

# Course evals

Please take a few minutes to fill out the course evaluations:

<https://uw.iasystem.org/survey/309412>

And thank you all for your hard work this quarter!



# Announcements

Almost done!

- HW6 due Monday
- Lab 5 and Final report due on Thursday of finals week **No late days**

# References

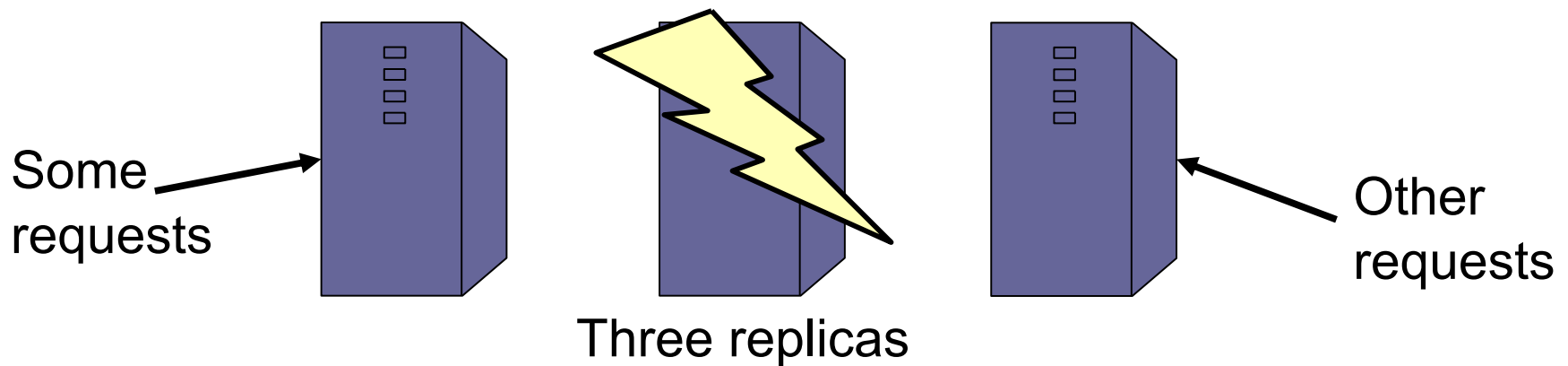
- Ullman Book Chapter 20.6
- Database management systems.  
Ramakrishnan and Gehrke.  
Third Ed. Chapter 22.11

# Outline

- Goals of replication
- Three types of replication
  - **Synchronous** (aka eager) replication
  - **Asynchronous** (aka lazy) replication
  - Two-tier replication

# Goals of Replication

- **Goal 1: consistency.** Always read latest update
- **Goal 2: availability.** Every request → a response
- **Goal 3: performance.** Fast read/writes



# Discussion: NoSQL

## New problem in the early 2000's

- Startup company launches Website backed up by MySQL, works fine with 50 users
- Suddenly, they are successful and have 1M users
- MySQL cannot keep up

# Discussion: NoSQL

## New problem in the early 2000's

- Startup company launches Website backed up by MySQL, works fine with 50 users
- Suddenly, they are successful and have 1M users
- MySQL cannot keep up

## NoSQL:

- Distributed database (replication, partition)
- Give up strong consistency in favor of availability and performance (as we'll see discuss next)



# Discussion: NoSQL

## New problem in the early 2000's


- Startup company launches Website backed up by MySQL, works fine with 50 users
- Suddenly, they are successful and have 1M users
- MySQL cannot keep up

## NoSQL:

- Distributed database (replication, partition)
- Give up strong consistency in favor of availability and performance (as we'll see discuss next)

Today: strong consistency is standard requirement

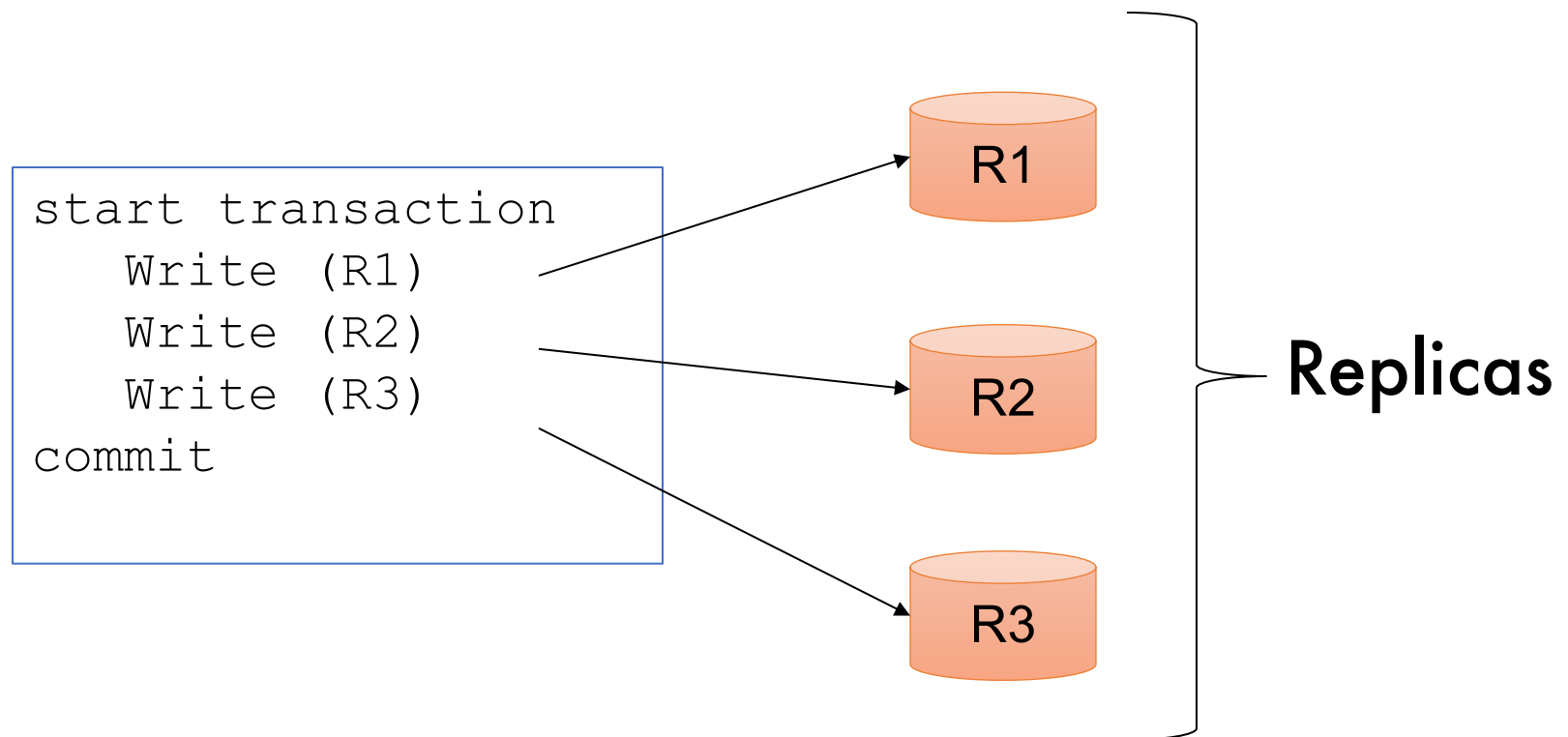
# Types of Replication

	Master	Group
Synchronous		
Asynchronous		

# Synchronous Replication

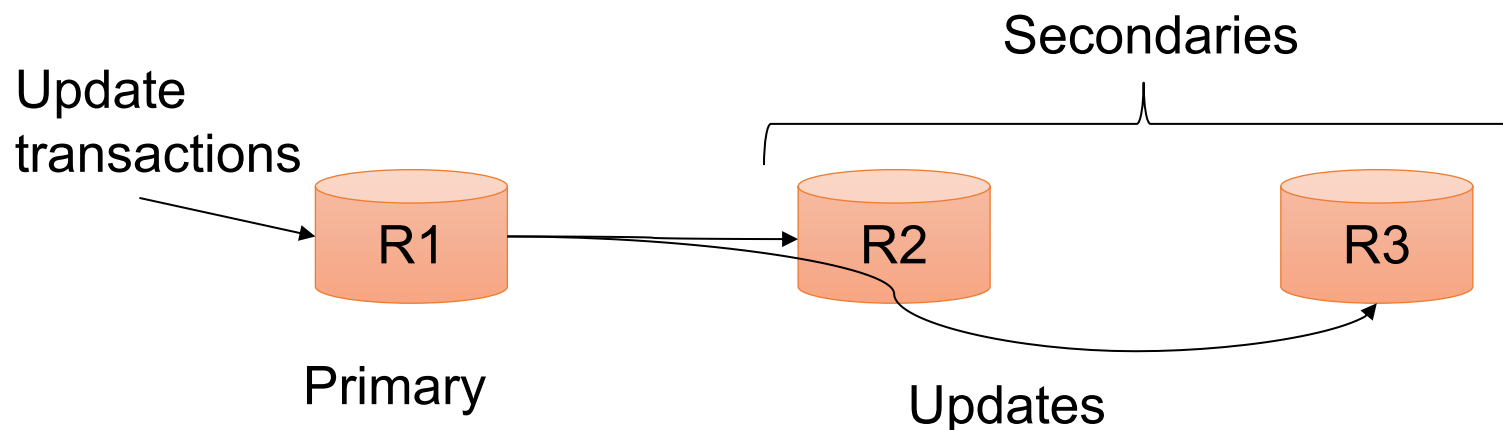
- Also called **eager replication**
- All updates are applied to all replicas (or to a majority) as part of a single transaction (need two phase commit)
- Transactions must acquire **global locks**
  - Nobody can read while we synchronize the replicas
- Main goal: as if there was only one copy
  - Maintain **consistency**
  - Maintain **one-copy serializability**
  - I.e., execution of transactions has same effect as an execution on a non-replicated db

# Synchronous Replication



# Synchronous Master Replication

- **One master for each object holds primary copy**
  - The “Master” is also called “Primary”
  - To update object, transaction must acquire a lock at the master
  - Lock at the master is global lock
- **Master propagates updates to replicas synchronously**
  - Updates propagate as part of the same distributed transaction
  - Need to run 2PC at the end



# Crash Failures

- What happens when a secondary crashes?

# Crash Failures

- What happens when a secondary crashes?
  - Nothing happens
  - When secondary recovers, it catches up

# Crash Failures

- What happens when a secondary crashes?
  - Nothing happens
  - When secondary recovers, it catches up
- What happens when the master/primary fails?



# Crash Failures

- **What happens when a secondary crashes?**
  - Nothing happens
  - When secondary recovers, it catches up
- **What happens when the master/primary fails?**
  - Blocking would hurt availability
  - Must choose a new primary: run election



# Network Failures

- **Network failures can cause trouble...**
  - Secondaries think that primary failed
  - Secondaries elect a new primary
  - But primary can still be running
  - Now have two primaries!

# Majority Consensus

- To avoid problem, only majority partition can continue processing at any time
- In general,
  - Whenever a replica fails or recovers...
  - ...a set of communicating replicas must determine...
  - ...whether they have a majority before they can continue

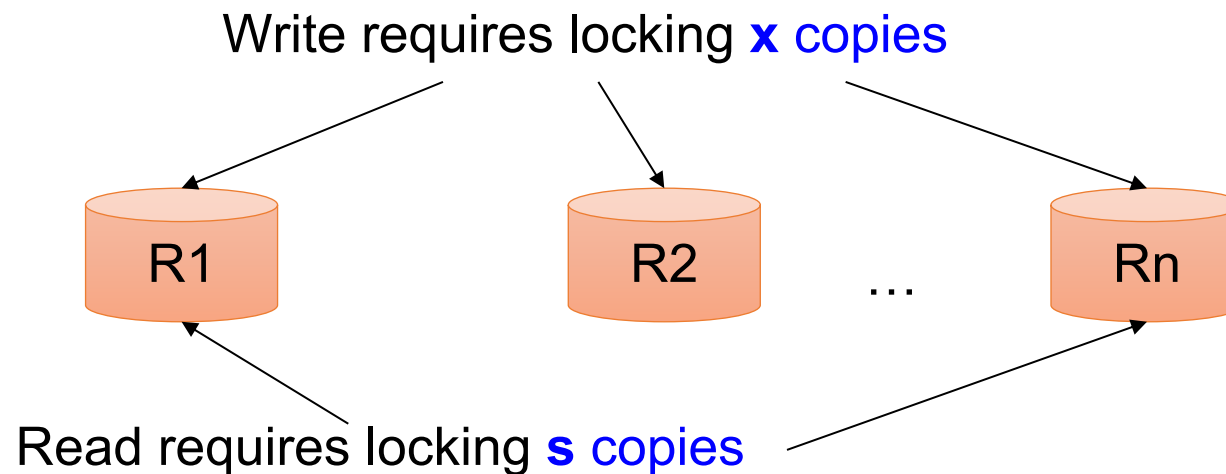
# Types of Replication

	Master	Group
Synchronous		
Asynchronous		

# Synchronous Group Replication

## ▪ Master-less

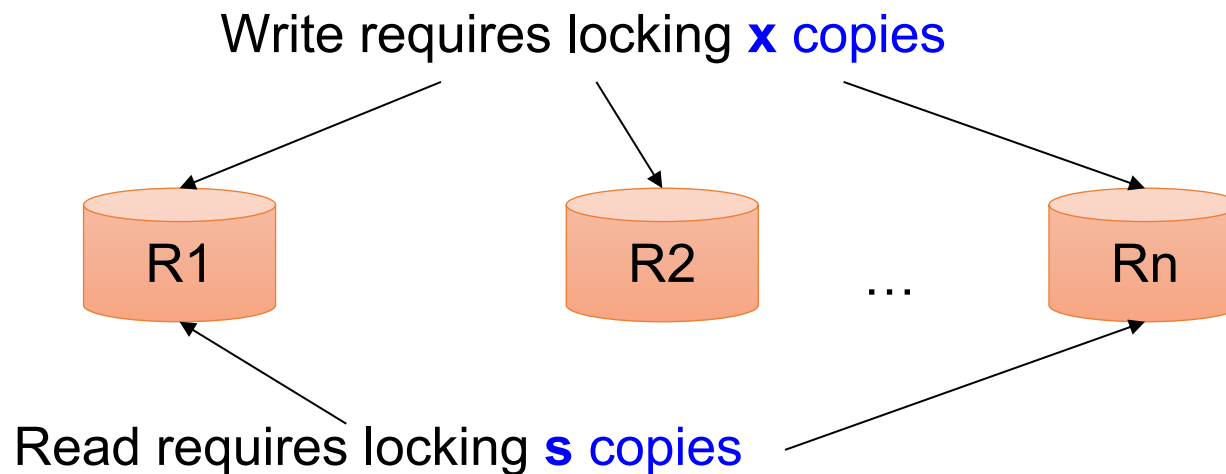
- Any node can initiate a transaction!
- Need to gather a number of nodes that agree on a particular transaction
- Each copy has its own lock



# Synchronous Group Replication

## ■ With $n$ copies

- Exclusive lock on  $x$  copies is global exclusive lock
- Shared lock on  $s$  copies is global shared lock
- Must have:  $x > n/2$  and  $s + x > n$
- Version numbers serve to identify current copy



# Synchronous Group Replication

## ▪ Majority locking

- $s = x = \lceil (n+1)/2 \rceil$  eg: 11 nodes: need 6 locked
- Usually not attractive because reads are slowed down

## ▪ Read-locks-one, write-locks-all




- $s=1$  and  $x = n$ , high read performance
- Reads are very fast

# Synchronous Replication Properties

- Favours **consistency** over availability
  - Only majority partition can process requests
  - There appears to be a single copy of the db
- **High runtime overhead**
  - Must lock and update at least majority of replicas
  - Two-phase commit
  - Runs at pace of slowest replica in quorum
  - So overall system is now slower
  - Higher deadlock rate (transactions take longer)



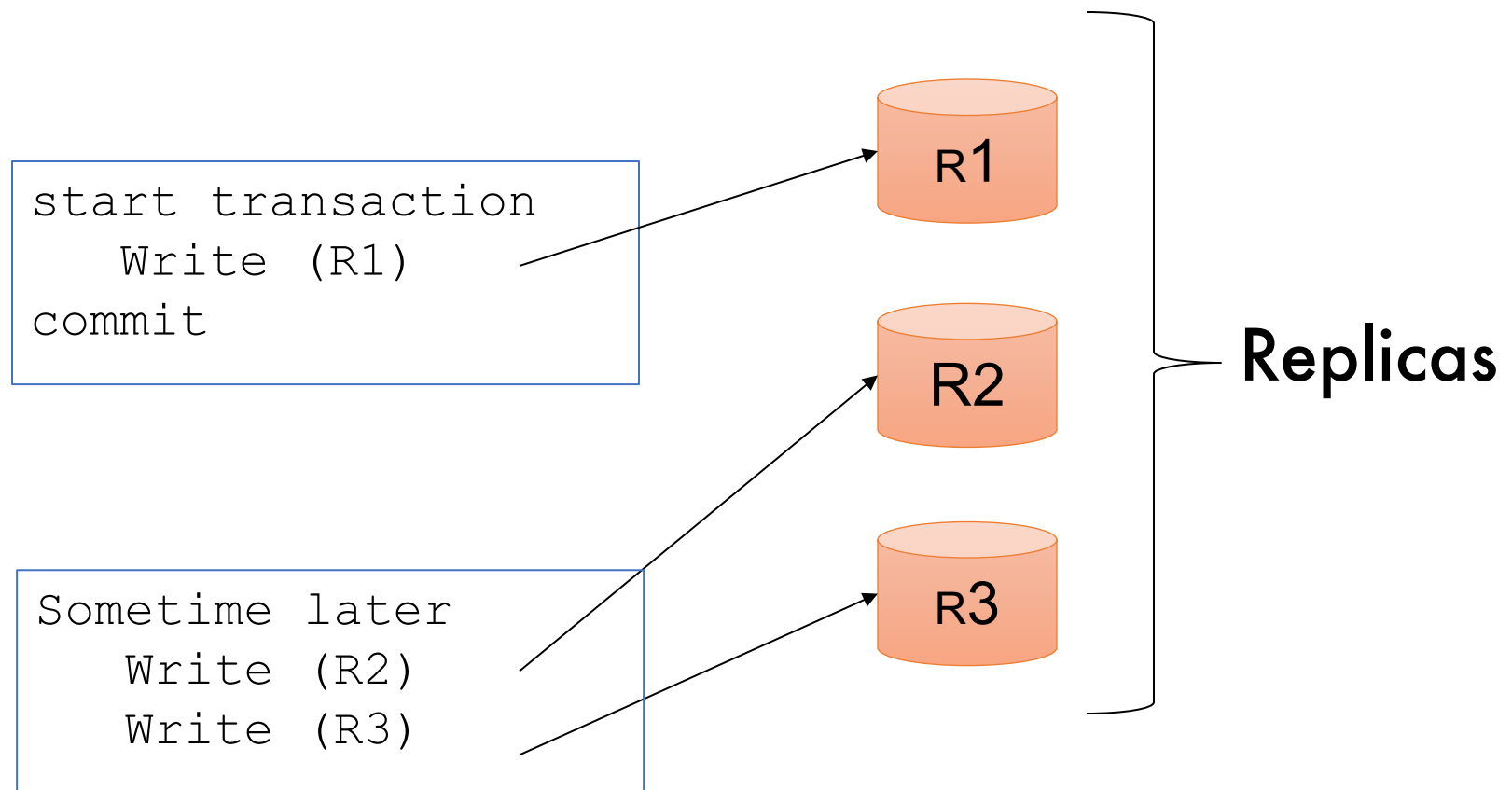
# Types of Replication

	Master	Group
Synchronous		
Asynchronous		

# Asynchronous Replication

- Also called **lazy replication**
- Also called **optimistic replication**
- Main goals: availability and performance
- Approach
  - One replica updated by original transaction
  - Updates propagate asynchronously to other replicas

# Asynchronous Replication



# Asynchronous Master Replication

## One master holds primary copy

- Transactions update primary copy
- Master asynchronously propagates updates to replicas, which process them in same order  
E.g. through **log shipping**
- Ensures single-copy serializability

## What happens when master/primary fails?

- Can lose most recent transactions when primary fails!
- After electing a new primary, secondaries must agree who is most up-to-date

# Discussion: Log Shipping

## A general problem:

- A master operates on a database
- The DB needs to be replicated to one or several replicas (e.g. hot stand-by databases)

# Discussion: Log Shipping

## A general problem:

- A master operates on a database
- The DB needs to be replicated to one or several replicas (e.g. hot stand-by databases)
- Log Shipping Technique

# Discussion: Log Shipping

## A general problem:

- A master operates on a database
- The DB needs to be replicated to one or several replicas (e.g. hot stand-by databases)
- Log Shipping Technique:
  - Master node ships the tail of the log to the replicas  
E.g. when it flushes the log tail to disk
  - Replicas REDO the log; this is very efficient
  - Need very little systems development: we create the log anyway, and we have the REDO function anyway





# Discussion: Log Shipping

## A general problem:

- A master operates on a database
- The DB needs to be replicated to one or several replicas (e.g. hot stand-by databases)
- Log Shipping Technique:
  - Master node ships the tail of the log to the replicas  
E.g. when it flushes the log tail to disk
  - Replicas REDO the log; this is very efficient
  - Need very little systems development: we create the log anyway, and we have the REDO function anyway
  - Complications due to the need to “remove” updates of active transactions (they may later abort)

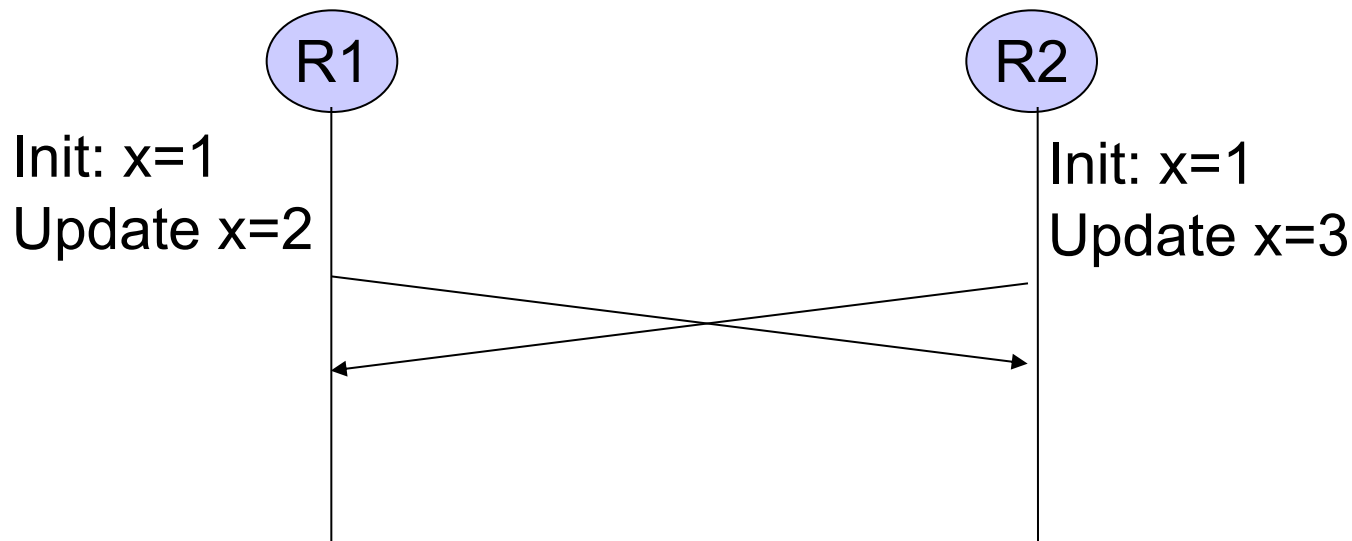


# Types of Replication

	Master	Group
Synchronous		
Asynchronous		

# Asynchronous Group Replication

- Also called **multi-master**
- Best scheme for availability
- **Cannot guarantee one-copy serializability!**



# Asynchronous Group Replication

- **Cannot guarantee one-copy serializability!**
- **Instead guarantee convergence**
  - Db state does not reflect any serial execution
  - But all replicas have the same state
- Called “Eventual Consistency” = if the DB stops operations, then eventually all copies are equal

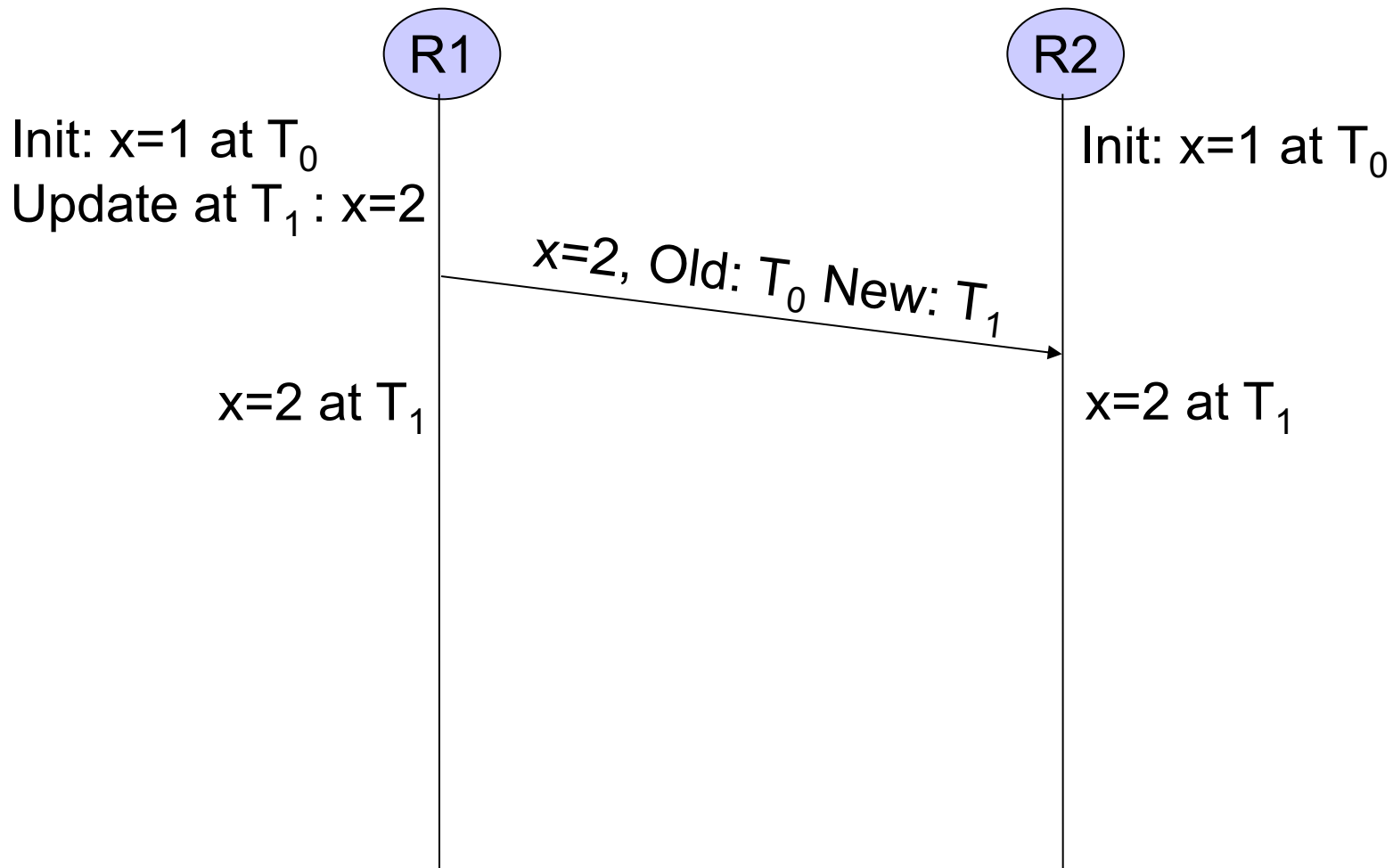
# Asynchronous Group Replication

- **Cannot guarantee one-copy serializability!**
- **Instead guarantee convergence**
  - Db state does not reflect any serial execution
  - But all replicas have the same state
- Called “Eventual Consistency” = if the DB stops operations, then eventually all copies are equal
- Detect conflicts and reconcile replica states

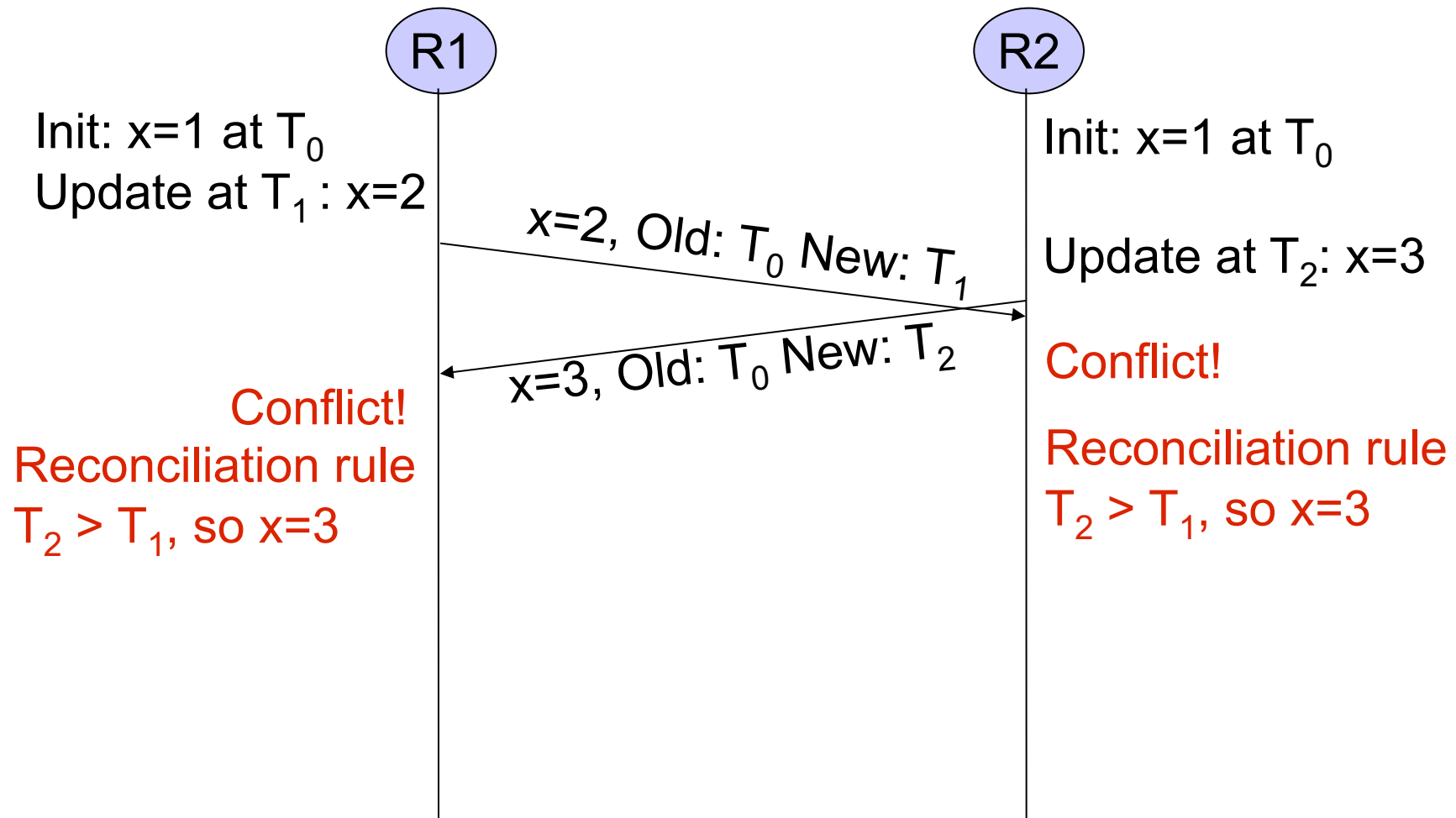
# Asynchronous Group Replication

- **Cannot guarantee one-copy serializability!**
- **Instead guarantee convergence**
  - Db state does not reflect any serial execution
  - But all replicas have the same state
- Called “Eventual Consistency” = if the DB stops operations, then eventually all copies are equal
- Detect conflicts and reconcile replica states
- Reconciliation techniques:
  - Most recent timestamp wins
  - Site A wins over site B
  - But also: user-defined rules, or even manual

# Detecting Conflicts Using Timestamps



# Detecting Conflicts Using Timestamps



# Conclusion

- Many innovations recently in
  - Big data analytics
  - Transaction processing at very large scale
- Many more problems remain open
- This course teaches foundations
- Innovate with an open mind!