Week 7
Spyros Mastorakis

Outline

- Distributed Transactions
- Midterm Material Recap

Distributed Transactions

Executing Transactions

- To ensure atomicity, execute one transaction at a time
  - Problem?
- Desired property: serializability
- Despite concurrent execution, externally visible effects equivalent to some serial order of execution

Example of Serializability

- Concurrent execution of transactions:
  - T1: transfer $10 from Alice to Bob
  - T2: Read balance in Alice's and Bob's accounts
  - Initial balance is $100 in both accounts

- Permissible outputs for T2?
  - (Alice: $100, Bob: $100) or (Alice: $90, Bob: $110)

- Invalid outputs for T2?
  - (Alice: $90, Bob: $100) or (Alice: $100, Bob: $110)

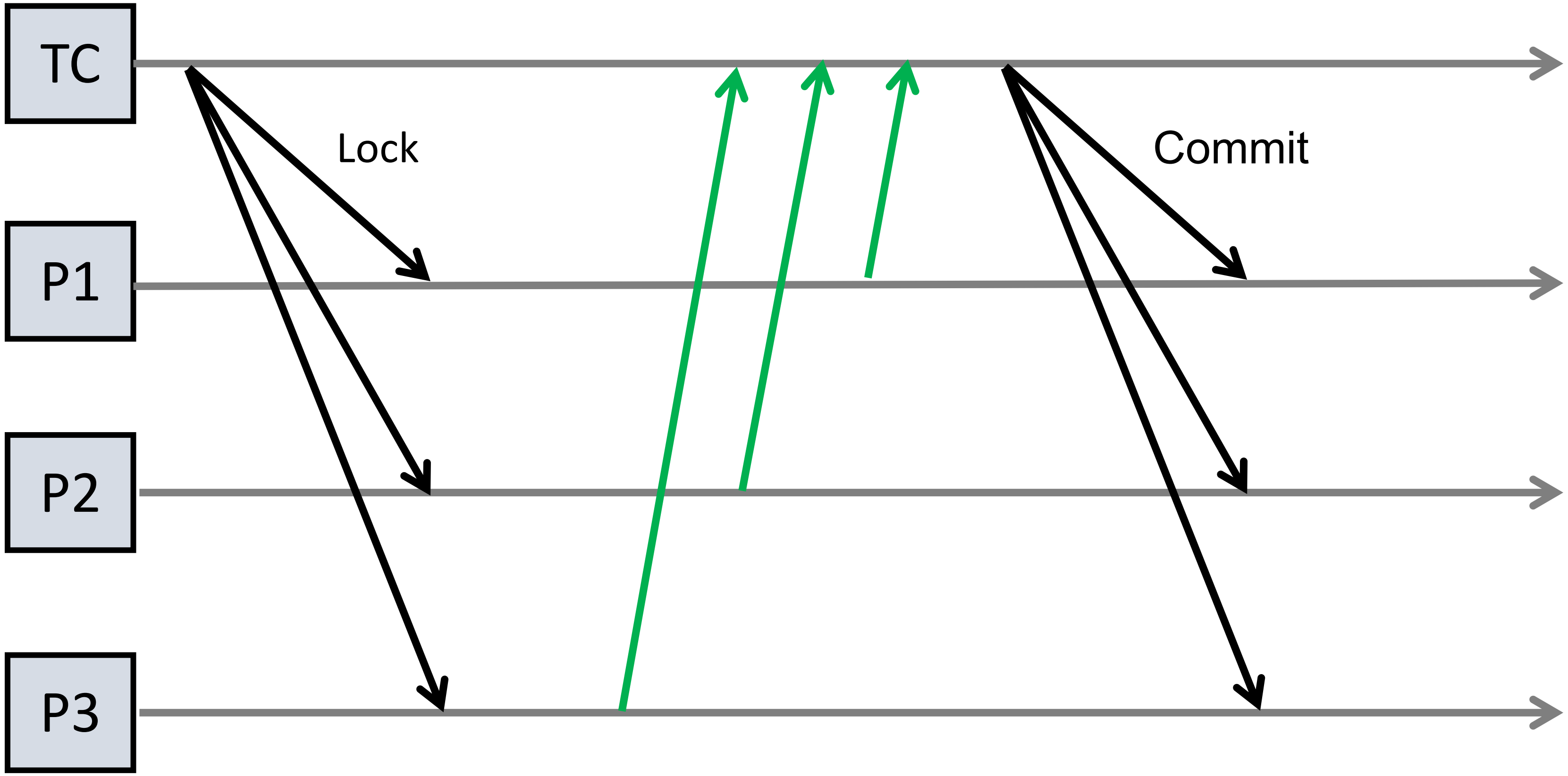
Example Scenario

- Many students are each looking to host a party
- Each host invites a subset of other students
- Party is on only if all invitees can make it
- Consensus among all students
  - Slow to identify all parties that are on
- How can each host decide whether his/her party is on without coordinating all students?

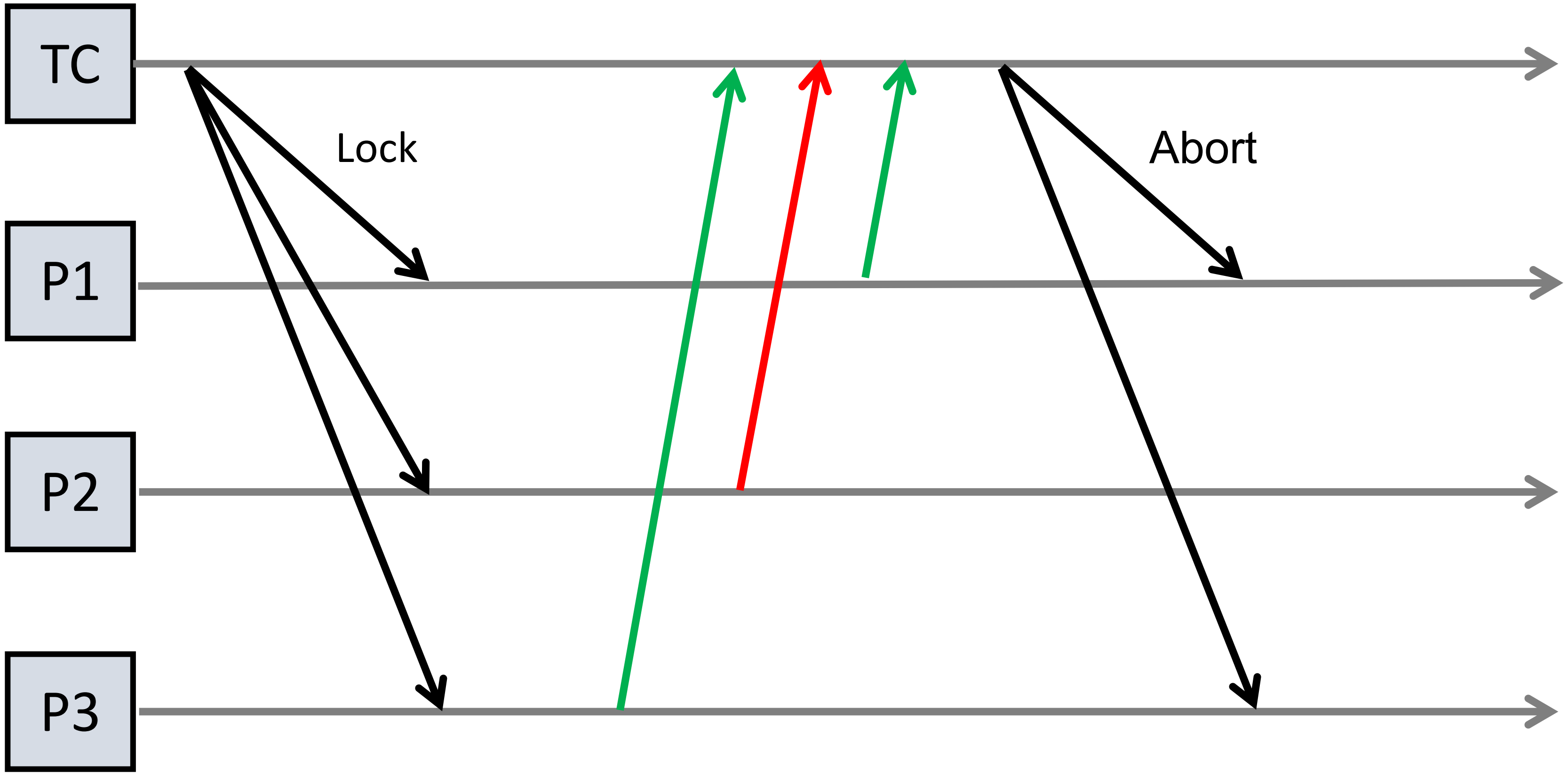
Achieving Serializability

- Client submits transaction to coordinator (TC)
- TC acquires locks on all data involved
- Once locks acquired, execute transaction and release locks

Two Phase Locking



Two Phase Locking



Two Phase Locking

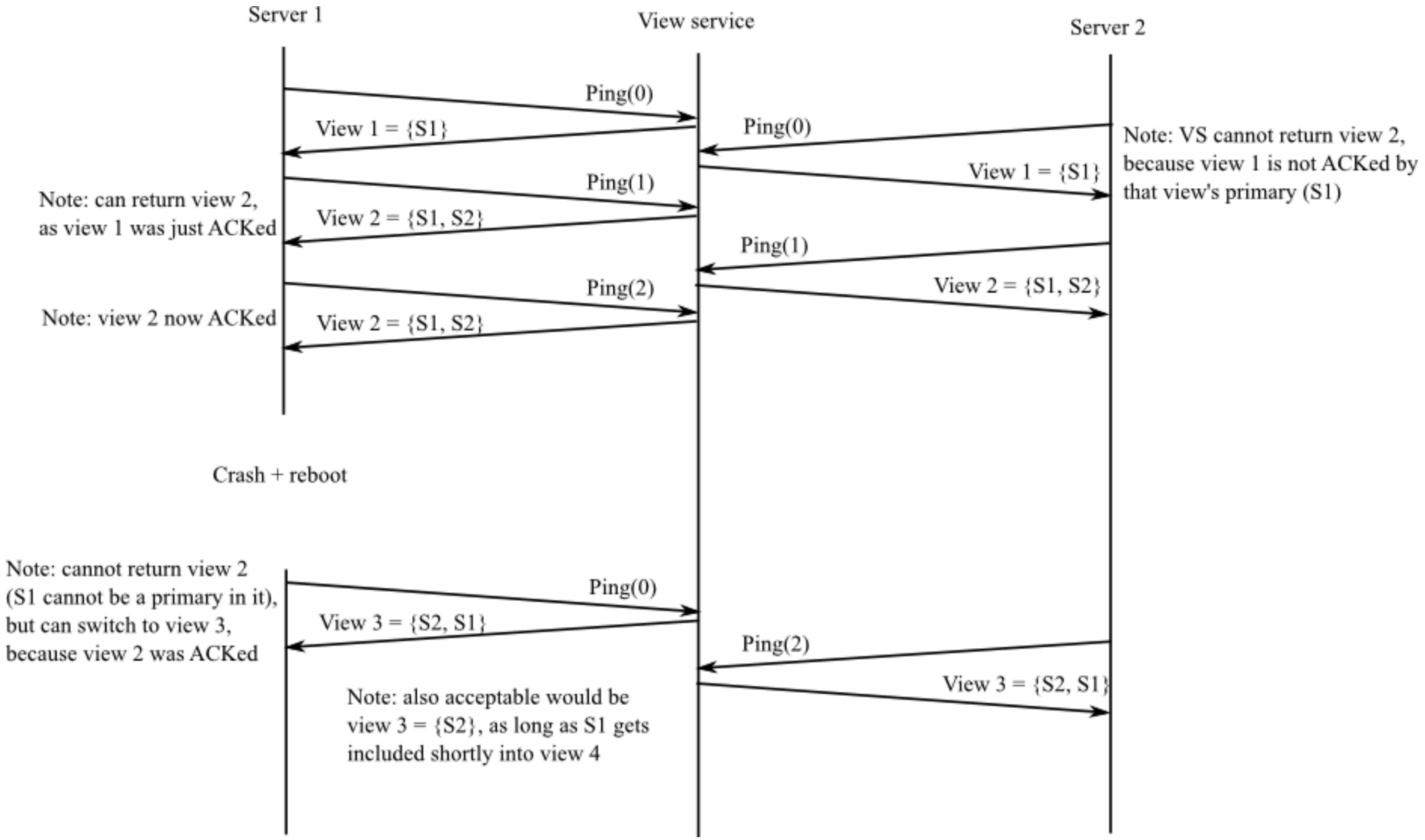
- TC acquires locks on all necessary shards before attempting transaction

- Disjoint transactions can execute concurrently
- Related transactions wait on each other

Midterm Material Recap

Project 2

- Viewservice



Paxos Overview

- Three phases in each round

- Prepare Phase
  - Proposer sends a unique proposal number to acceptors
  - Waits to get commitment from majority of acceptors

- Accept Phase
  - Proposer sends proposed value to acceptors
  - Waits to get proposal accepted by majority

- Learn Phase
  - Learners discover value accepted by majority

Paxos: sample execution

- Acceptor1: P1 A1-X P2
- Acceptor2: P1 A1-X P2 P3
- Acceptor3: P1 A1-X P3

Paxos: sample execution

- Acceptor1: P1 A1-X P3 A3-X
- Acceptor2: P1 P2 A1-X P3 A2-Y A3-X
- Acceptor3: P2 A2-Y

Paxos: sample execution

• Acceptor1:	P1			
• Acceptor2:	P1	A1-X	P2	A2-X
• Acceptor3:			P2	A2-X

Paxos: sample execution

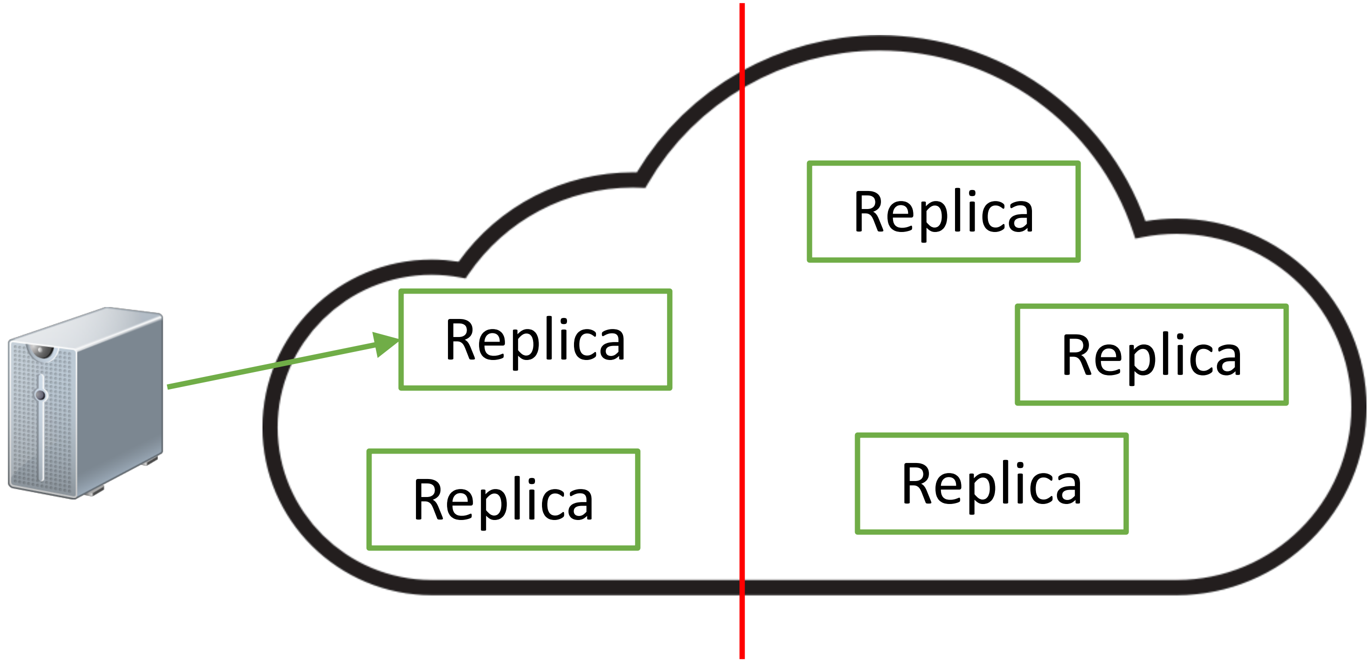
- Acceptor1: P1 A1-X
- Acceptor2: P1 P2 A2-Y
- Acceptor3: P2 A2-Y

Paxos: Race Condition

- Acceptor1: P1 A1-X P3
- Acceptor2: P1 P2 A1-X P3 A2-Y P4
- Acceptor3: P2 A2-Y P4

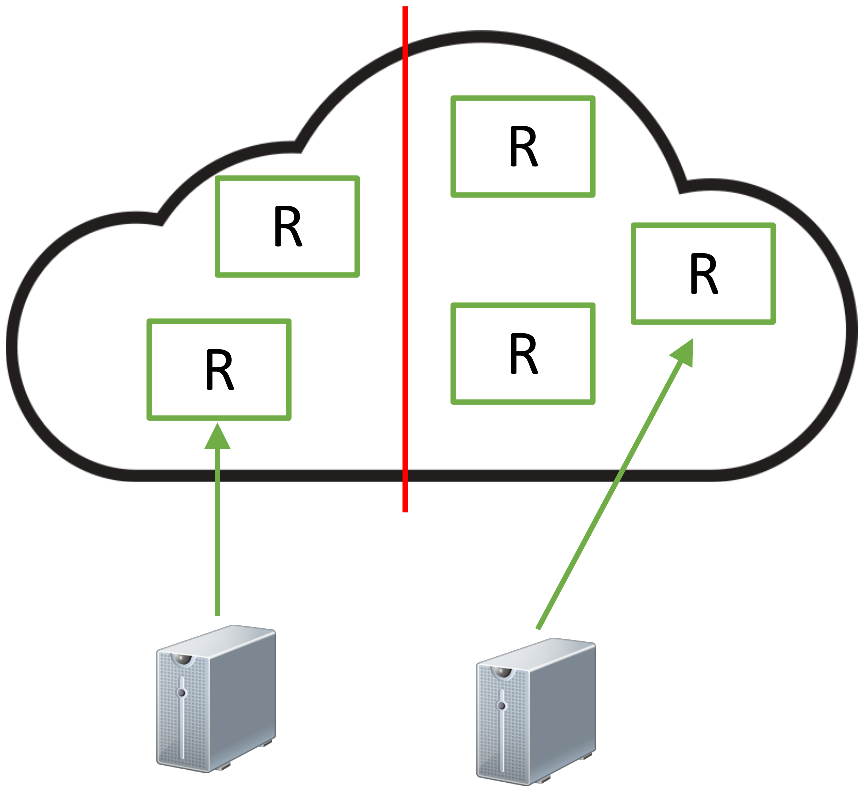
CAP Theorem

- Pick any two: consistency, availability, and partition-tolerance
  - In practice, choose between CP and AP



PACELC

- If partition,
  - Choose availability vs. consistency
- Else,
  - Choose latency vs. consistency



Consistency Spectrum

Eventual | **Read-after-write** | **Causal** | **Sequential** | **Linearizability**

→ *Consistency*

→ *Ease of programming*

→ *Latency*

← *Availability*

Good Luck!