CONSISTENCY AND REPLICATION

Distributed Systems

4. Sem BSc Informatics

IMC FH Krems

LECTURE OUTLINE

- Introduction
- Consistency models and protocols
  - Data-centric
  - Client-centric
- Replica management

INTRODUCTION

- Performance and reliability requirements often mean that **data replication** has to be introduced in some way.
- **Performance**: related to scaling requirements.
  - Scaling by size: number of processes that need access to data increases. If data would be managed by a single server, it would become a bottleneck.
  - Scaling geographically: placing copies of the data nearby to reduce latency effects.
- **Reliability**: if one replica is not available, a process can switch to another replica. If data is corrupted, the probability of restoring data successfully increases.
- Main problem introduced by replication: **inconsistency.**
- In this lecture, we will focus on replication as performance measure.
  - Next lecture: fault tolerance.

INTRODUCTION

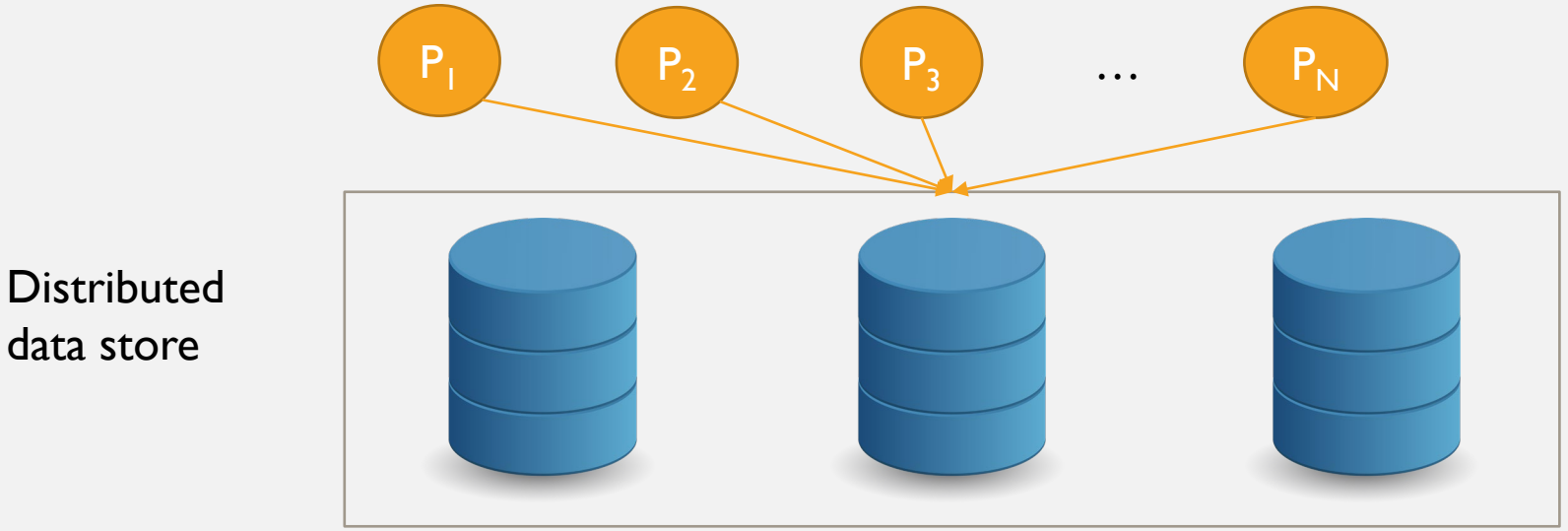
- What is the price to pay for replication for performance improvements?
- If data is always read, then we have the maximum performance gain.
- If data is updated, then we face a tradeoff (consistency vs. performance):
  - If we want data to be **always** in sync (strong consistence), we need to propagate updates as **atomic** operations (for instance, using distributed transactions).
    - Global synchronization needed → This can **degrade** performance.
  - If we don't propagate updates immediately, some replicas may be in different states than others.
- Consequence: in many situations consistency requirements will have to be relaxed.
- Enter **consistency models.**

LECTURE OUTLINE

- Introduction
- **Consistency models and protocols**
  - Data-centric
  - Client-centric
- Replica management

CONSISTENCY MODELS

- A **consistency model** is a contract between a **data store** and a process



- Different consistency models guarantee different levels of consistency.
- "Right" model depends on application requirements.

CONSISTENCY MODELS

Continuous consistency models

- Based on measurements of replica deviation by (numerical) data differences.
- Define an upper bound on absolute deviation.
  - E.g. a replica is not allowed to deviate more than 0.5% from any other replica.
- How to measure these deviations?

a) Numerical deviations:

Numerical variable (e.g. stock price) can be measured in all replicas. The deviation of a replica with respect to another is the difference in value with that replica.

b) Staleness deviations:

Last update timestamp is compared in two replicas. Some deviation is tolerated depending on the application context (e.g. browser cache, weather app)

CONSISTENCY MODELS

Consistent ordering of operations

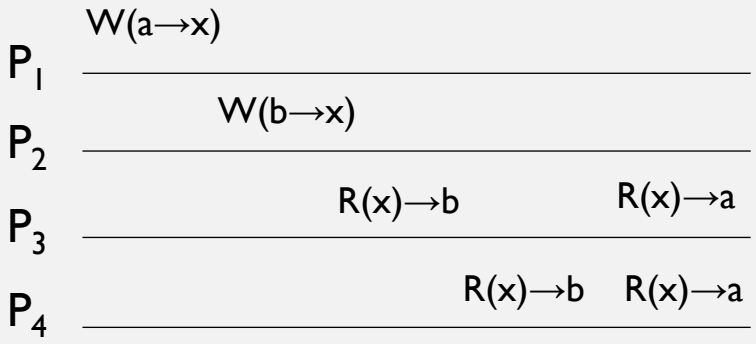
- Ensures that updates are executed in the same order in all replicas.
- Two consistency models depending on what events are exactly required to follow the same ordering.

1. Sequential consistency
2. Causal consistency

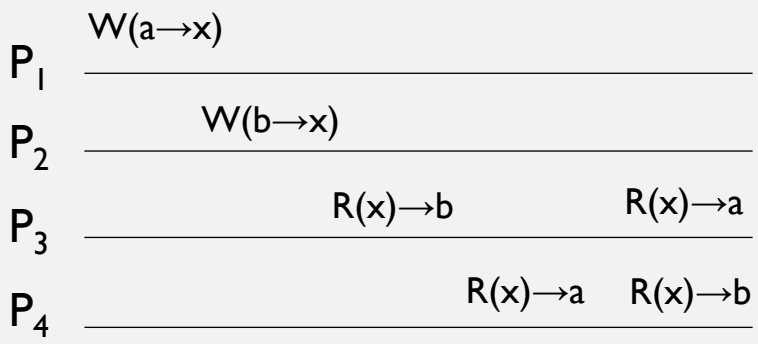
CONSISTENCY MODELS

1. Sequential consistency

- All processes are required to see *the same sequence of events* (Lamport, 1979).
- This sequence of events has to be compatible with the sequence of operations of each process individually.



(a) Sequentially consistent

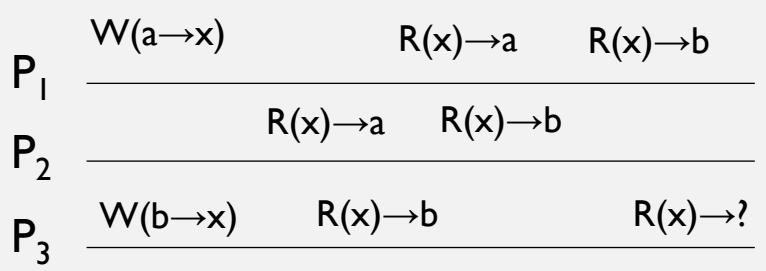


(b) Not sequentially consistent

CONSISTENCY MODELS

1. Sequential consistency

P_1	P_2	P_3
1. W(a→x)	1. R(x)	1. W(b→x)
2. R(x)	2. R(x)	2. R(x)
3. R(x)		3. R(x)



What has to be the value read by P_3 to satisfy sequential consistency?

CONSISTENCY MODELS

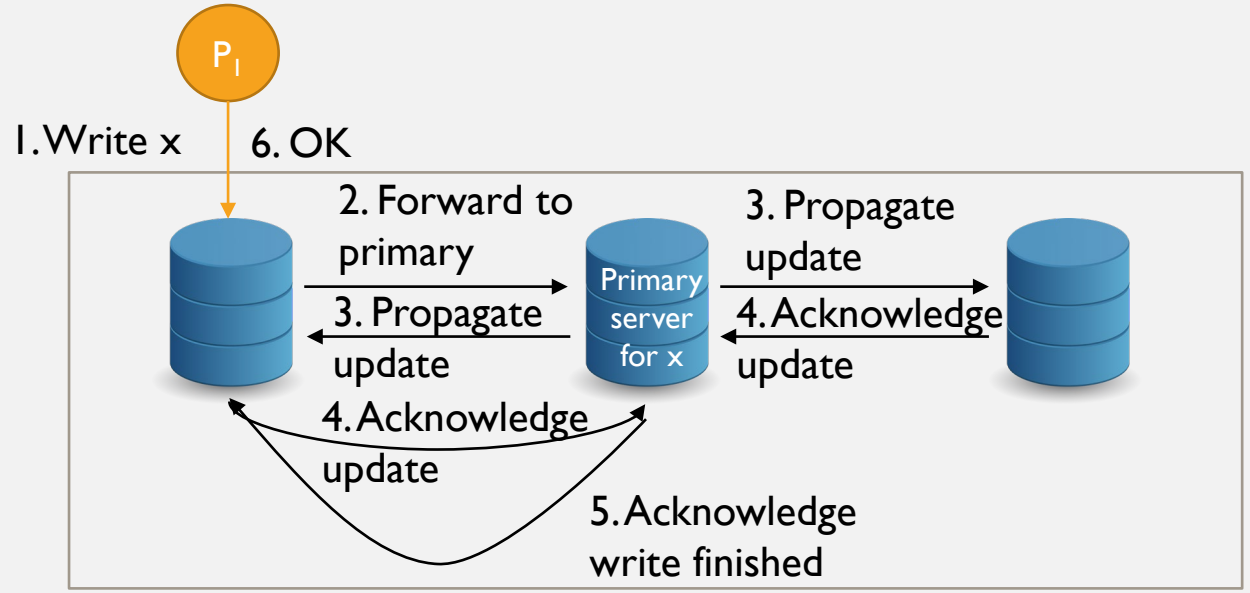
1. Sequential consistency

- How to implement it? → **primary** based protocols.
- Example: **remote write protocol.**
  - Each data item has an associated primary server.
  - Write requests are forwarded to the primary server.
  - Primary server then propagates update to replicas.
  - Replicas send acknowledgement.
  - Write operation finished.
- Why is sequential consistency satisfied?

CONSISTENCY MODELS

1. Sequential consistency

- Remote write protocol



Primary orders all write operations, so all processes see the same order of writes

CONSISTENCY MODELS

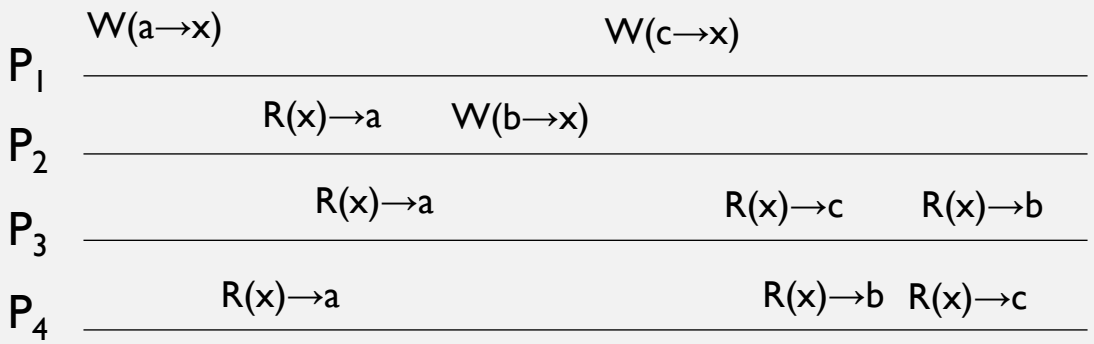
1. Sequential consistency

- Remote write protocol blocks for a relatively long time (until all replicas are updated)
- Local write protocol: primary is moved to local replica.
  - Client receives OK as soon as local replica is updated.
  - Blocking time is reduced.
- Active replication
  - Instead of propagating data, we propagate the write operations themselves.
  - Operations need to be carried out in the same order.
  - Coordinator (**sequencer**) is elected so that all replicas get the operations in the right order.

CONSISTENCY MODELS

2. Causal consistency

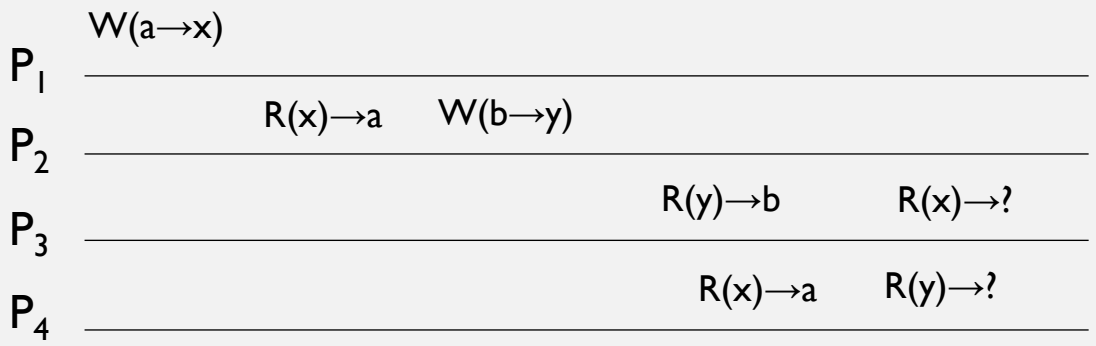
- Weakening of sequential consistency so that only **causally related** events are required to be seen in the same sequence by all processes.



- $W_2(b \rightarrow x)$ is causally related to $W_1(a \rightarrow x)$
- $W_2(b \rightarrow x)$ and $W_1(c \rightarrow x)$ are concurrent
- Is this ordering sequentially consistent?

CONSISTENCY MODELS

2. Causal consistency: another example (values are initialized to null)



- What should be the value of x read by P_3 to satisfy causal consistency?
- What should be the value of y read by P_4 to satisfy causal consistency?

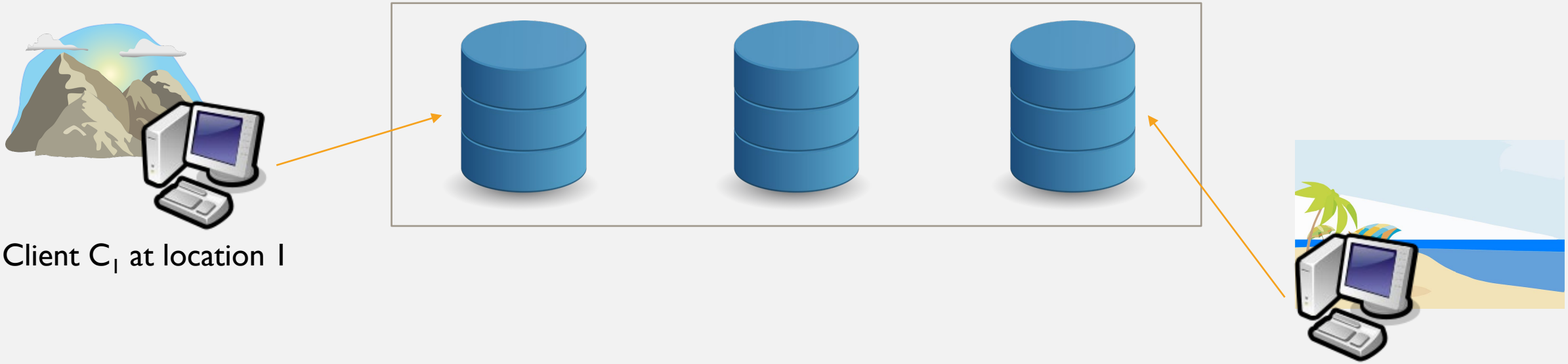
CONSISTENCY MODELS

3. Eventual consistency

- Weak (but popular) consistency model.
- Assume only a few processes update a data store, but many processes read from the data store.
- In general: only the owner/authorized processes are allowed to make updates.
- Eventual consistency requires that all replicas eventually converge to the same copy of the data.
- Updates are gradually propagated to all replicas.
- Examples:
  - DNS.
  - Master/slave replication on relational DBs like MariaDB.
  - Amazon Dynamo.

CONSISTENCY MODELS

Client-centric consistency models



Client C_1 at location 1

Client C_2 at location 2

CONSISTENCY MODELS

Client-centric consistency models

- User in a given location is attached to a replica and performs some write and read operations.
- When moving to a new location, the user is attached to another replica.
  - New replica may also be assigned to the user.
- Main idea: bring the new replica up to date depending on the operations the client has performed before.
- Different consistency models provide different guarantees as to what data is seen by the user when interacting with the new replica.

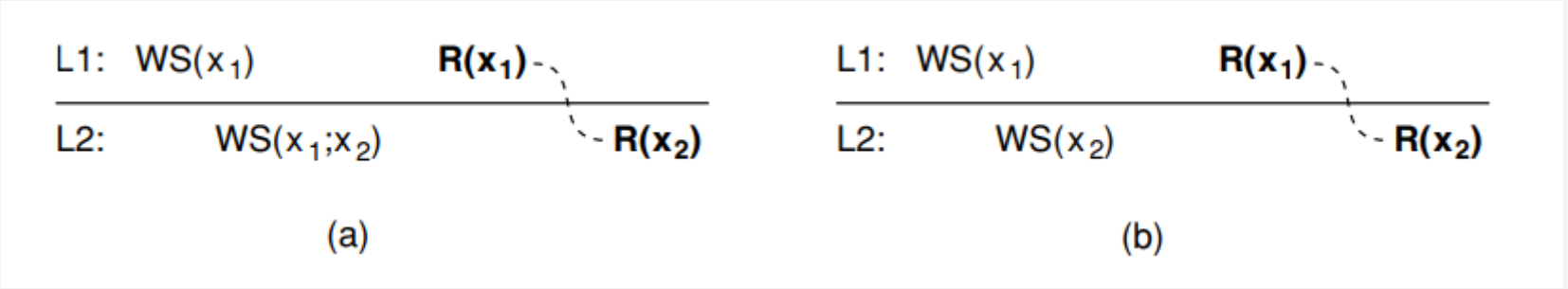
CONSISTENCY MODELS

Client-centric consistency models

1. **Monotonic reads:** If a process reads the value of a data item, any successive read operation on that item *by that process* will always return the same value or a more recent one
2. **Monotonic writes:** A write operation by a process on a data item is completed before any successive write operation on that data item *by the same process*
3. **Read your writes:** The effect of a write operation by a process on a data item will always be seen by a successive read operation on that data item *by the same process*
4. **Writes follow reads:** A write operation by a process on a data item following a previous read operation on that item *by the same process* is guaranteed to take place on the same or a more recent value of that item that was read

CONSISTENCY MODELS

Monotonic reads: If a process reads the value of a data item, any successive read operation on that item *by that process* will always return the same value or a more recent one.

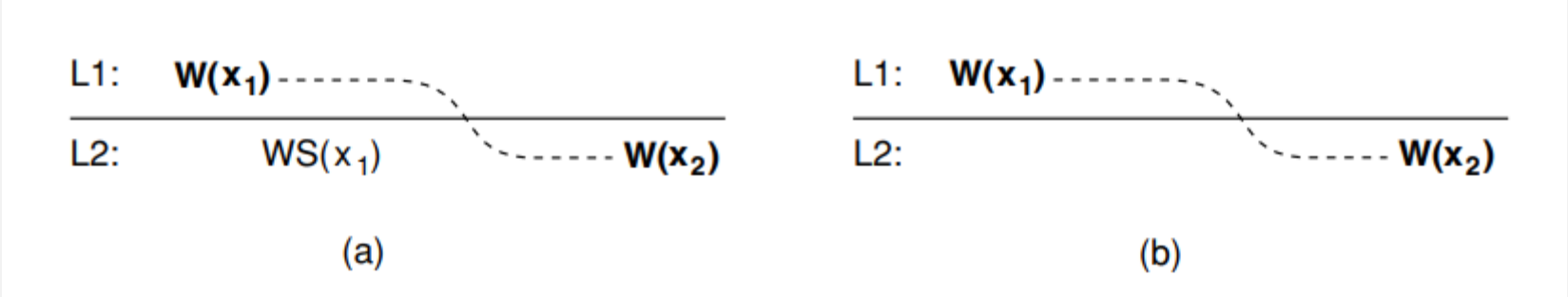


(a) To guarantee monotonic-reads, all write operations on x in L_1 should be propagated to L_2. We denote that by $WS(x_1; x_2)$

(b) Monotonic-read consistency is not guaranteed.

CONSISTENCY MODELS

Monotonic writes: A write operation by a process on a data item is completed before any successive write operation on that data item *by the same process*.

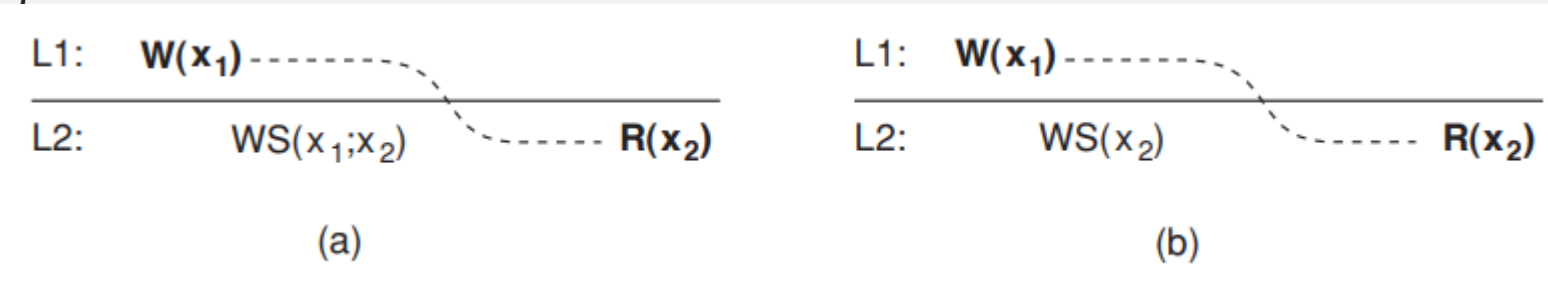


(a) Monotonic-writes consistency is guaranteed, since the write operation on L_1 is propagated to L_2 before the write operation.

(b) Monotonic-writes consistency is not guaranteed.

CONSISTENCY MODELS

Read your writes: The effect of a write operation by a process on a data item will always be seen by a successive read operation on that data item *by the same process*

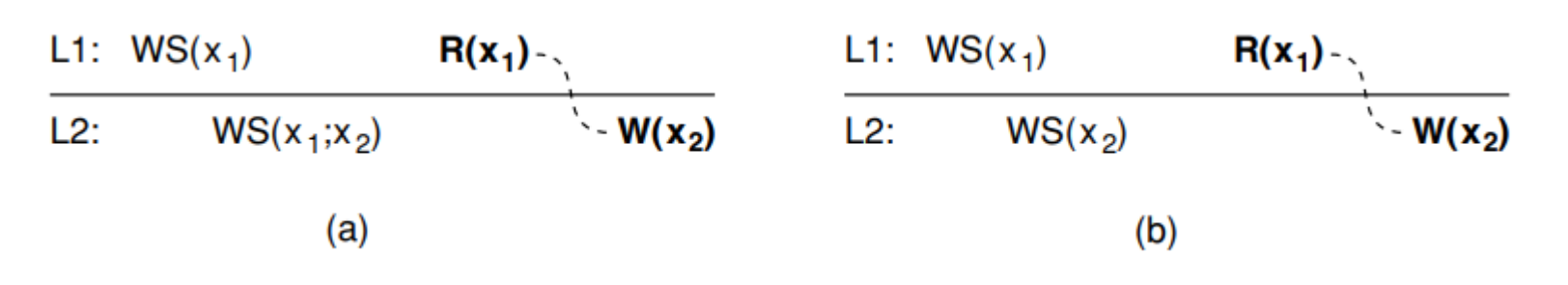


(a) Read operation is done at a different local copy. The effect of the write operation should be seen in subsequent reads.

(b) The effect of the write operation is not propagated to the new replica.

CONSISTENCY MODELS

Writes follow reads: A write operation by a process on a data item following a previous read operation on that item *by the same process* is guaranteed to take place on the same or a more recent value of that item that was read



(a) The read value of $R(x_1)$ is propagated to L_2 before performing the write operation.

(b) Not guaranteed that the write operation occurs on the most recently seen value of x.

LECTURE OUTLINE

- Introduction
- Consistency models and protocols
  - Data-centric
  - Client-centric
- **Replica management**

REPLICA MANAGEMENT

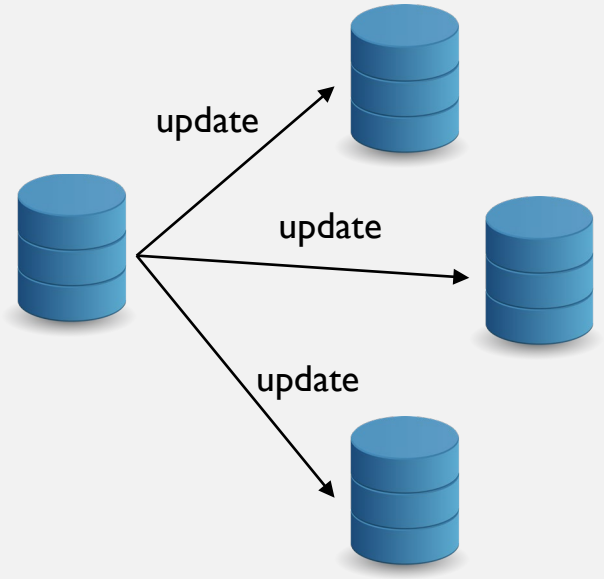
- How to distribute content between replicas?

1. Propagate notifications of content updates.
   - A part of the data is *invalidated*.
   - Requests on invalidated data will trigger fetching of updated data.
2. Propagate the updated data.
   - Directly send the updated data to the other replicas.
3. Propagate the update operations to other replicas.
   - The operations themselves are sent and executed at the other replicas.
4. Who initiates propagation?

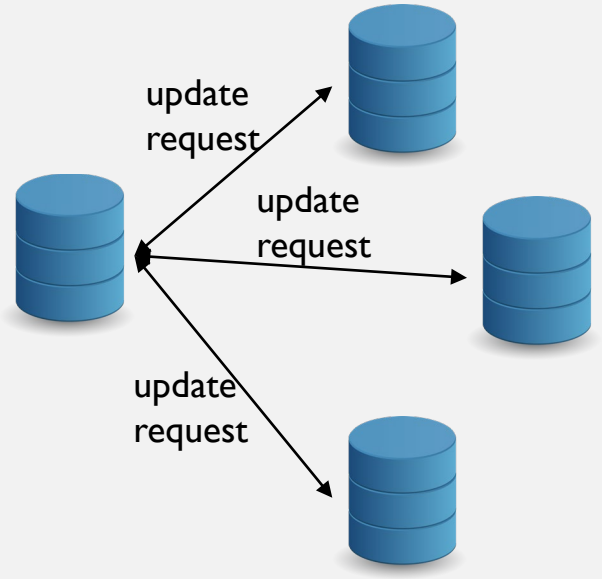
REPLICA MANAGEMENT

Push vs pull protocols

$\frac{read}{update}$ high



$\frac{read}{update}$ low



REPLICA MANAGEMENT

Push vs pull protocols

- **Push**: updates are actively sent to replicas.
  - Server-based protocols.
  - Efficient when many reads and fewer updates.
  - Broadcast/Multicast.
- **Pull**: replica asks for new updates.
  - Client-based protocols.
  - Efficient when many updates and fewer reads.
  - Point-to-point communication.