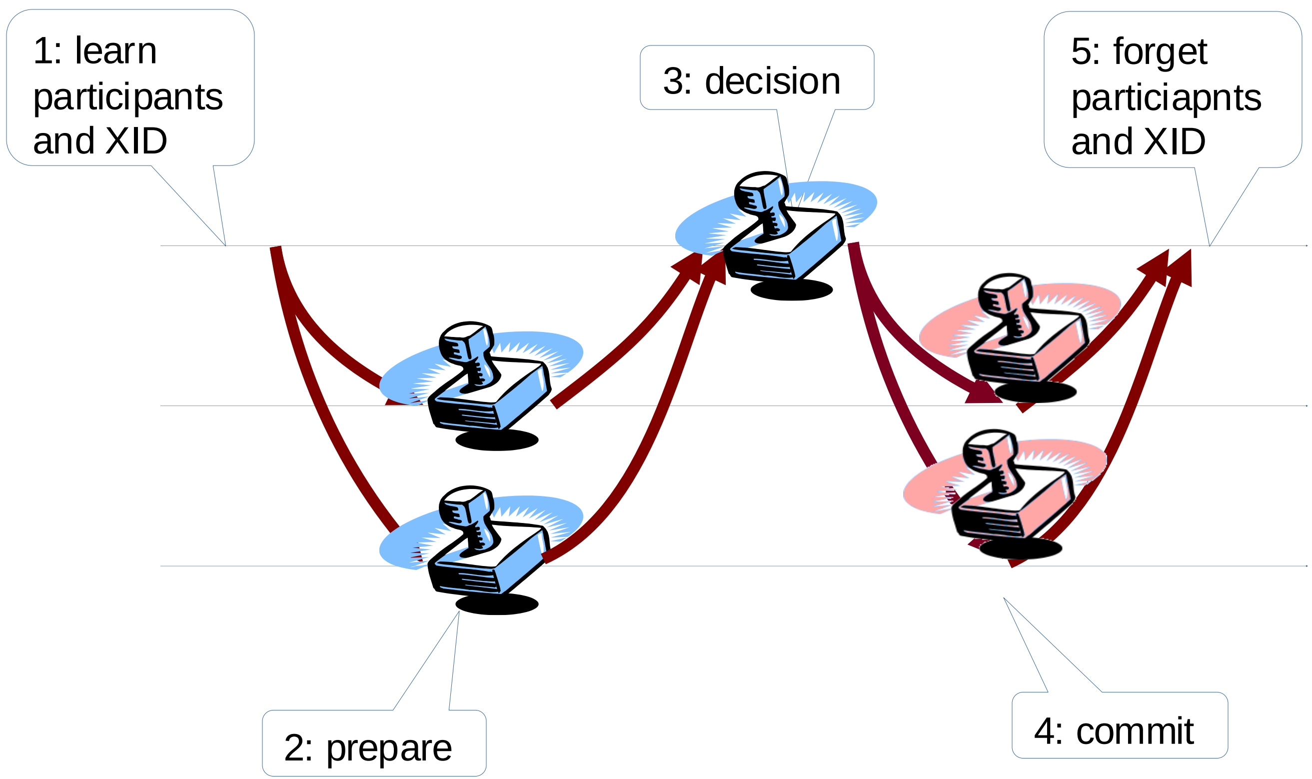
- A distributed system is composed by autonomous computing elements and can tolerate faults to avoid failure
- Fault model describes types of faults:
  - Omissive: process crash, lost message, …
  - Assertive (a.k.a. Byzantine): corrupted messages, …
- Fault model describes the number of faults:
  - Example: Number of processes that can crash

Transactional commit

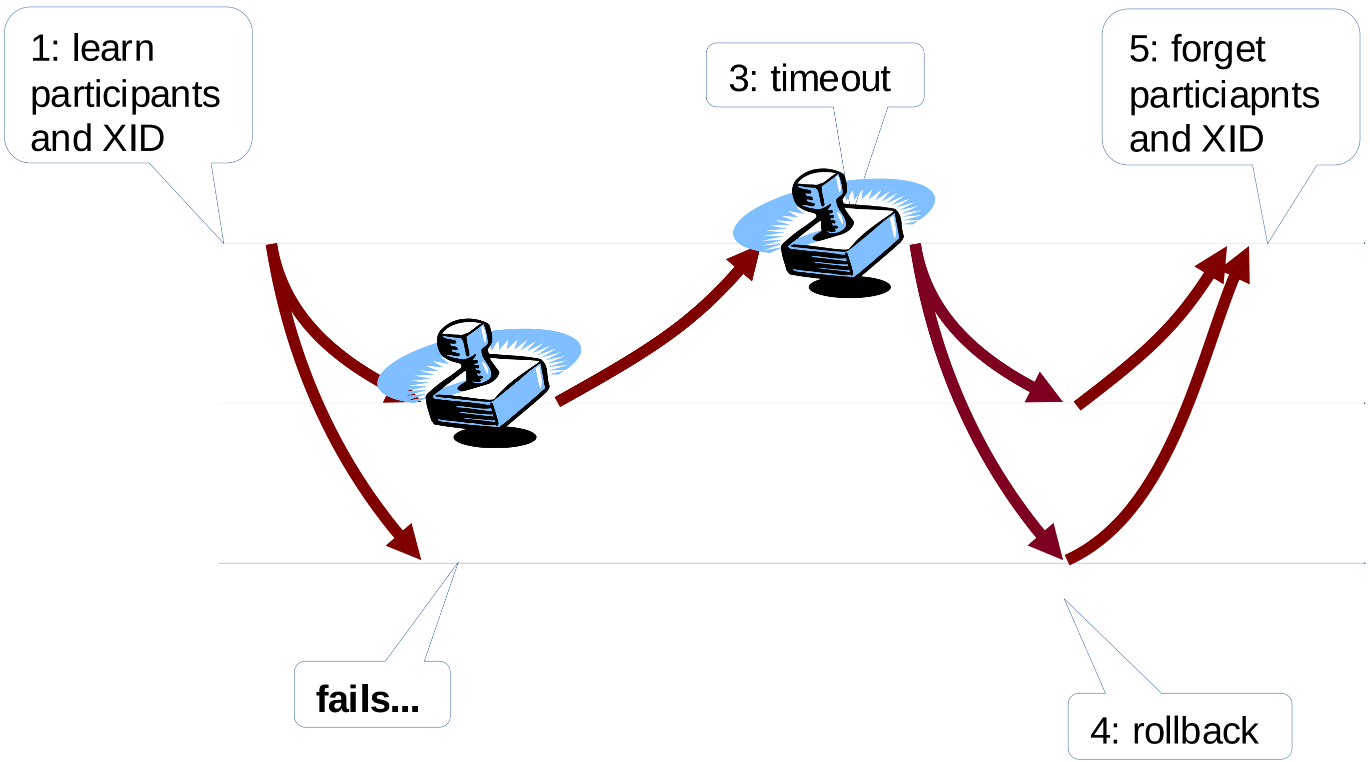
- Coordinate multiple irreversible actions across a distributed system:



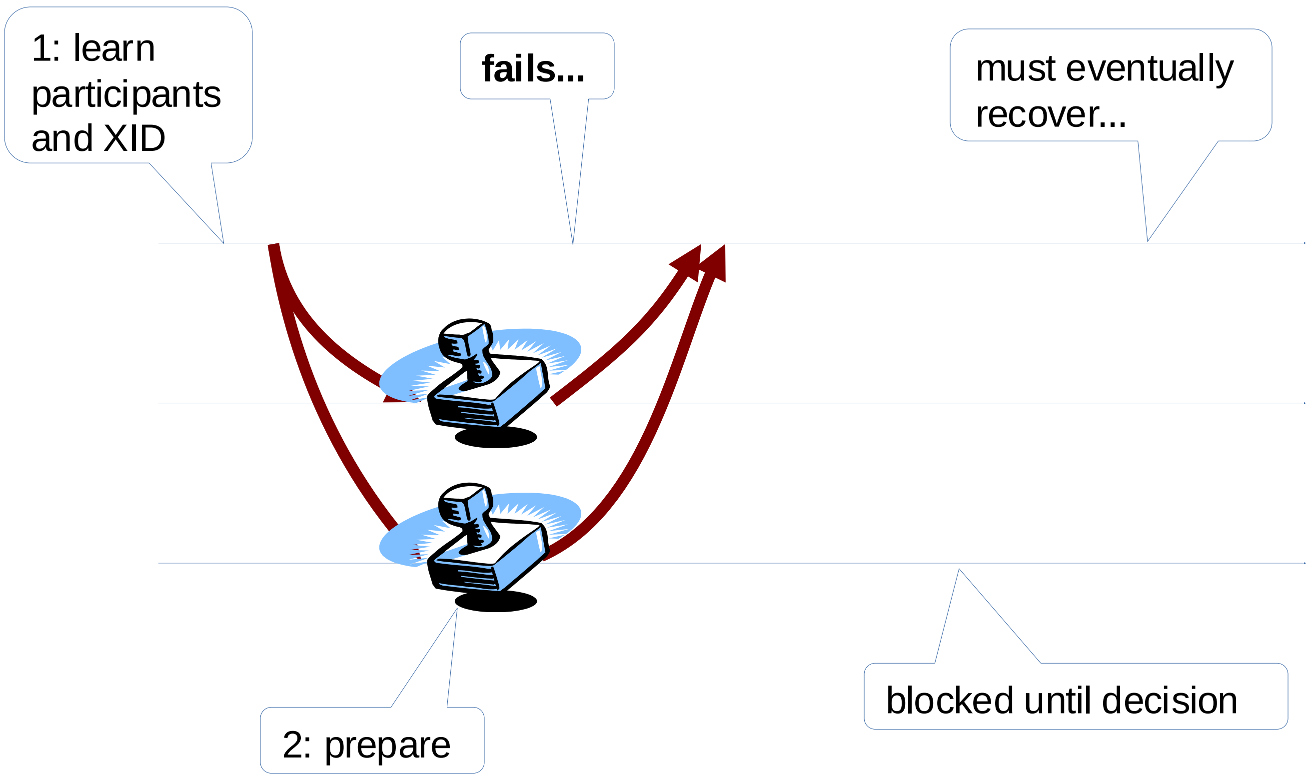
2-phase commit (2PC)



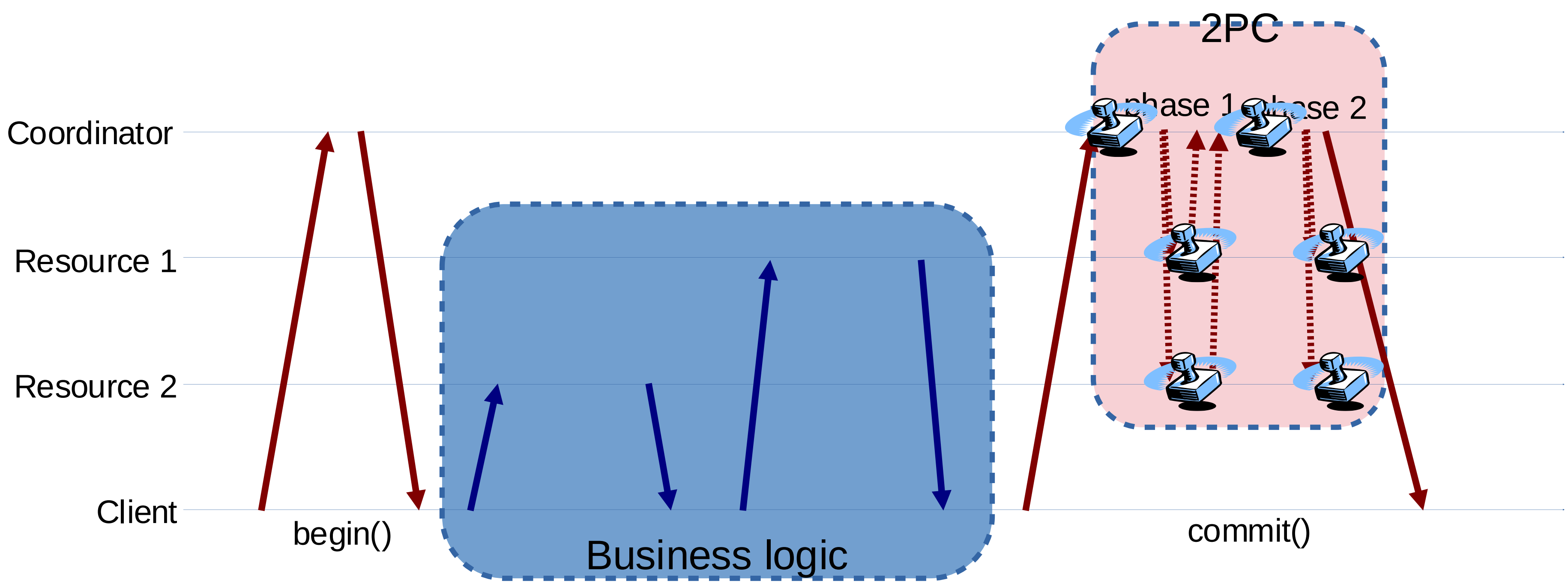
2PC: Participant failure



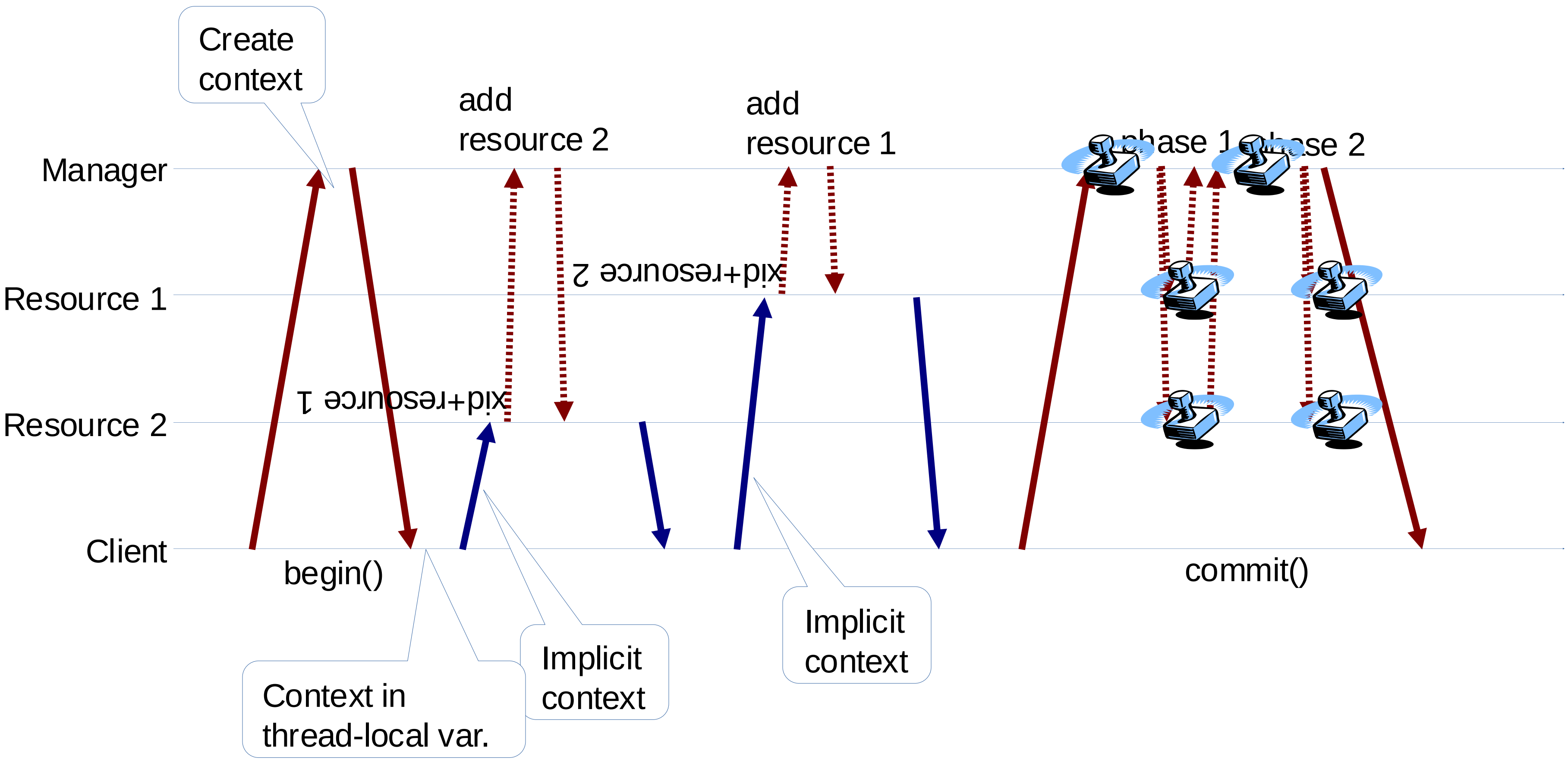
2PC: Coordinator failure



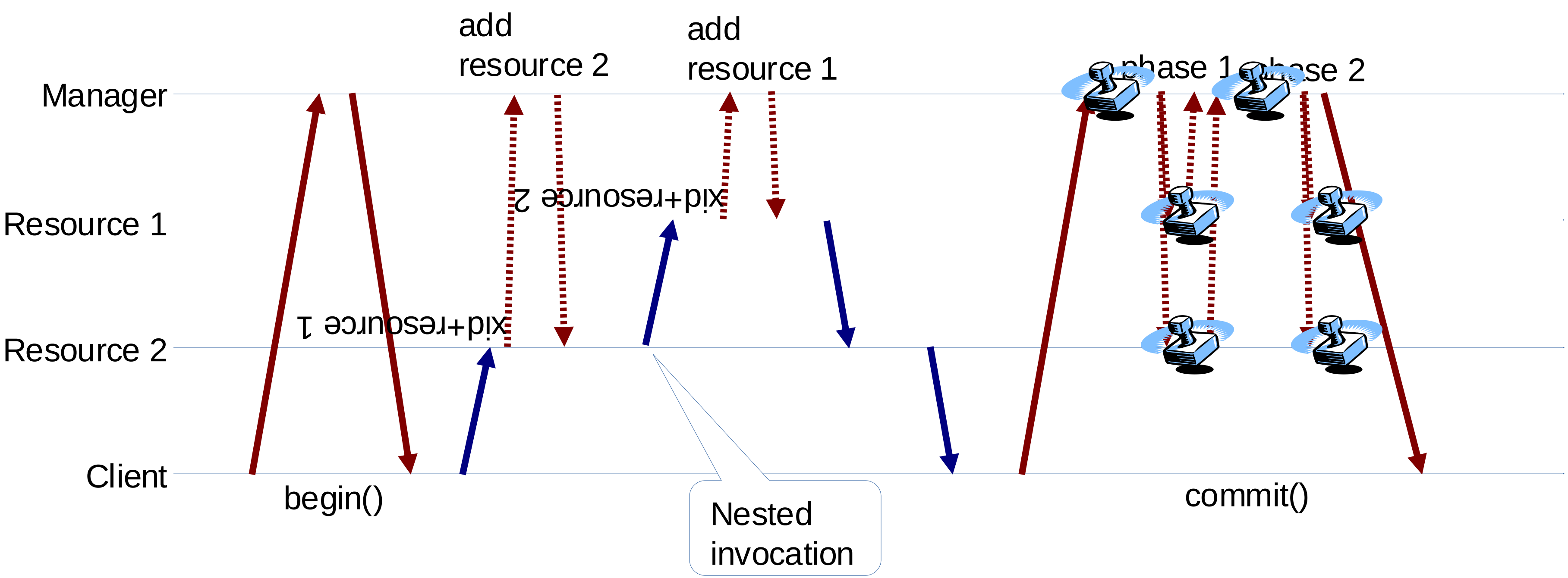
2PC in systems



Transactional RPC



Transactional RPC

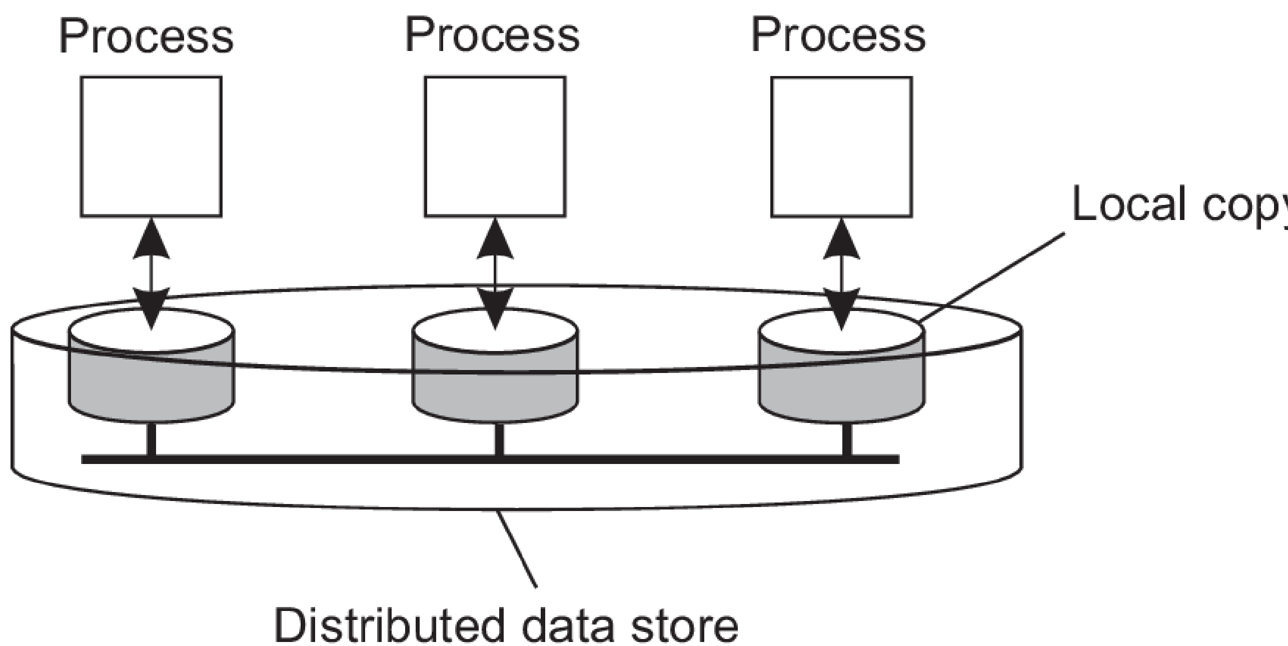


Summary

- Distributed transactions with 2-phase commit (2PC) support agreement in systems with faults
  - Limited to crash-recovery of the coordinator
- Is widely used in enterprise middleware for application integration

Replication

- Keep multiple copies of the same data or service
  - Distribute the load for scalability
  - Tolerate server faults

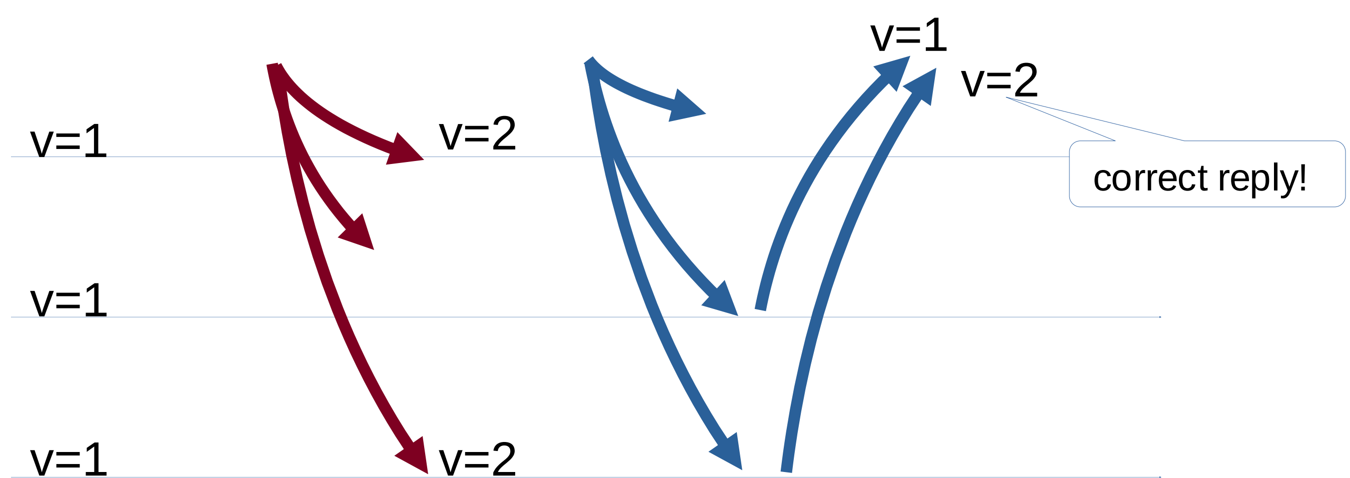


Replication

- Naive solution: write then propagate
  - state may diverge
  - clients observe paradoxes when reading
  - not fault-tolerant
- 2PC: Correct, but progress only with all up (tolerates reboots, not crashes)

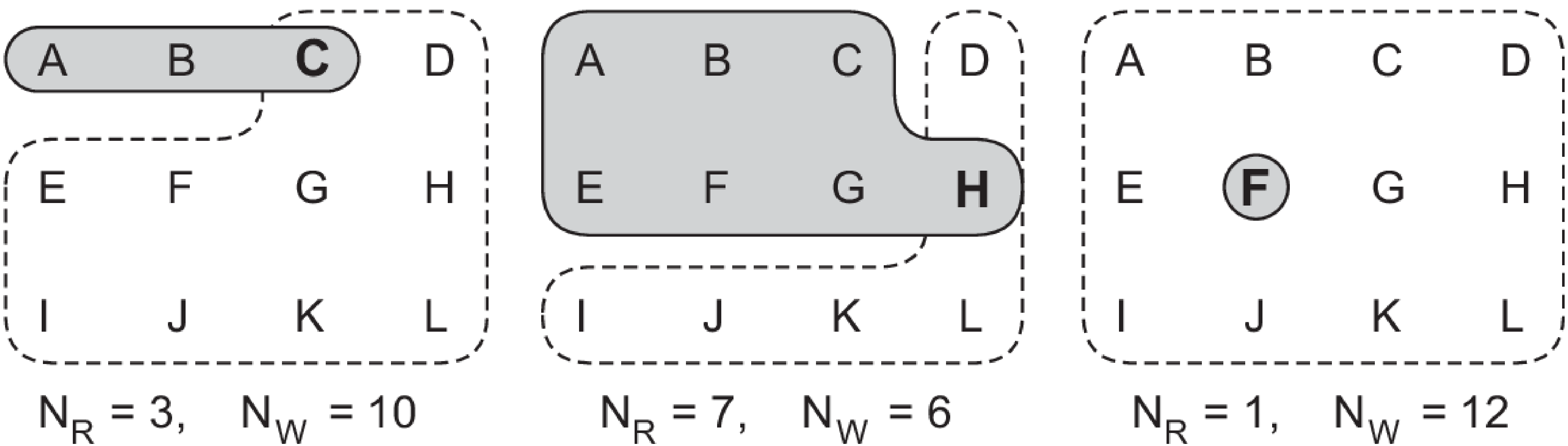
Replication

- Assume that:
  - operations are reads and writes
  - we keep a timestamp with each item
- It might be possible to read and write from fewer processes...



Quorum

- Assume 2-phase protocol for writing (phase 1 == read)
- Quorum rules for replicated data:
  - $N_R + N_W > N \rightarrow$ readers get the latest value
  - $N_W > N/2 \rightarrow$ concurrent writers conflict



Quorum

- Additional rules for fault-tolerance when assuming at most *f* faults:
  - $N_R + f \leq N \rightarrow$ readers never block
  - $N_W + f \leq N \rightarrow$ writers never block
- Can be configured to ensure both or either of them
- Typical solution is having a majority:
  - $N_R = N_W = f + 1$
  - $N = 2f + 1$
- Examples: N=3 for f=1, N=5 for f=2, ….

Summary

- Flexible and efficient solution for data replication
  - Example: Amazon Aurora DB
- Wasteful when there are multiple concurrent writers:
  - At most one of multiple write operation can be accepted
  - But it can happen that none is accepted if each operation is applied in less than N_W servers