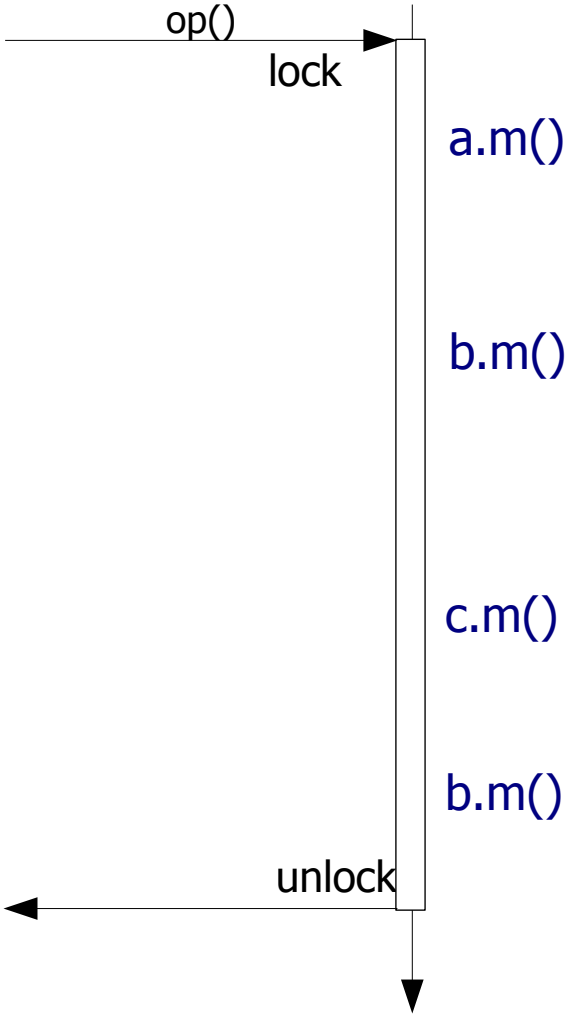
Challenges

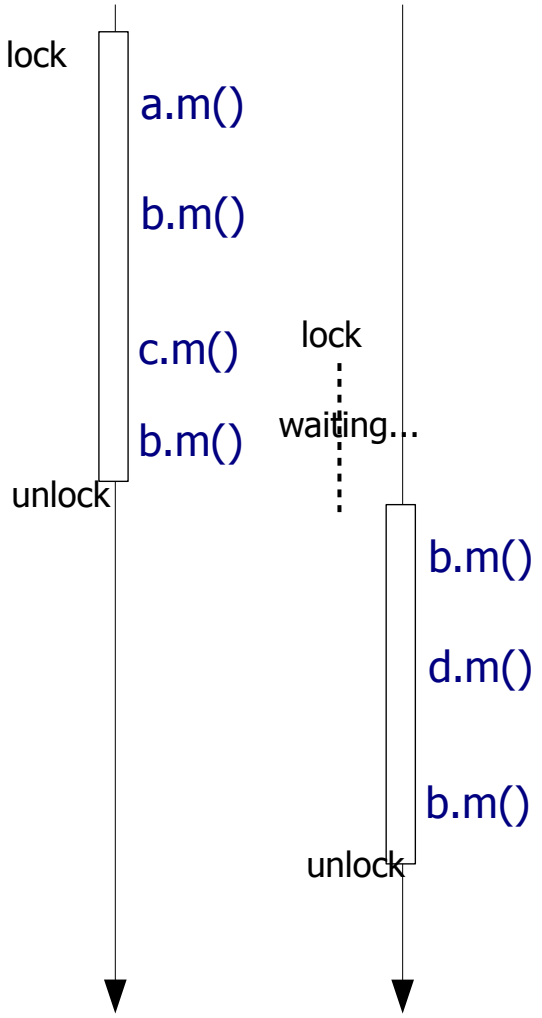
- Atomic operations on multiple distributed objects
  - Concurrent client invocations
  - Client and server failure

Locking



- Lock/unlock operations in a lock server
- A single global lock protects all data items
- Acquired for the duration of the composite operation

Locking

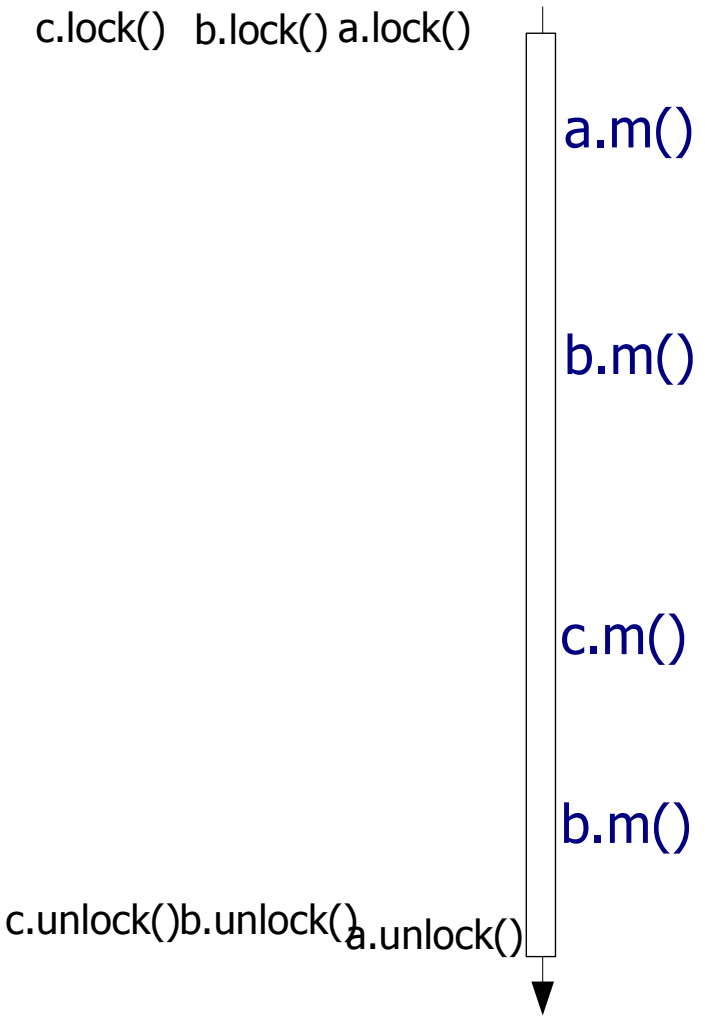


- Operations never overlap
- History is a sequence of operations

Summary

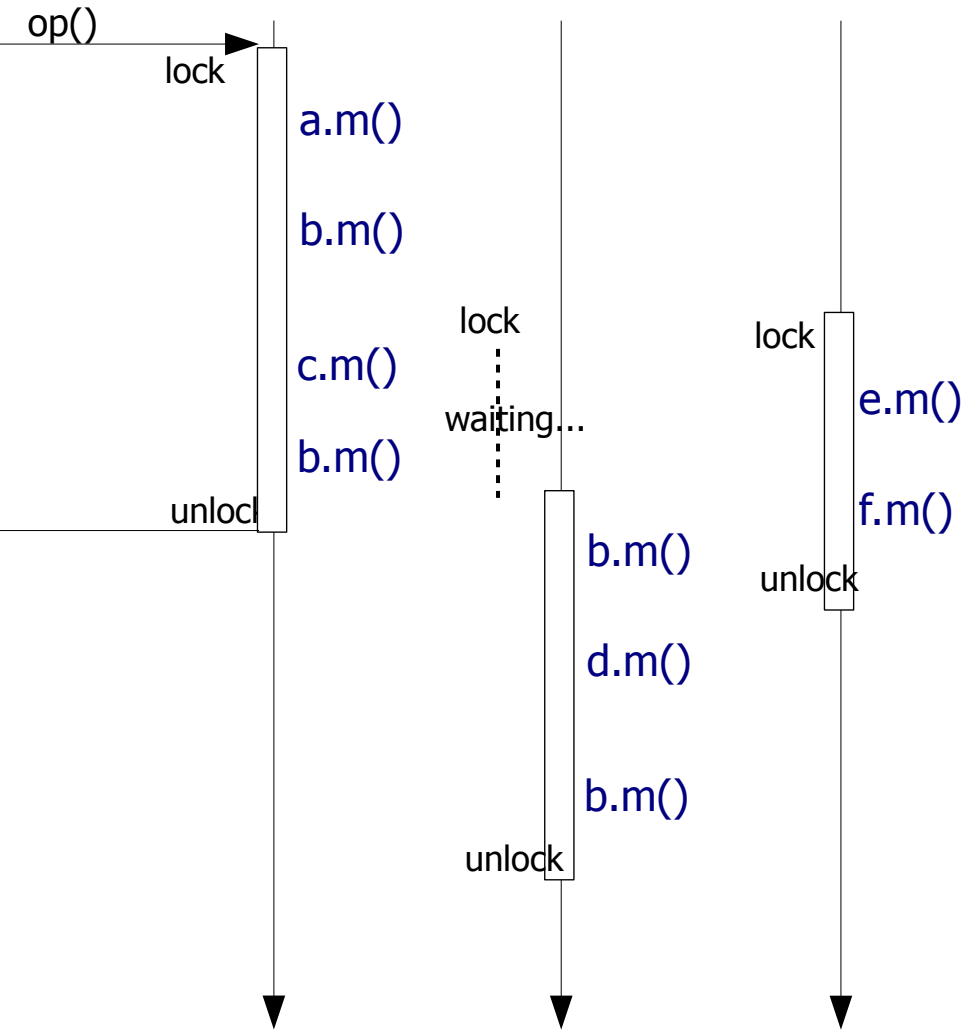
- Performance bottleneck:
  - At most one operation executing
- Remote invocations needed for lock / unlock
- Single point of failure:
  - Lock server fails
  - Lock client fails without unlock()

Locking



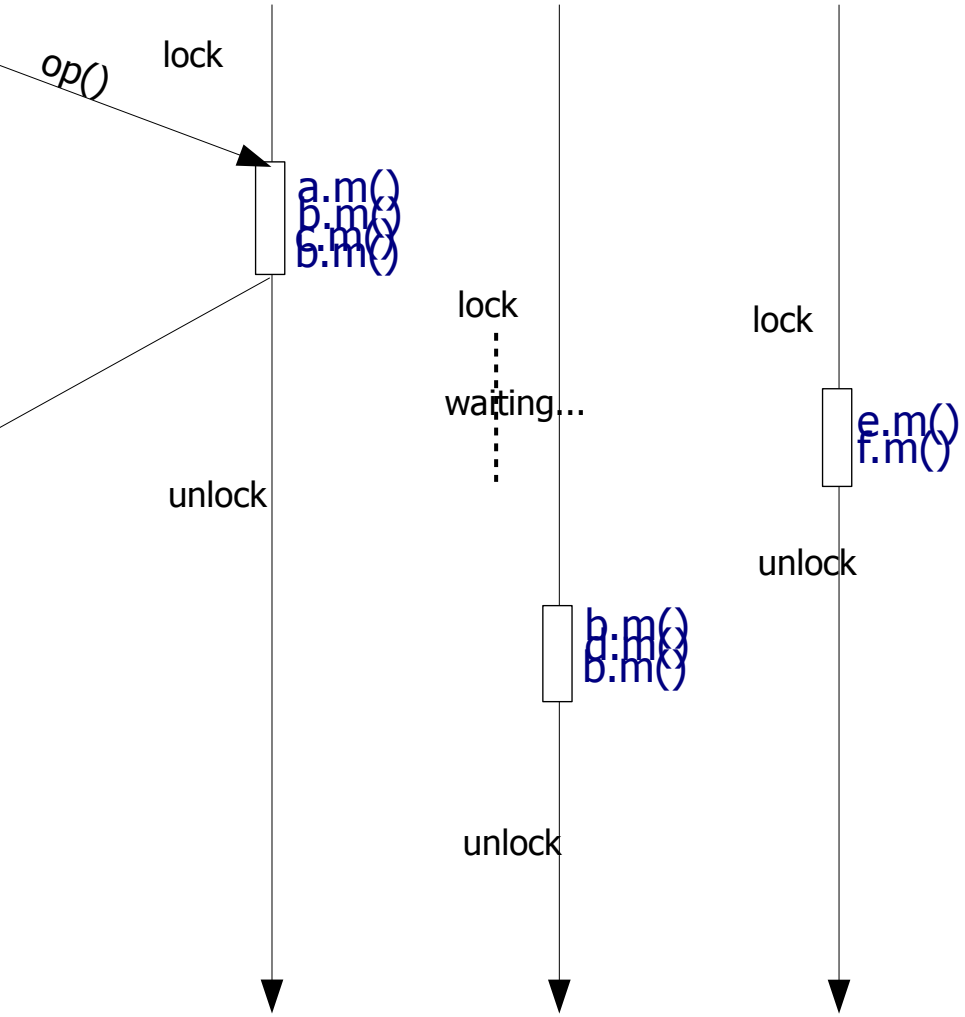
- Multiple lock objects
- Each data item is protected by a lock
- All locks are acquired during the operation

Locking



- Operations on different items may overlap

Locking

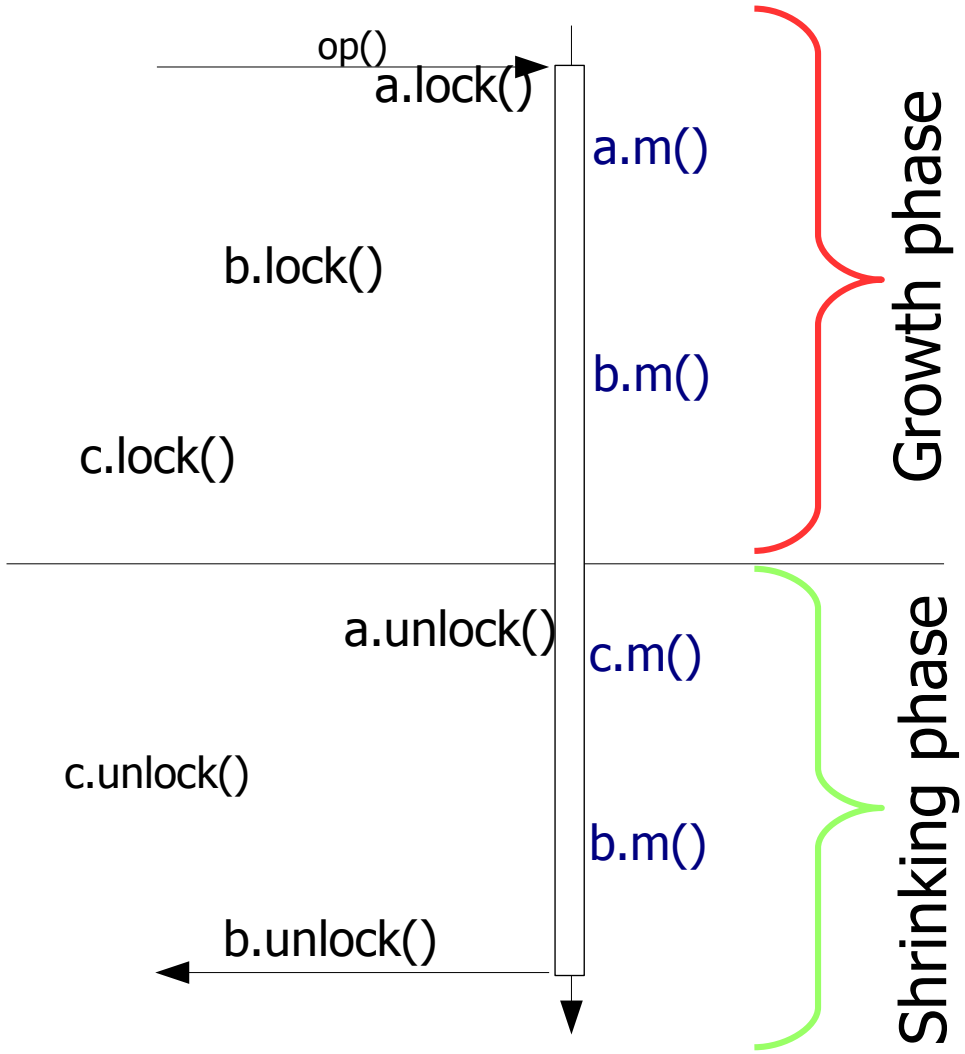


- Indistinguishable from:
  - Instantaneous execution with delay on round-trip
- Clients cannot deny that operations do not overlap

Summary

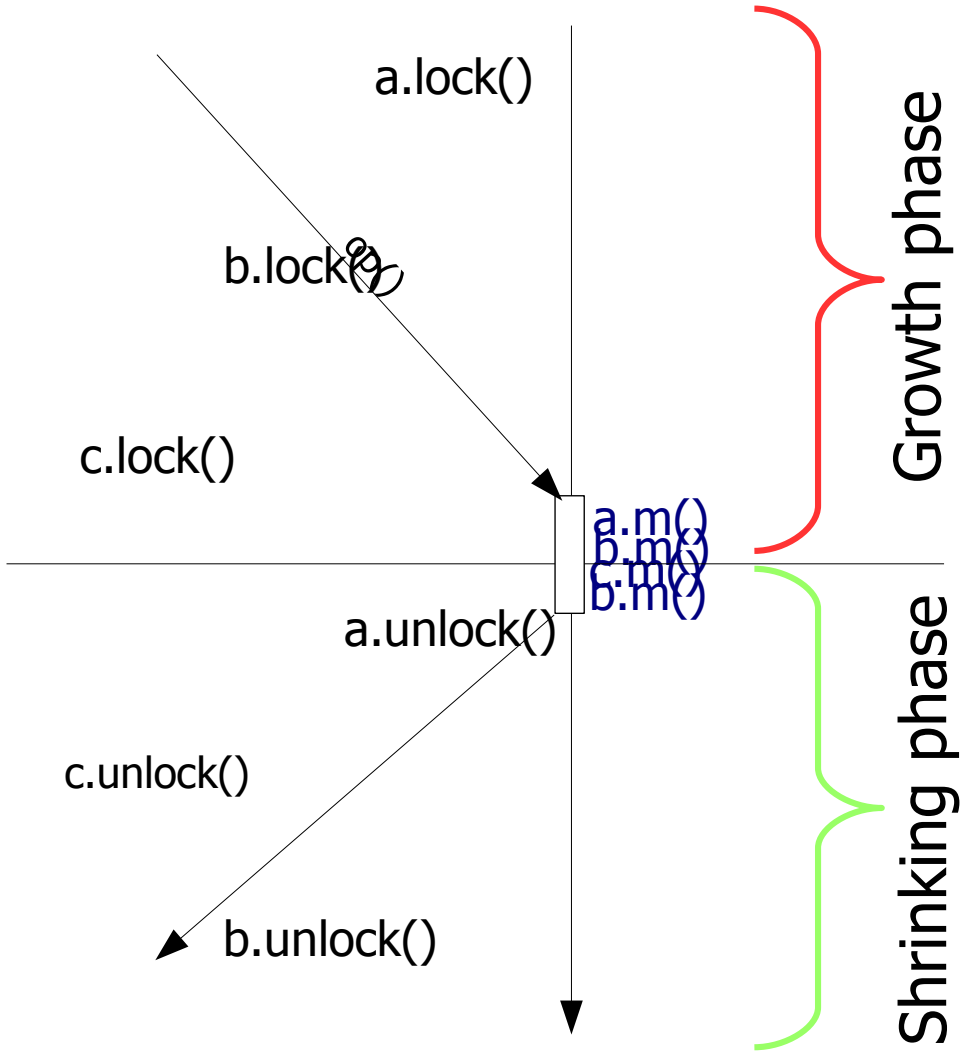
- ~~Performance bottleneck~~
- Remote invocations needed for lock / unlock
- Single point of failure:
  - ~~Lock server fails~~
  - Lock client fails without unlock()
- Must know all items accessed on start
- Deadlocks

2-Phase locking



- All lock requests precede all unlock requests
- Lock can be implicit in operation

2-Phase locking

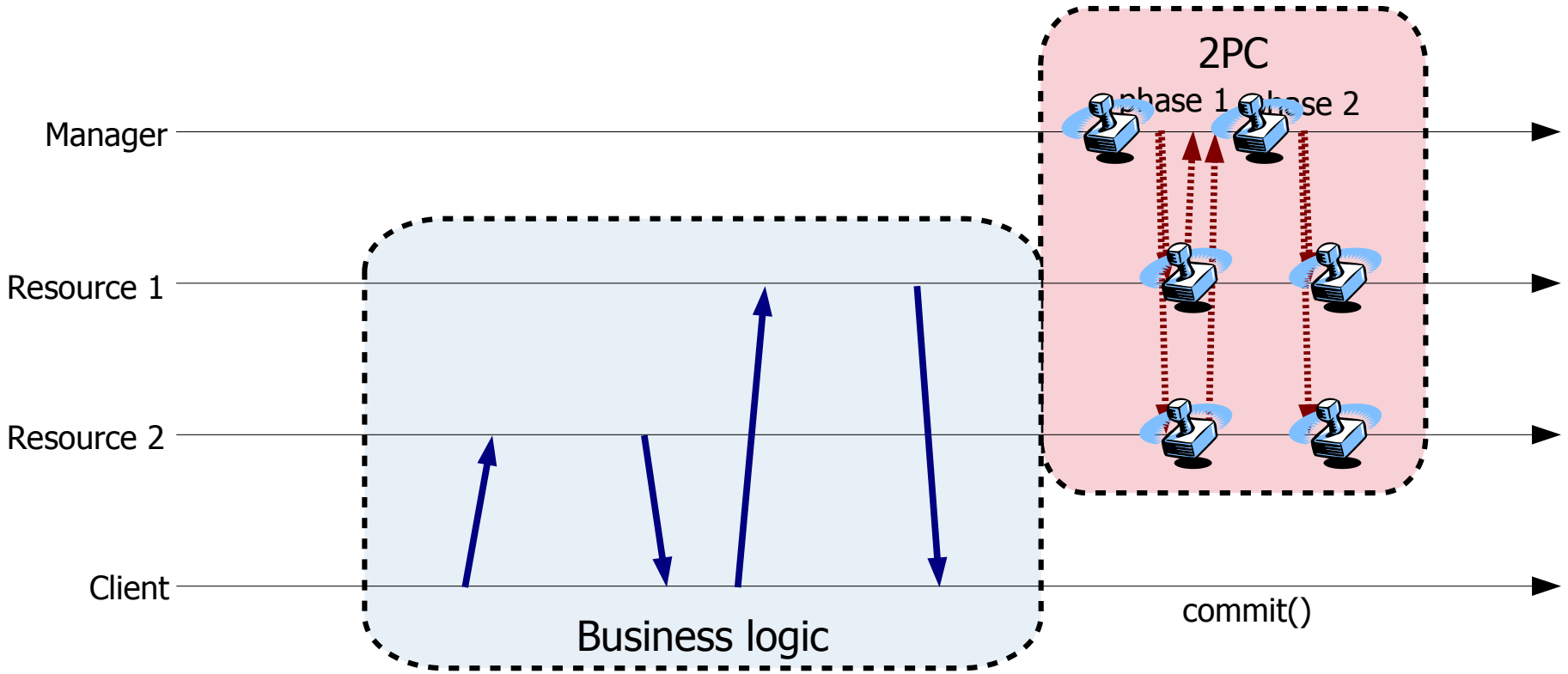


- An observer cannot deny that all happens in-between phases

Summary

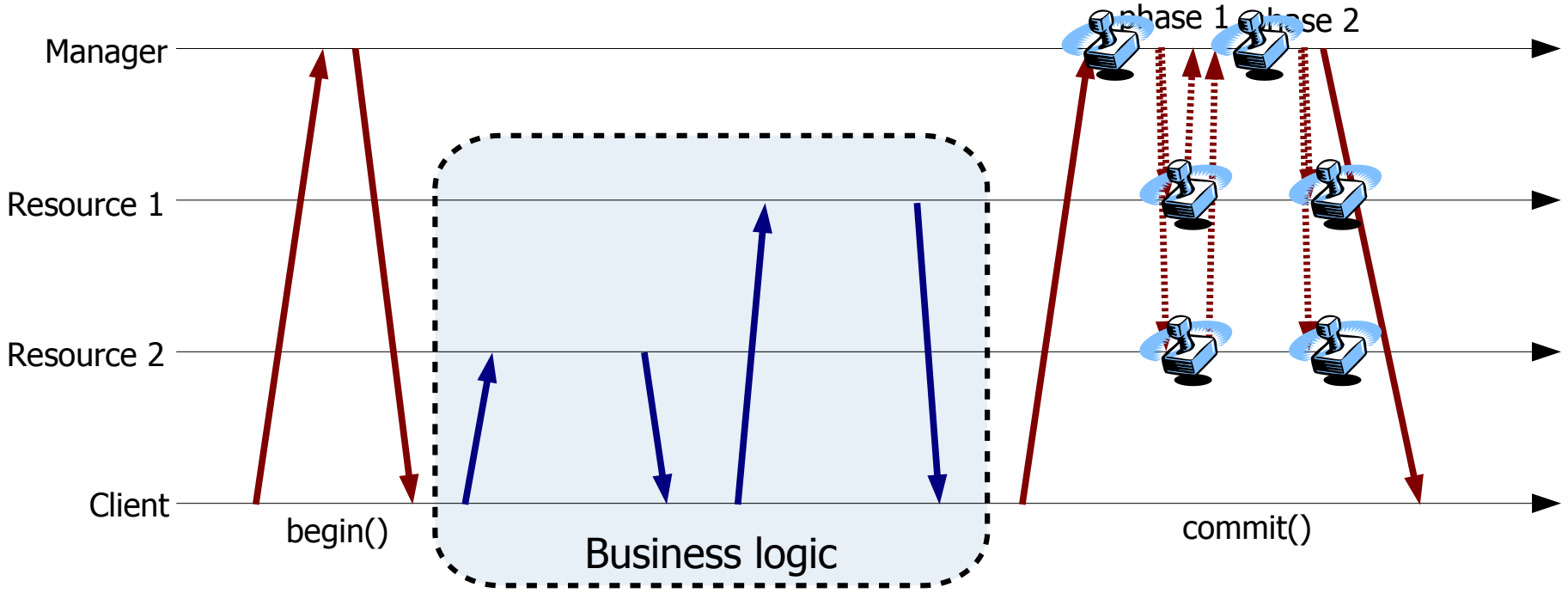
- ~~Performance bottleneck~~
- Remote invocations needed for ~~lock~~ / unlock
  - Lock operation is implicit
- Single point of failure:
  - ~~Lock server fails~~
  - Client fails without unlock()
- ~~Must know all items accessed on start~~
- Deadlocks

Application vs Transactions



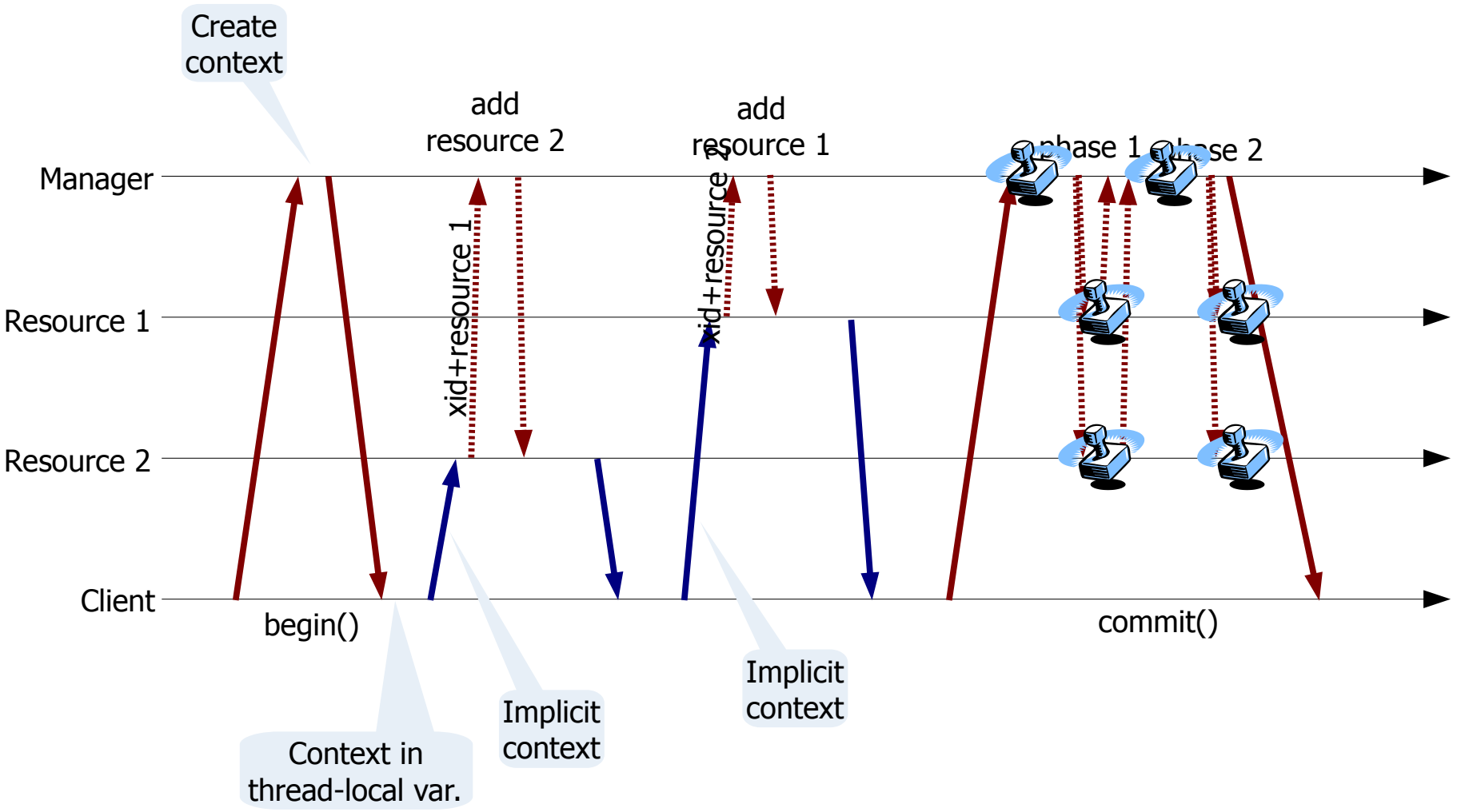
- How to connect them?
  - Don’t change business logic

Transaction demarcation

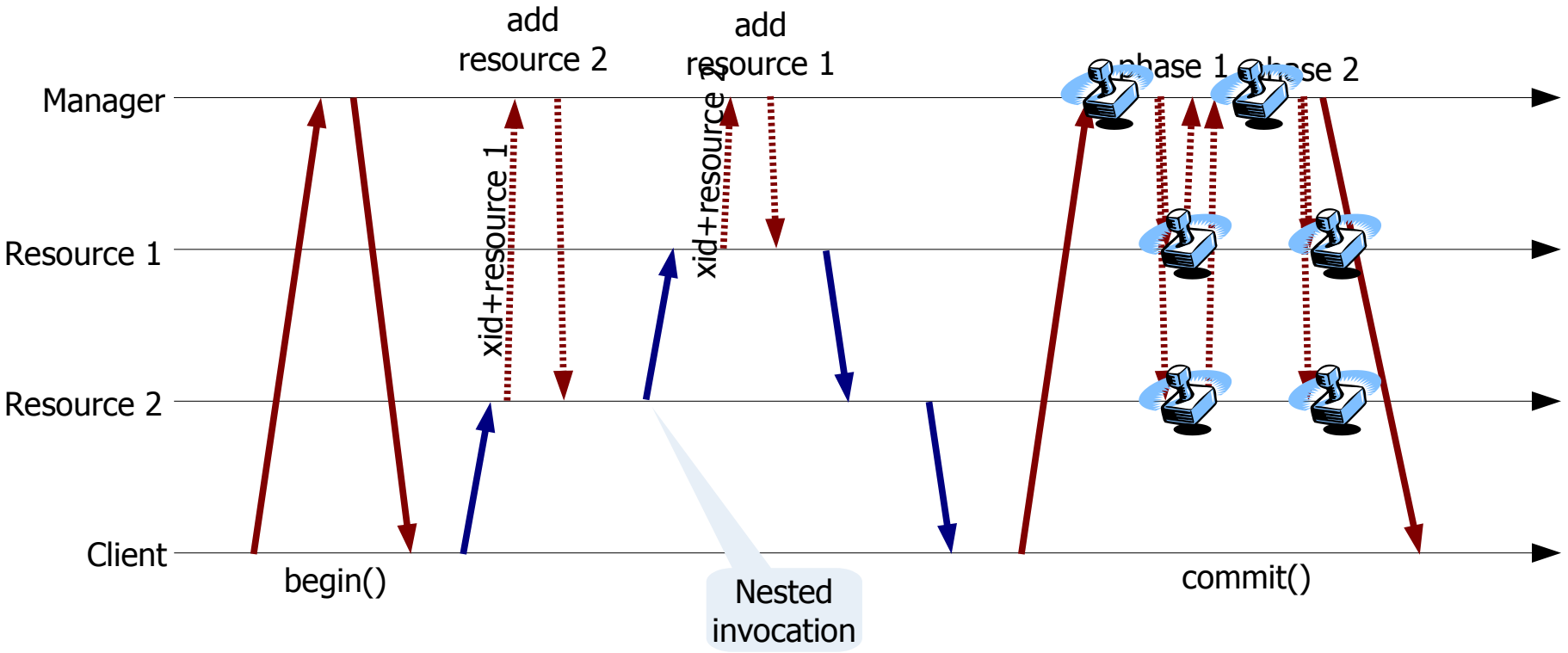


- Call transaction manager before/after
- How does manager discover participants?

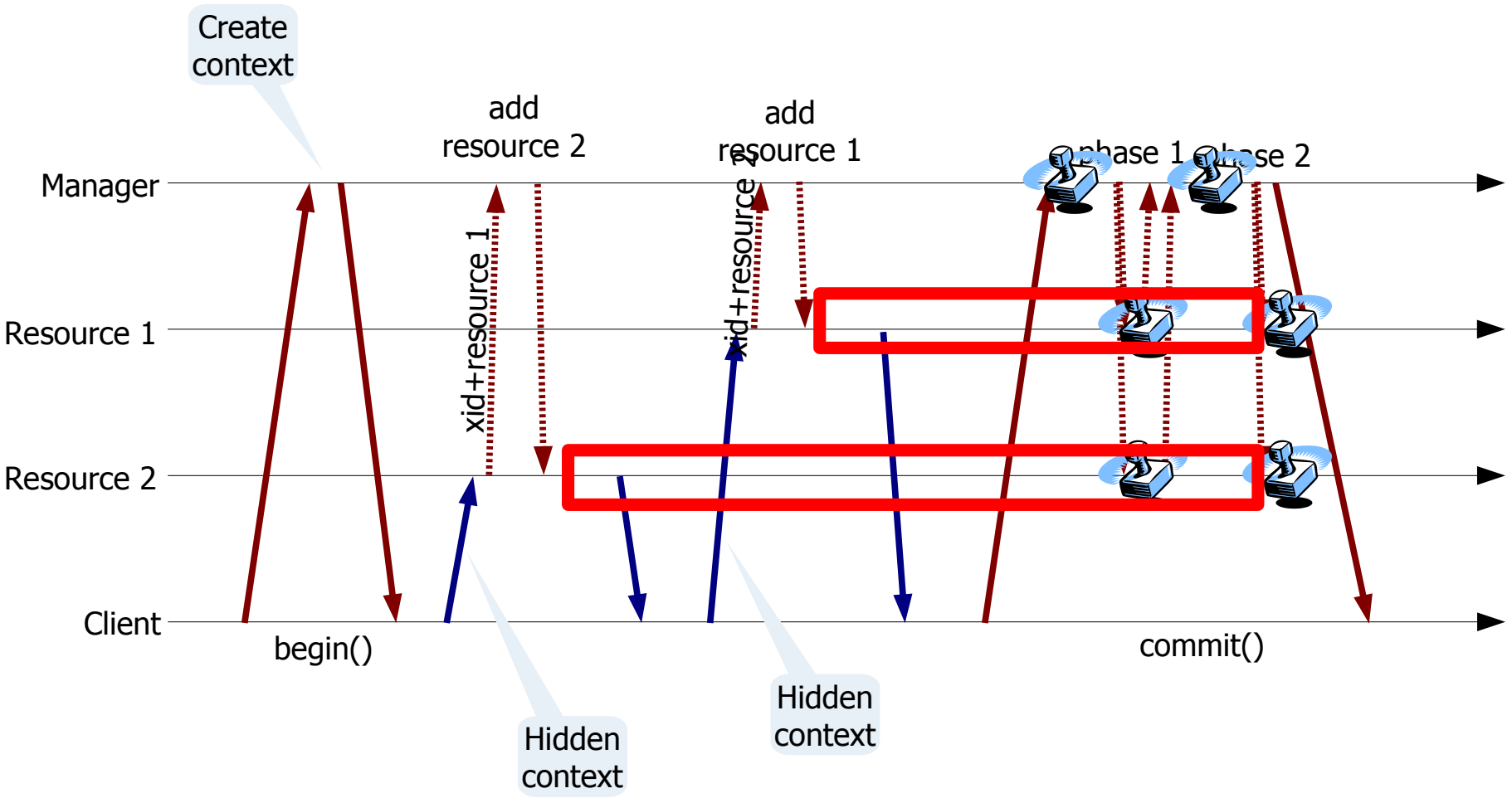
Transactional RPC + 2PC



Transactional RPC + 2PC



Transactional RPC + 2PC + 2PL



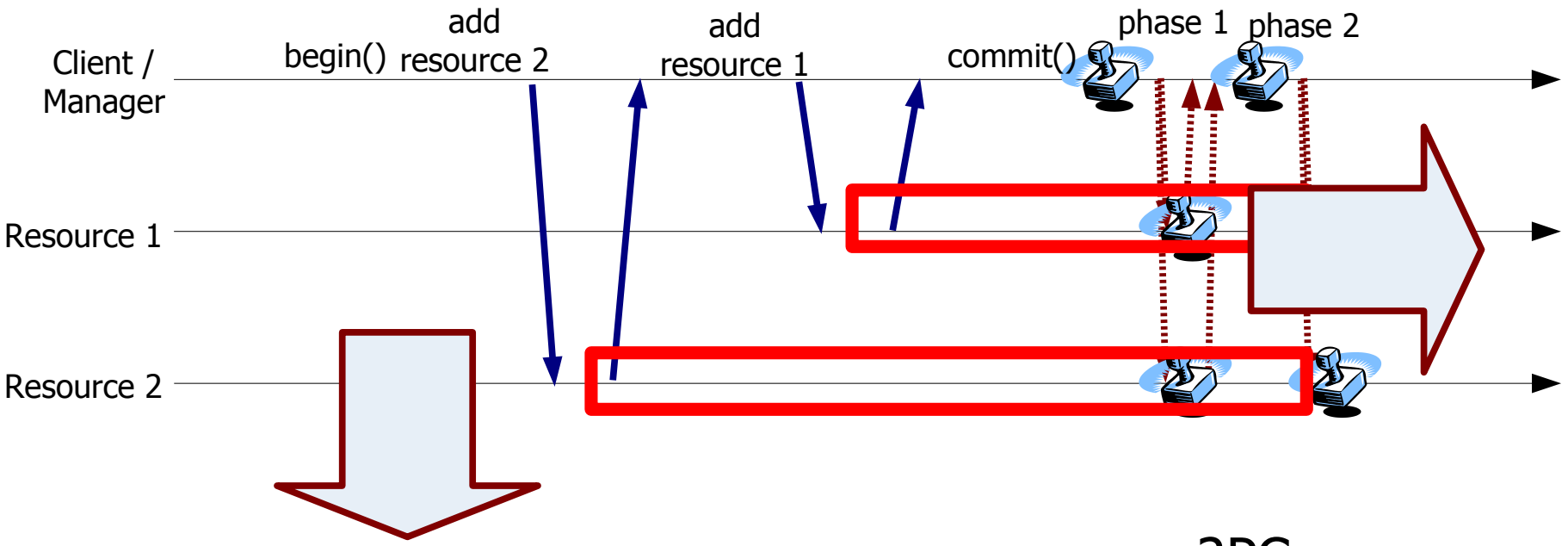
Summary

- ~~Performance bottleneck~~
- ~~Remote invocations needed for lock / unlock~~
  - Lock operation is implicit
  - Unlock operation is implicit (txn. commit)
- Single point of failure:
  - ~~Lock server fails~~
  - ~~Lock client fails without unlock()~~ (rollback)
- ~~Must know all items accessed on start~~
- ~~Deadlocks~~ (rollback)

Summary

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

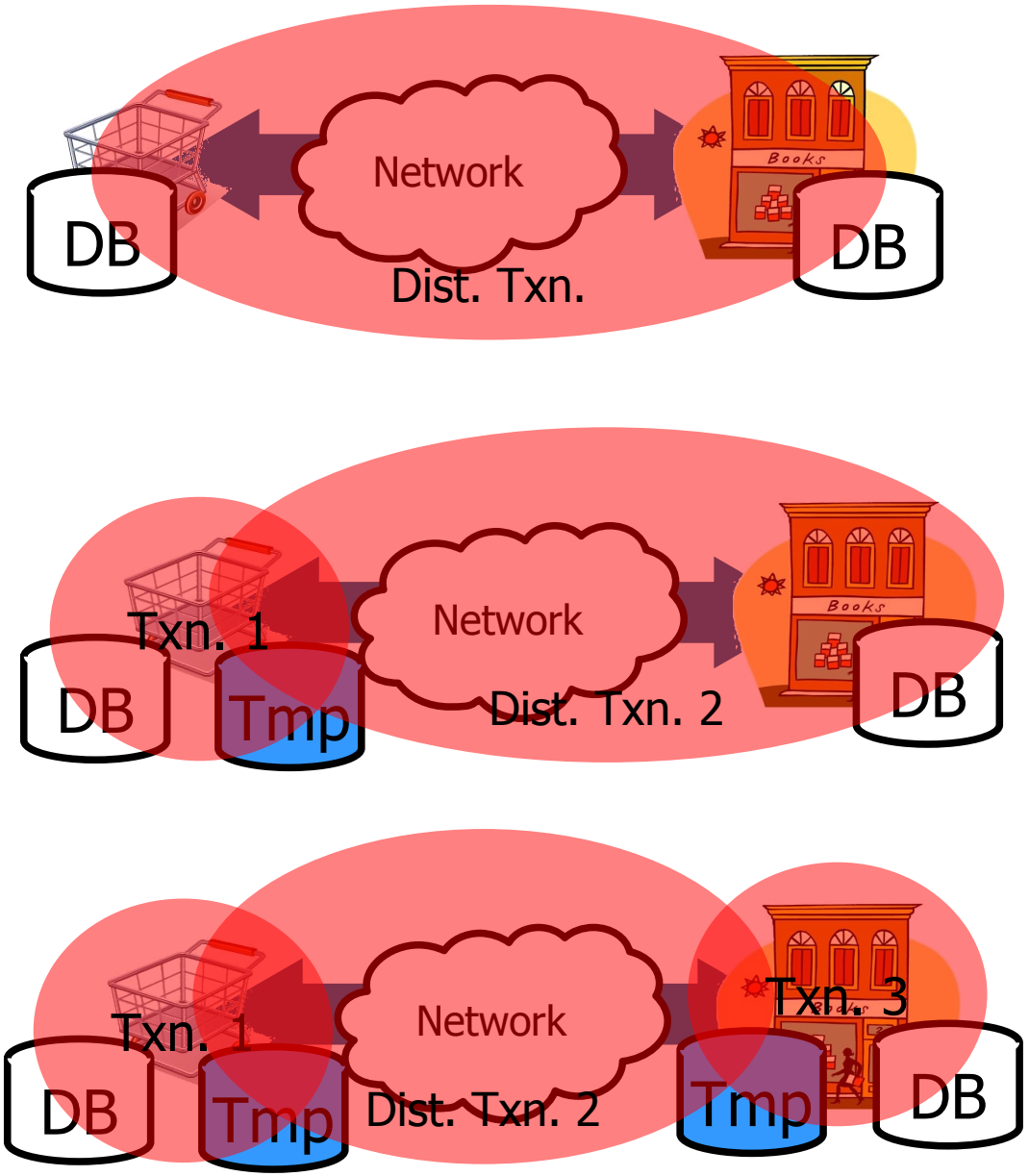
2PC+2PL: Challenges



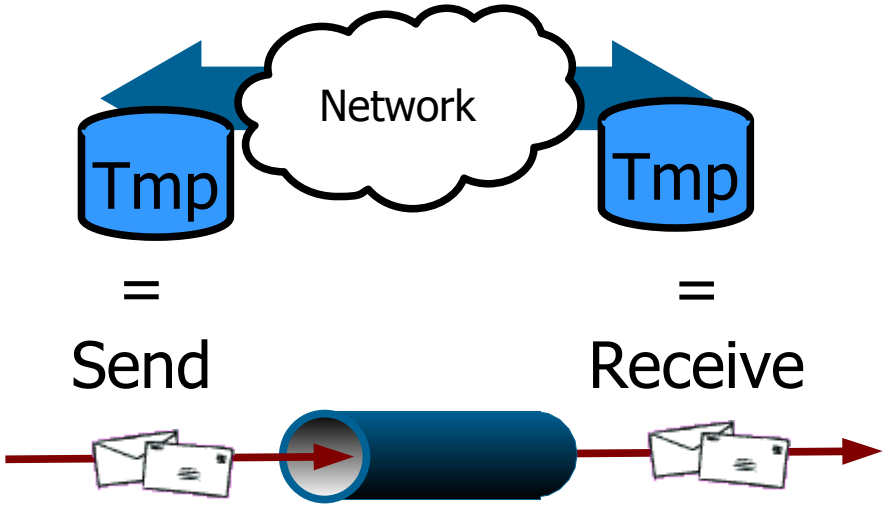
2PC

- Scalability with:
  - More resources?
  - Slower 2PC (remote resources)?

Store and Forward



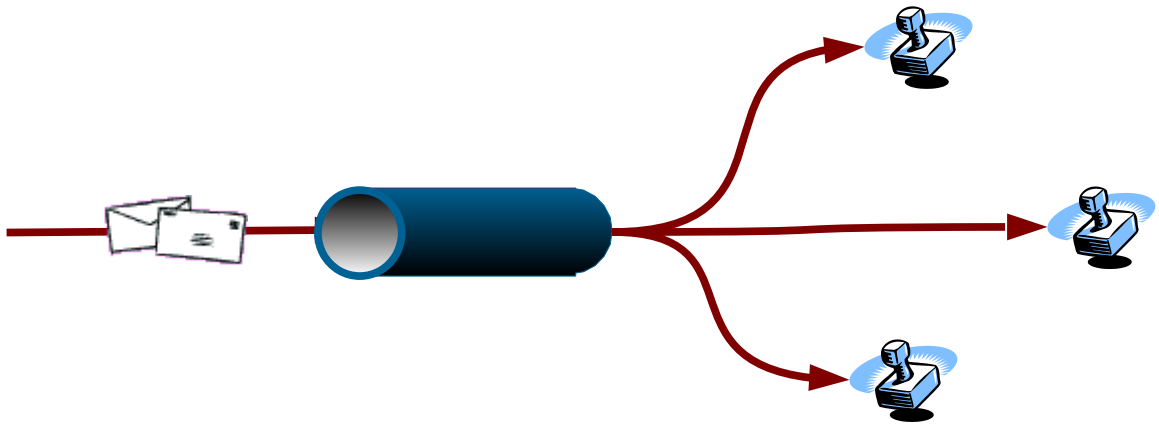
Transactional queues



- Insert and remove from temporary DBs abstracted as transactional send/receive
- Advantages:
  - Performance and fault decoupling
  - Asynchronous (pull) operations

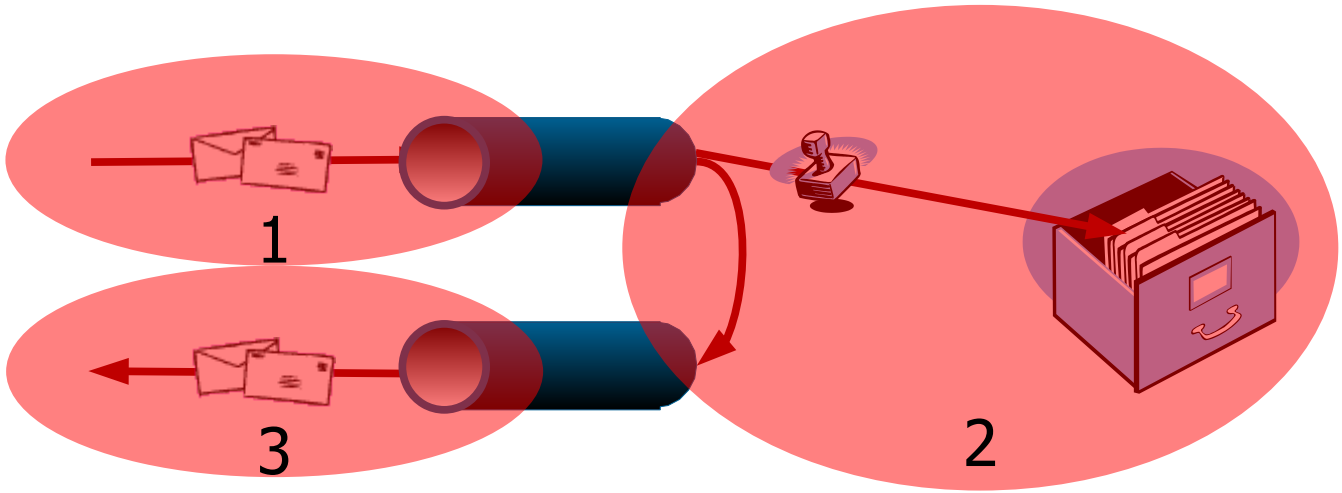
Server cluster

- Multiple servers polling the same queue:
  - Each message processed exactly once
  - No order enforced



Request/Reply

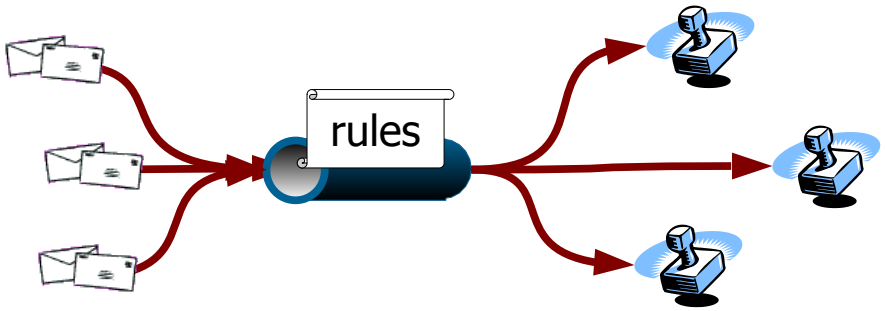
- Client/server:
  - Server polls for requests
  - Client polls for replies
- Multiple transaction contexts:



Limitations of queues

- Consider adding another application...
- Addressing is explicit in applications:
  - Each sender must know exactly which receivers are interested in a message

Message brokers



- Goal: Decouple message routing from sending and receiving
- Receivers subscribe to messages that fit some criteria
- Senders publish messages

Topic-based routing

- Publisher tags messages with a topic id
- Subscriber requests a topic
- Options:
  - Flat topics
  - Hierarchical topics

Content-based routing

- Publisher structures message content using a common structure
- Subscriber registers a content filter
- Options:
  - Relational data / SQL WHERE clauses
  - XML / XQuery

Summary

- Transactional messaging decouples:
  - Faults
  - Performance
  - Application logic
- Two options:
  - Tightly coupled applications: 2PL + 2PC + TRPC
  - Loosely coupled: 2PC + MQ