Welcome to CSCI 7000-001 Lec 9 (Feb 11)!

State Replication

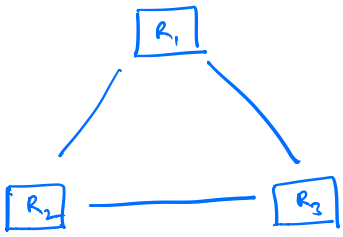
→ Applications often replicate state across distributed machines

→ Each machine is said to be a **replica** of the state

→ Goals of replication : * **Low latency** access to data
* **Resilience** (data loss prevention)

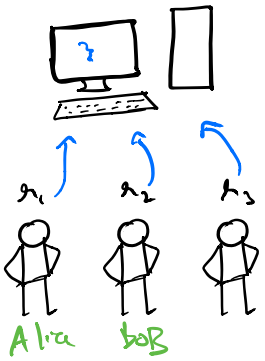
→ A non-goal of replication: Introduce new behaviours into the application

→ Ideally, we want a distributed system to behave as if it is a **Single Computational unit.**



ideally ⟺ R

→ Execution history of a single computational node is Linearizable.

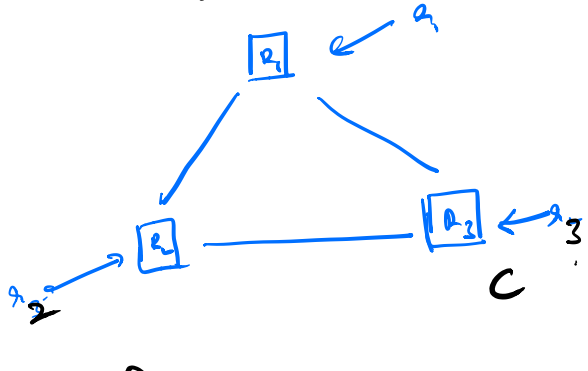


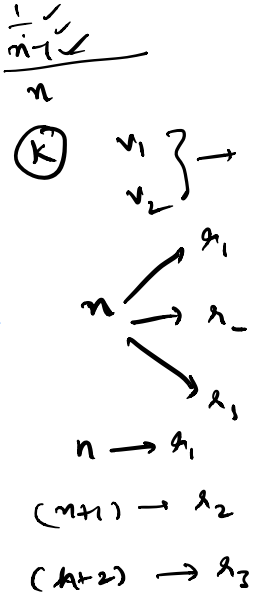
→ Sequential execution of requests

→ Consistent with an external notion of time.

Linearizability

→ How do we achieve Linearizable state replication in an asynchronous distributed system?





→ Paxos implementation

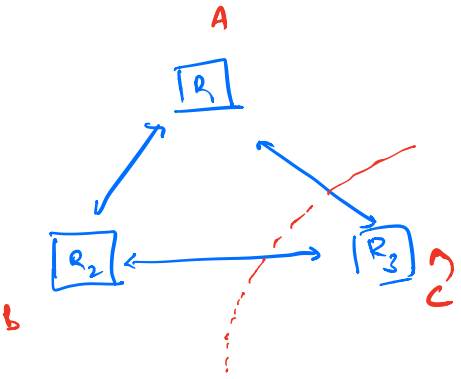
→ Strong Consistency under Asynchronous System

CAP

PACELC

else Latency vs Strong Consistency

⇒ high latency

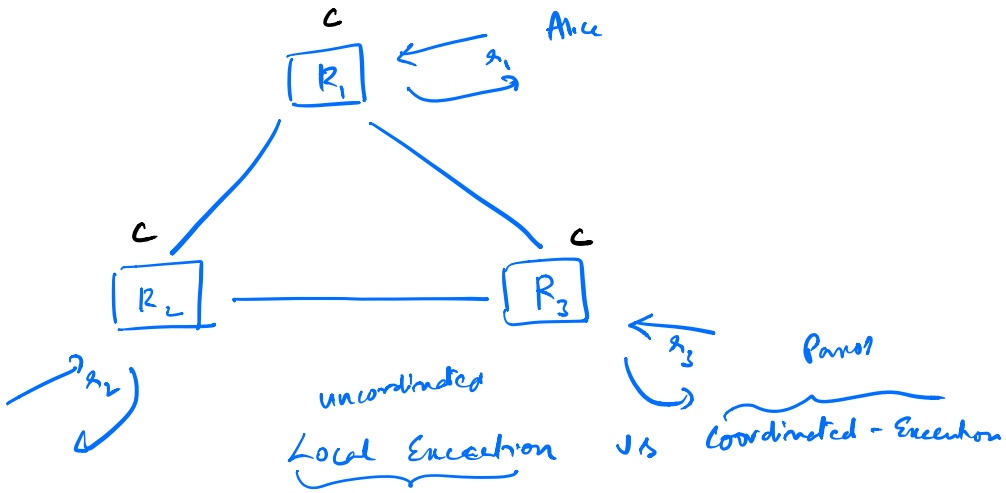


Practically not available

⇒ Prepare, Promise, Propose, Accept } Idealized Paxos

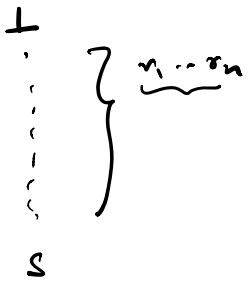
4 RTTs

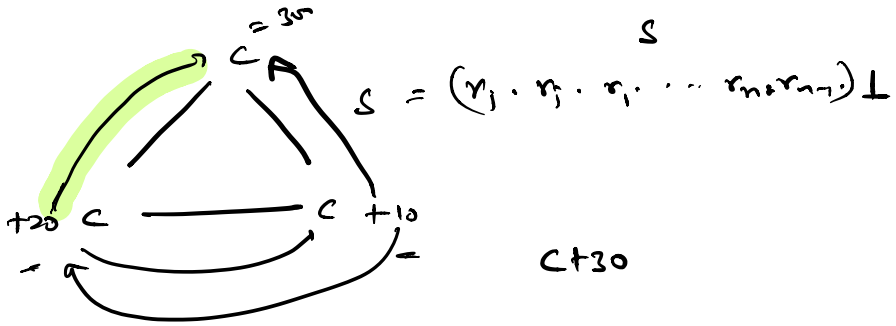
⇒ Requests getting repeatedly denied

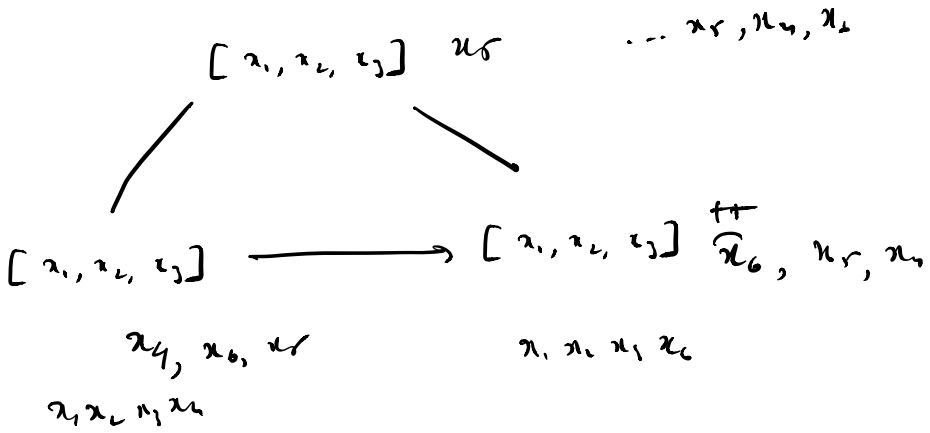


→ Replica states might diverge.

→ No Linearizability / Serializability







$$op_1 \cdot op_2 = op_2 \cdot op_1$$

CRDTs 2011

Conflict-free Replicated Data Types

- CvRDTs
- CmRDTs
  Commutative

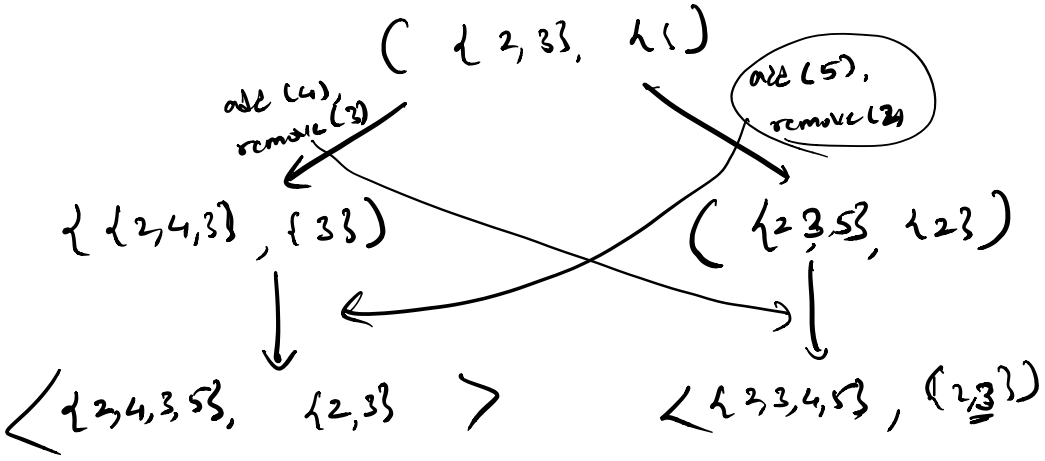
→ Counter is naturally a CmRDT (CRDT)

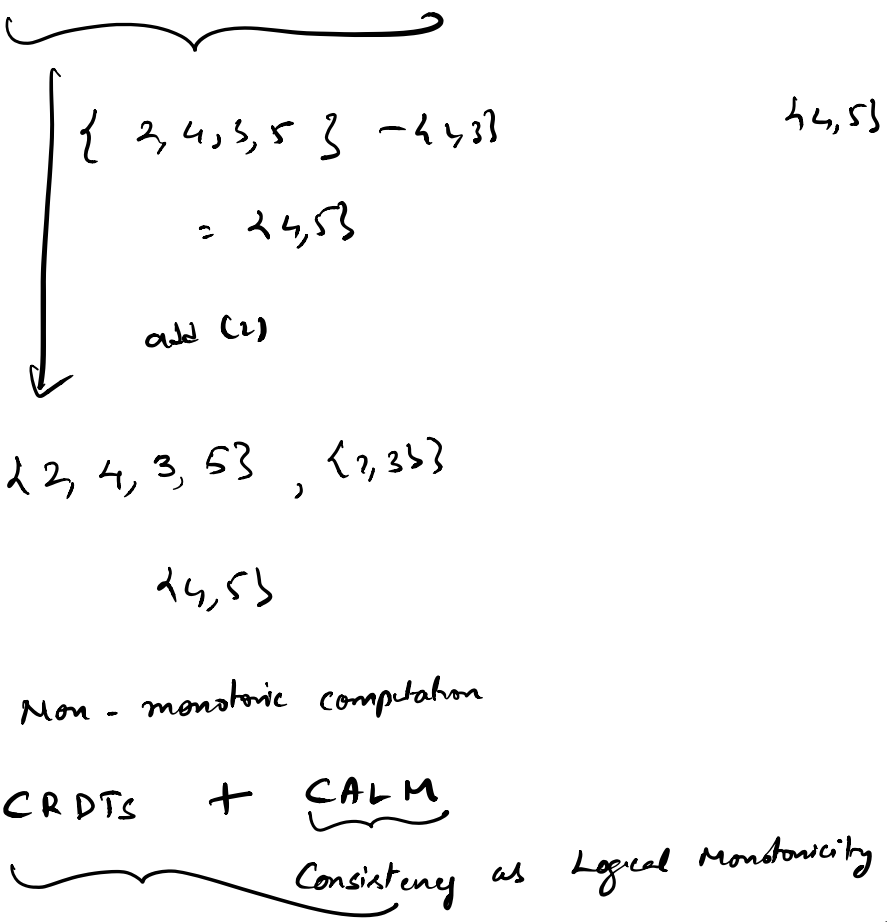
→ Set data type.

- → add(x) → remove(x) = ∅
- → remove(x) → add(x) = {x}

→ ({added elts}, {removed elts})

{removed elts} = tombstones





→ Non-monotonic computation

CRDTs + CALM

Consistency as Logical Monotonicity

Application working correctly under weak consistency