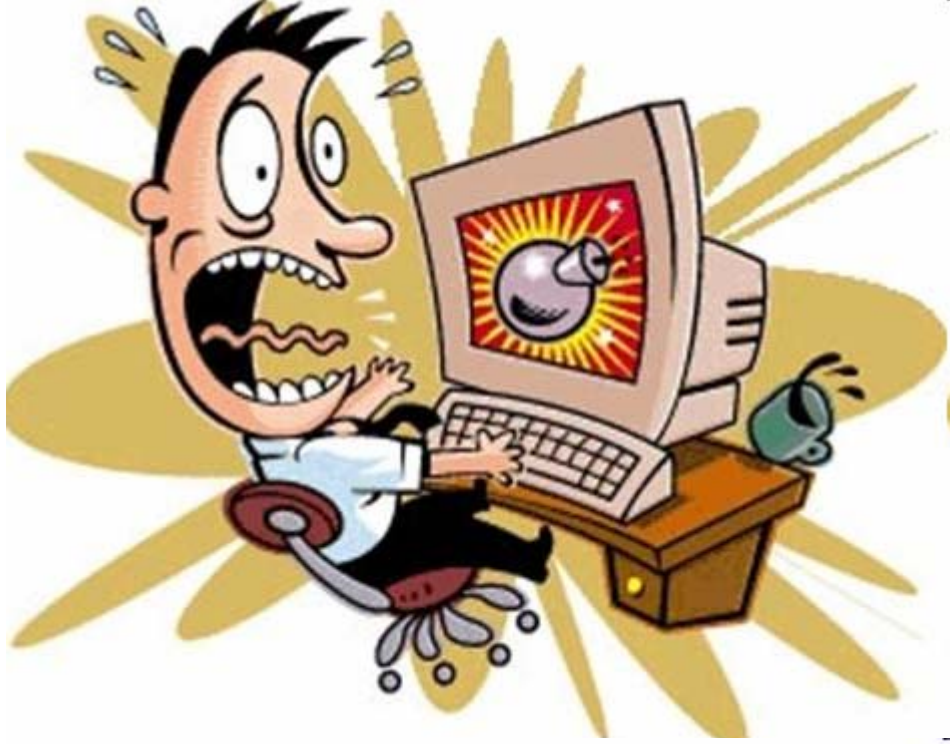
Sistem Terdistribusi 8

Replication & Consistency

Masalah Sistem

- Crash!
- Overload!

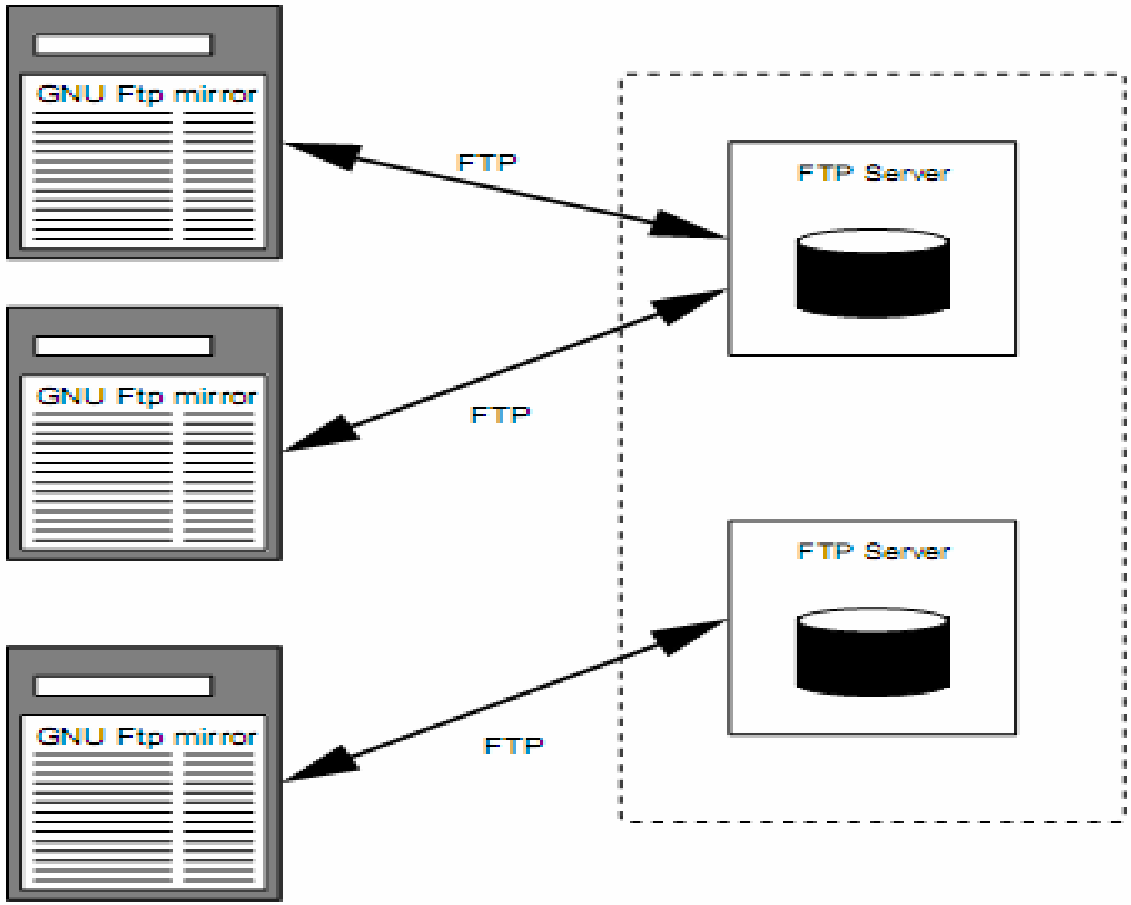


- Solusi?
  - REPLICATION

Replication

- Process sharing information to ensure **consistency** between **redundant resources** such as software or hardware components to improve **reliability, fault-tolerance,** or **accessibility**
- Make copies of services on **multiple machines**

Server Replication



Why Replication?

- Reliability: dapat diandalkan
- Performance enhancement: performa meningkat
- Scalability: dapat diperluas
- Fault tolerance
- Availability: tetap dapat diakses
  - Akibat dari server failure
  - Network partiiton / disconected network

Reasons detail

- Reliability:
  - If a replica crashes, system can continue working by switching to other replicas.
  - Avoid corrupted data:
    - can protect against a single, failing write operation.
- Improving Performance
  - Important for distributed systems over large geographical areas.
  - Divide the work over a number of servers.
  - Place data in the proximity of clients.
  - Caching

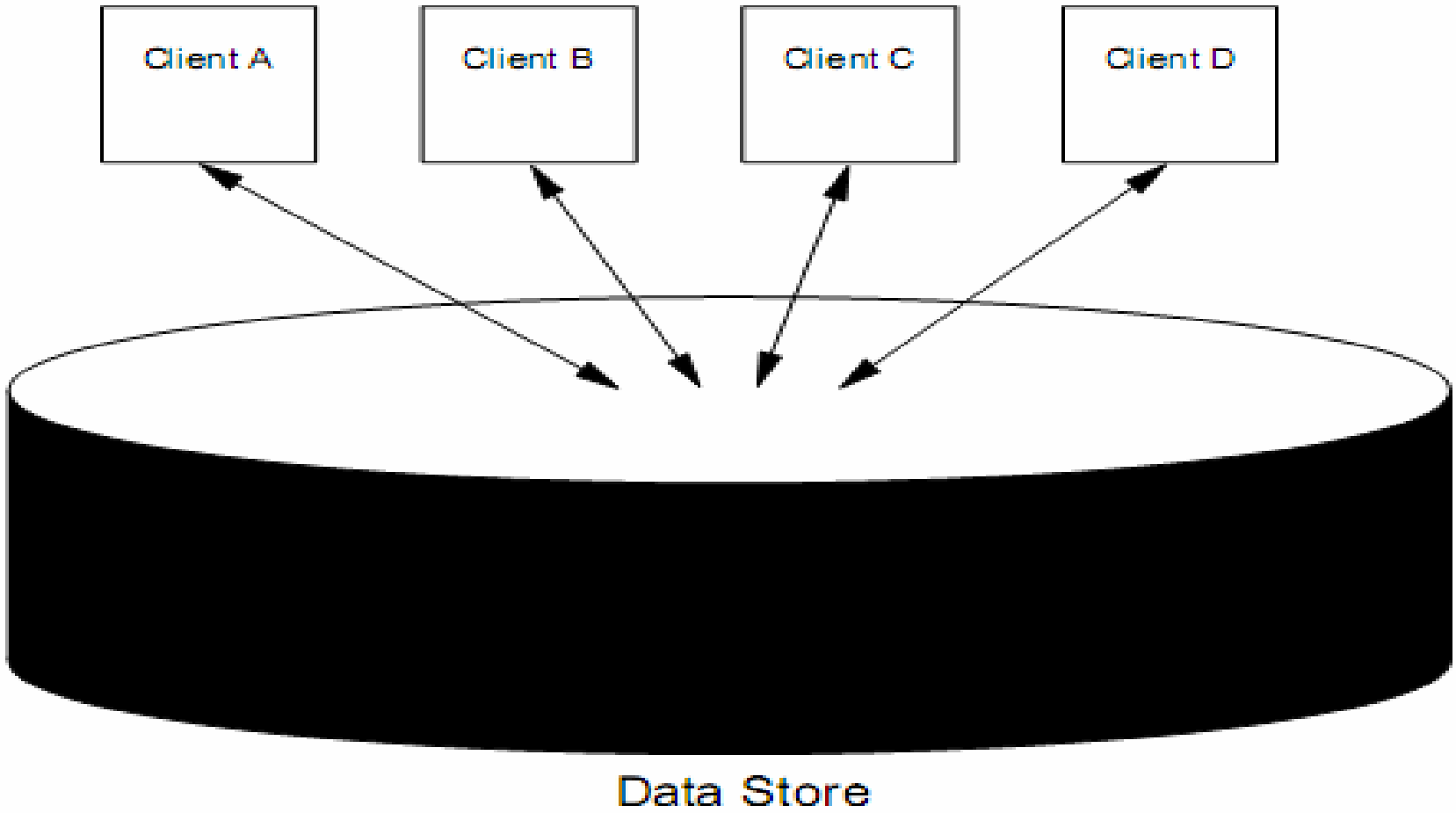
Issue

- Updates
  - Consistency?
    - Whenever a copy is modified, it becomes different from the rest.
  - Sinkronisasi dan Locking?
- Replica placement
  - How many?
  - Where?
- Redirection / Routing
  - Which replica should be used by client?

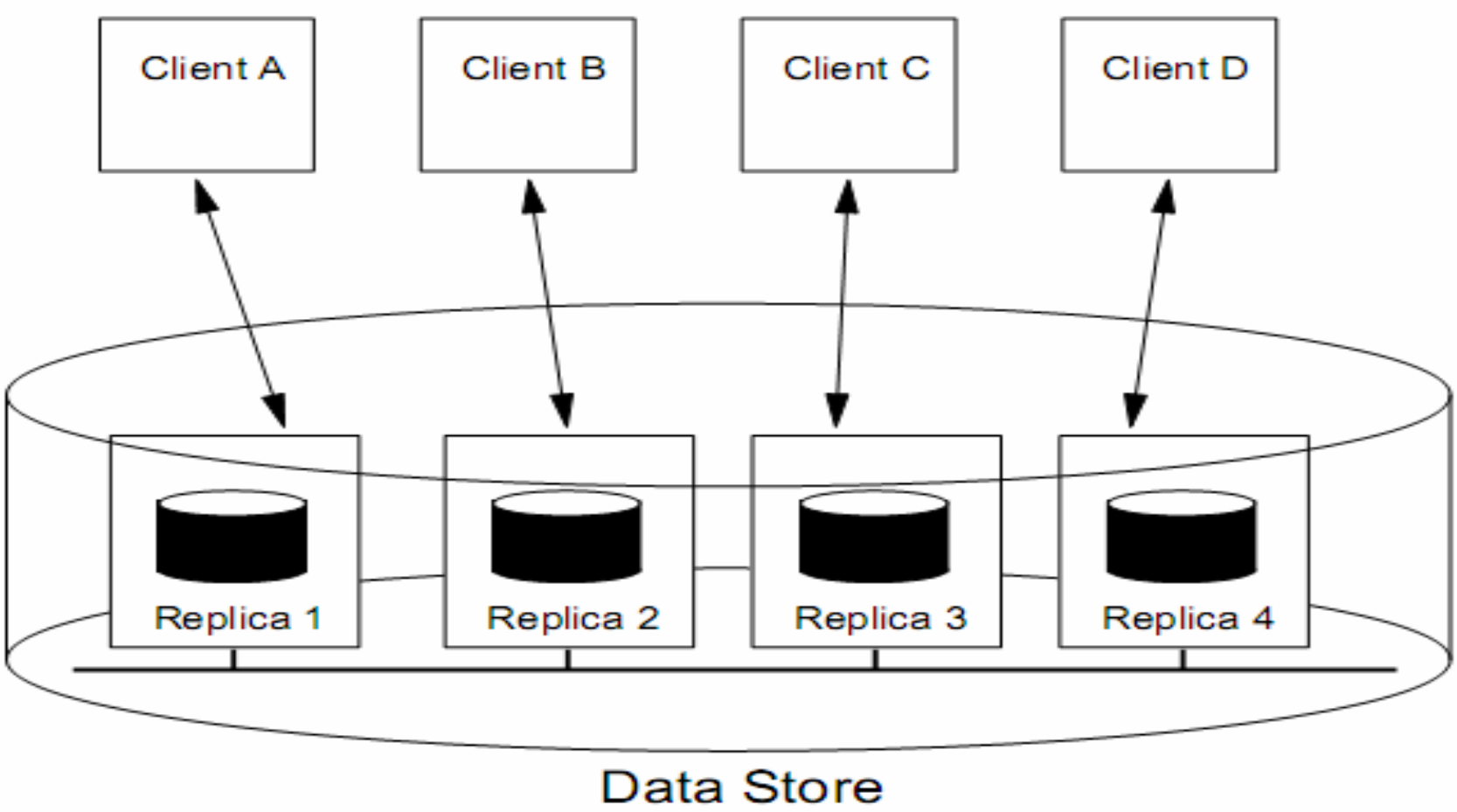
General Approach of Replication

- Update a single item in local replica
  - Atomically
- Replica propagates update to all of its other replicas
  - Periodically
- Receiving replica *merges* update with its own copy
  - Conflicting updates resolved arbitrarily to latest time-stamp

Model



The truth

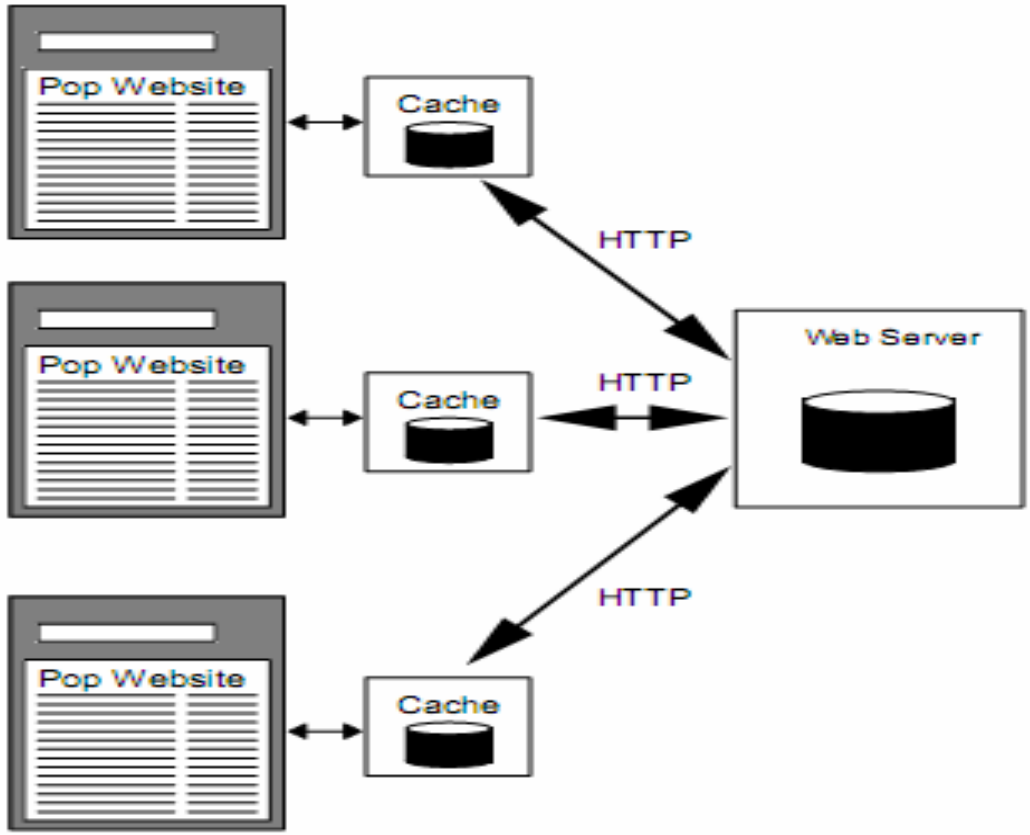

Client A
Client B
Client C
Client D
Replica 1
Replica 2
Replica 3
Replica 4
Data Store

Rules

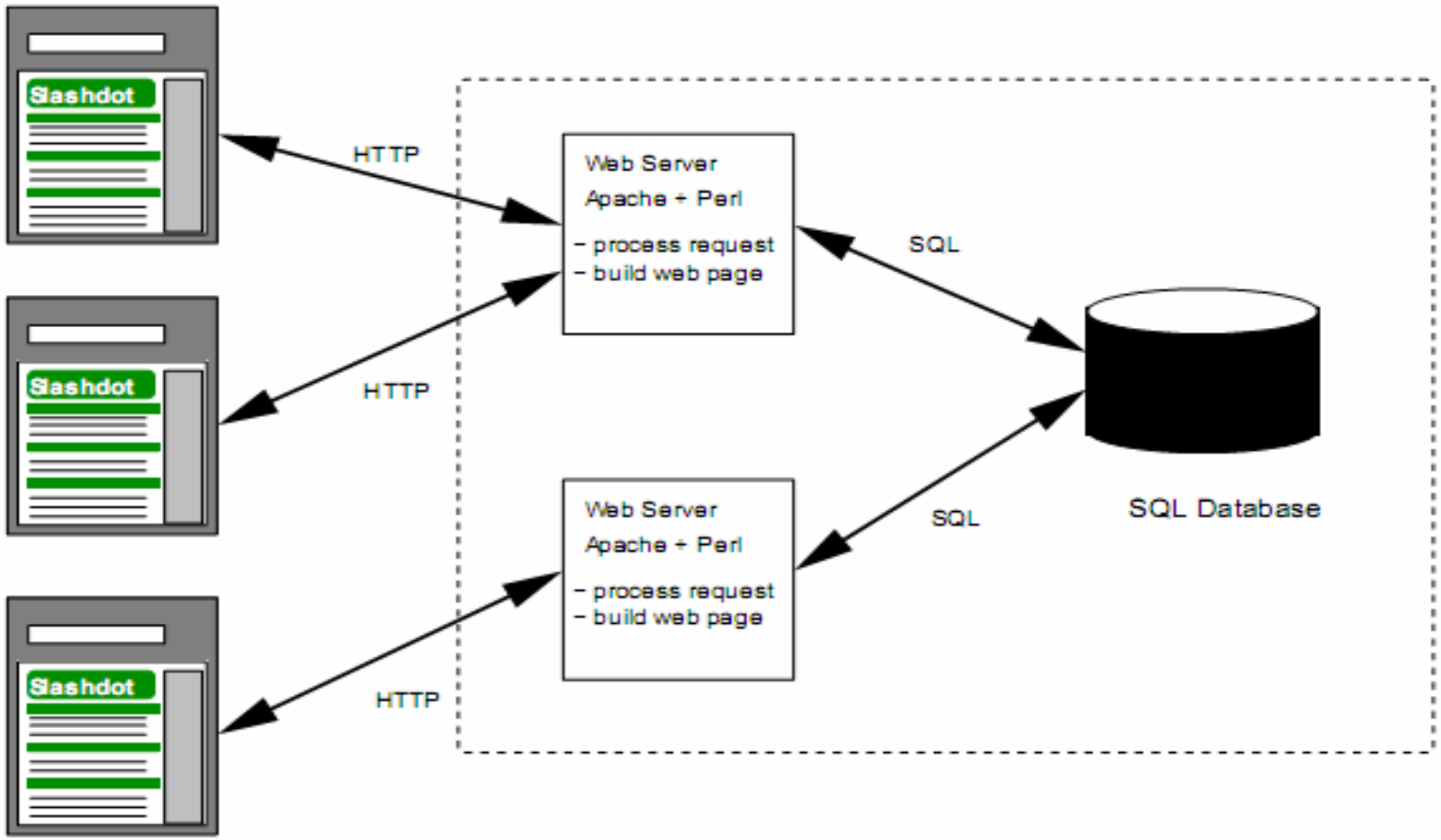
- Why replicate?
  - Reliability
    - Avoid single points of failure
  - Performance
    - Scalability in numbers and geographic area
- Why not replicate?
  - Replication transparency
    - Consistency issues
  - Updates are costly

Caching

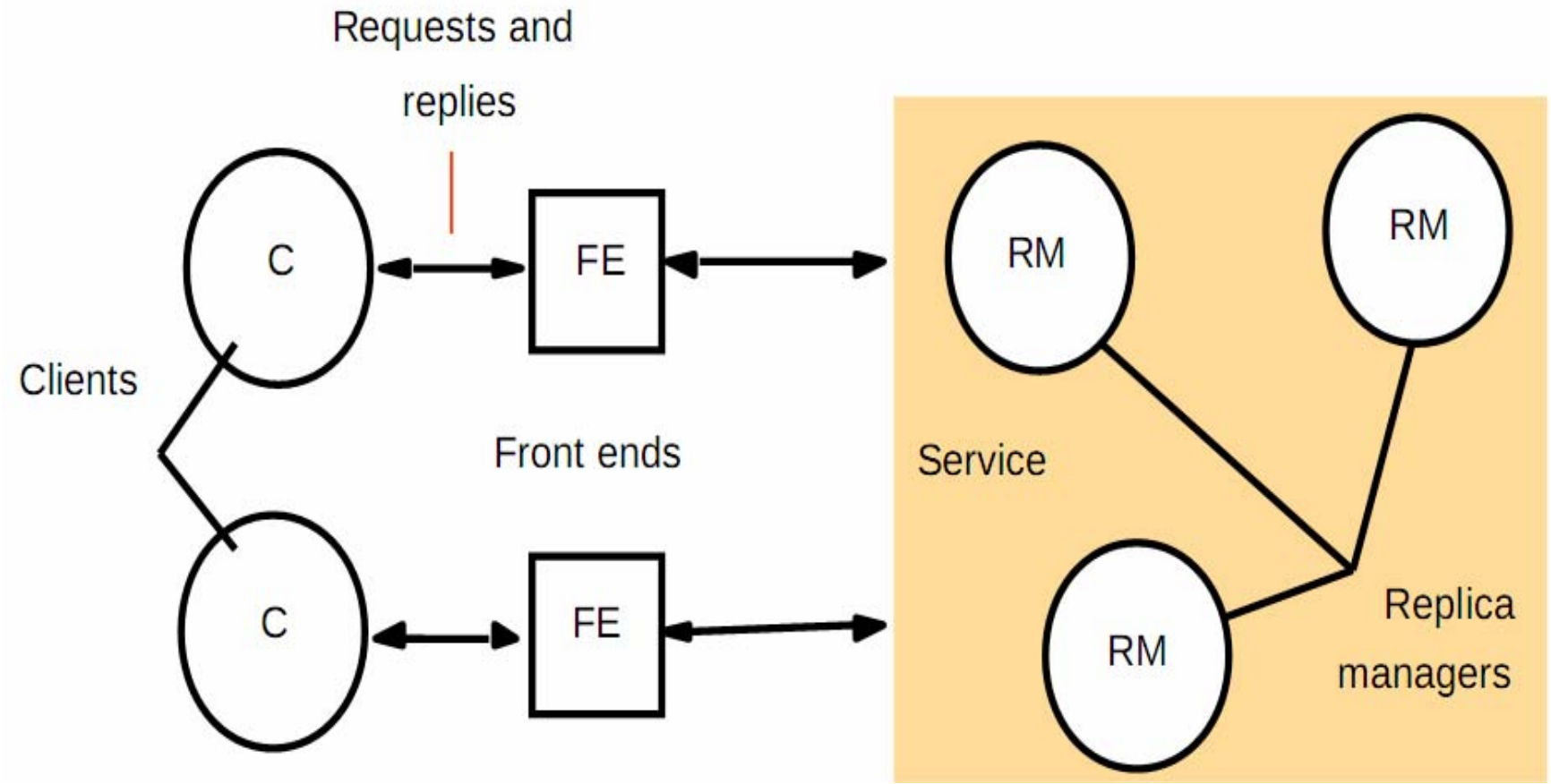
Data Replication (Caching):



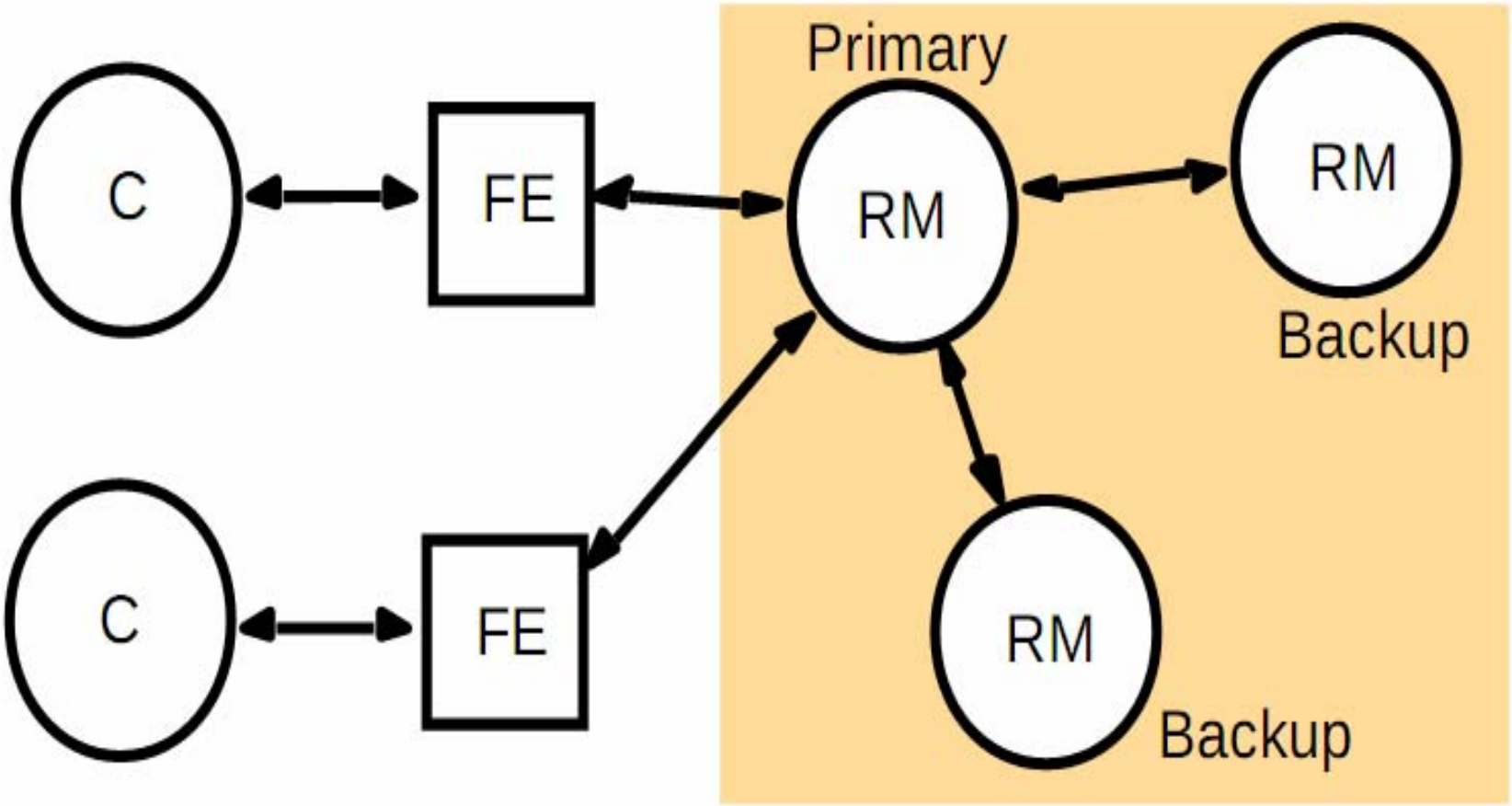
Control Replication



Arsitektur


Requests and
replies
C
FE
Clients
Front ends
C
FE
RM
RM
Service
RM
Replica
managers

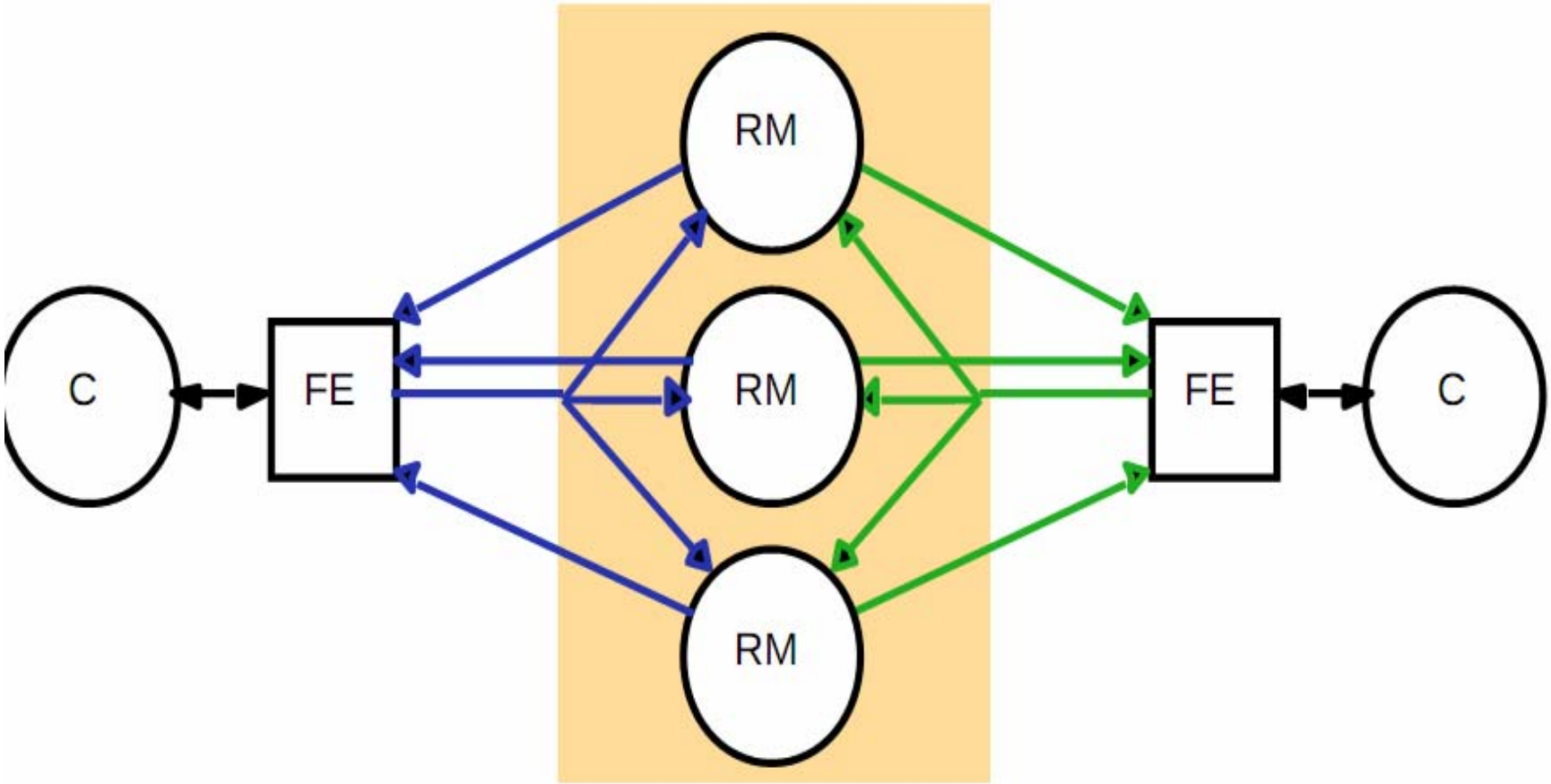
Passive Replication



Passive Replication

- If primary server is down, pick one backup to be primary
- Disadvantage: big overhead (primary must wait until all data is propagated to backups)
- Variant: FE sent all read request directly to backups
- Example: Sun Network Information System (NIS)

Active Replication

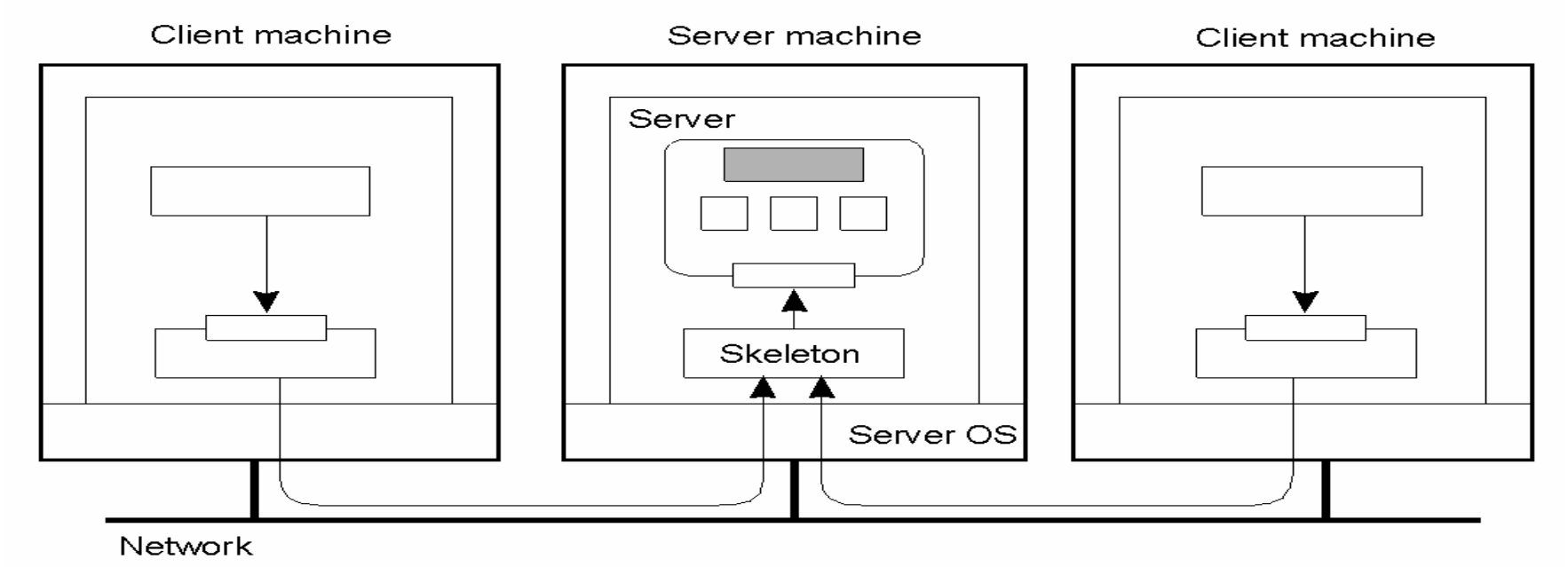


Active Replication

- Request
  - FE send multicast request to RM
  - Read access only to one RM
  - Write request goes to all RM in sequential orders
- Coordination
  - Group communication system send request to each RM

Object Replication (I)

- A distributed remote object shared by two different clients.



The object itself can handle concurrent operations

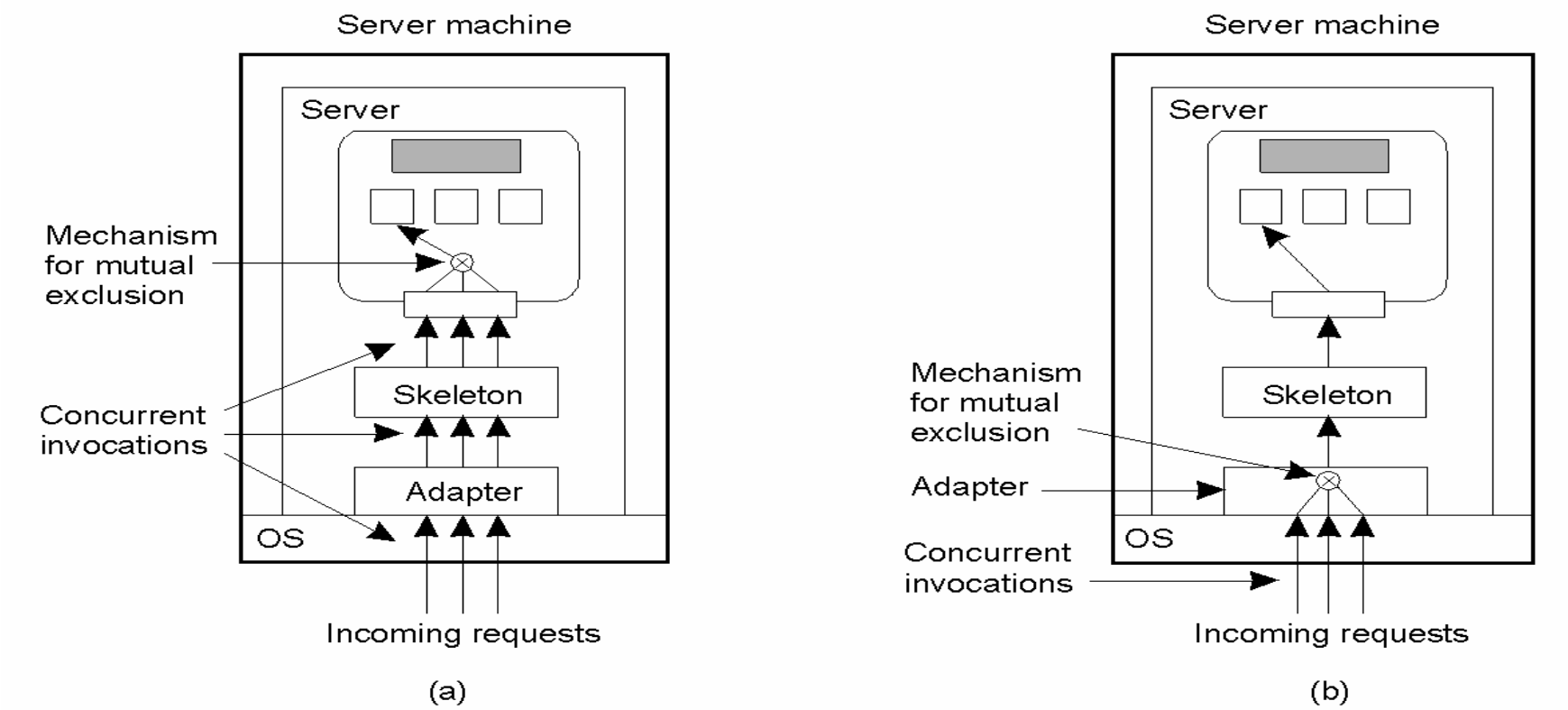
... OR:

The server protects the object in the presence of concurrency

Object Replication (II)

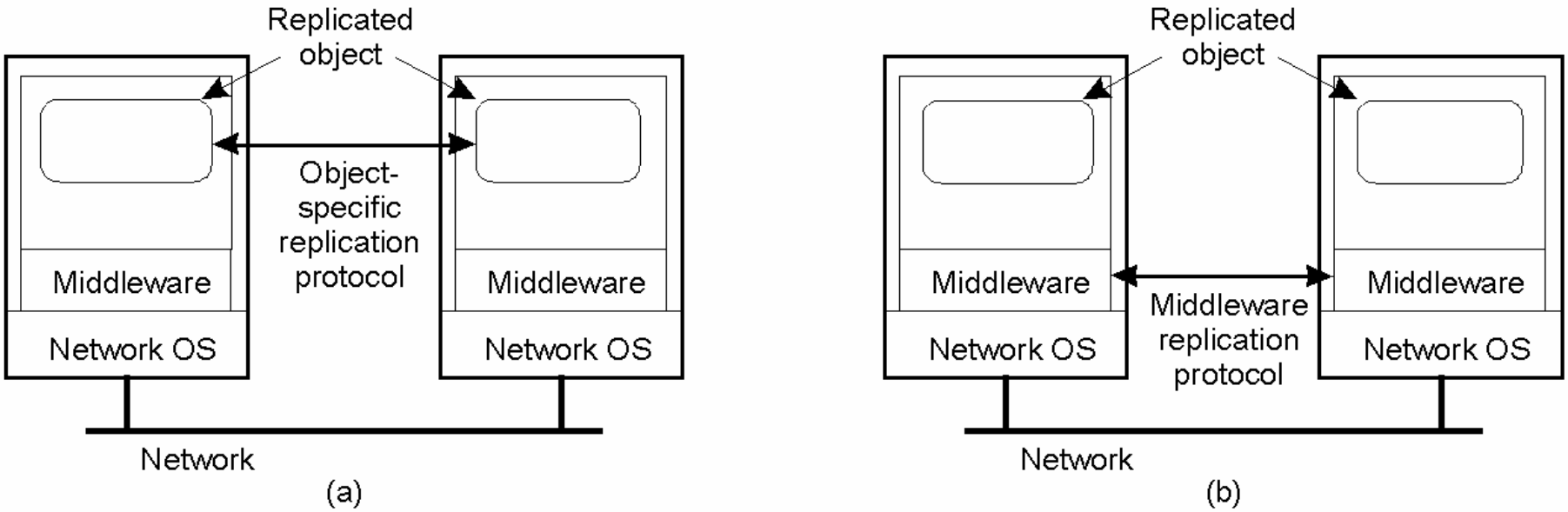
a) A remote object capable of handling concurrent invocations on its own.

b) A remote object for which an object adapter is required to handle concurrent invocations



Concurrency-aware objects vs system-managed concurrency

Object Replication (III)



Replication-aware objects can adopt object-specific policies

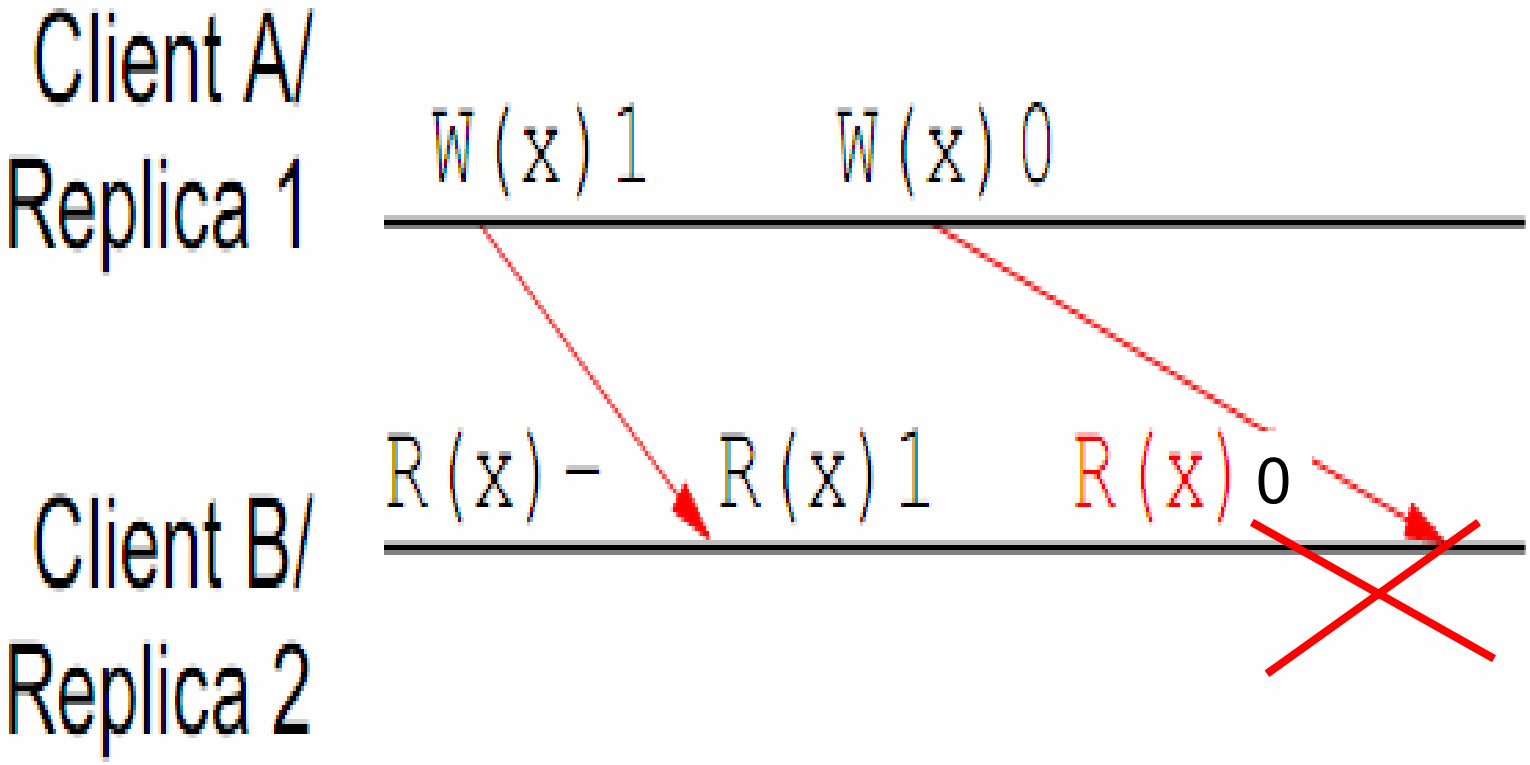
a) Replication-aware distributed objects.

b) A distributed system responsible for replica management

Operation on data store

- **Read**. Ri(x)b Client i performs a read for data item x and it returns b
- **Write**. Wi(x)a Client i performs write on data item x setting it to a
- Operations bergantung pada:
  - Time of issue (when request is sent by client)
  - Time of execution (when request is executed at a replica)
  - Time of completion (when reply is received by client)

Inconsistency



Inconsistency

- Masalah data:
  - How old is the data?
  - How old is the data allowed to be?
  - Time
  - Versions
- Operation order:
  - Were operations performed in the right order?
  - What orderings are allowed?

Consistency

- Clients can modify resource on any of the replicas.
- *What happens if another client requests resource before replica has informed others of modification, as in cache consistency in distributed file systems?*

Consistency

- Write/Update : operation could cause inconsistency on data
- Operation orders is critical
- Operations Conflicts:
  - Read-write conflict (only 1 write)
  - Write-write conflict (multiple concurrent writes)

Contoh

Client A: `x = 1; x = 0;`

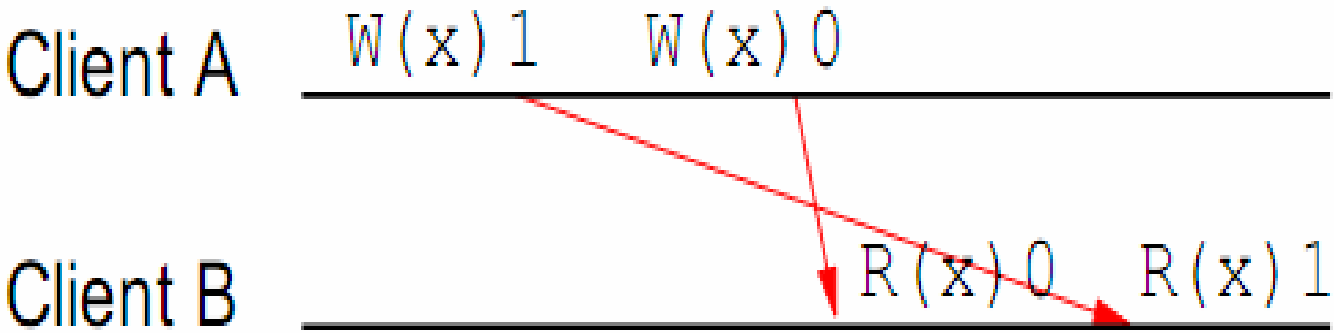
Client B: `print(x);`

`print(x);`

Possible results:

--, 11, 10, 00

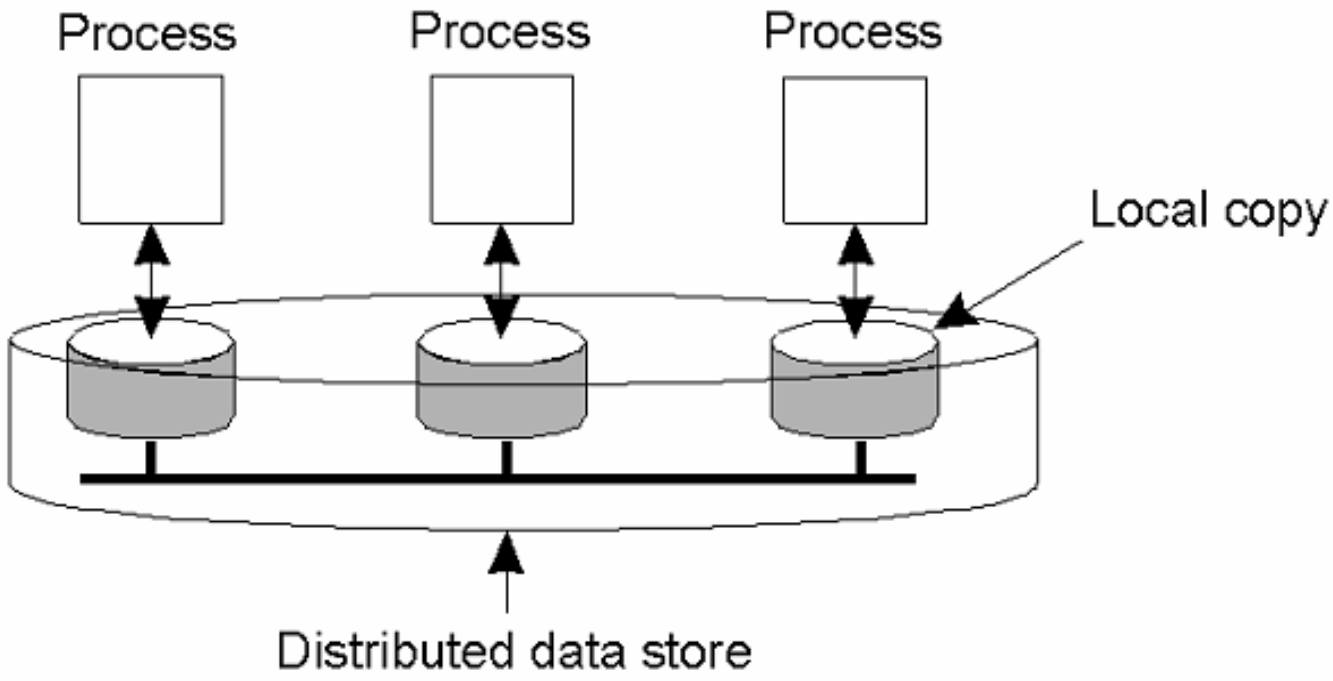
How about 01?



Consistency Model

- **Data centric**
  - Strict consistency
  - Sequential consistency
  - Causal consistency
  - Release consistency
  - Lazy release consistency
- **Client centric**

Data centric consistency model



The general organization of a logical data store, physically distributed and replicated across multiple machines.

Data centric

- Contract between processes and the data store. If processes obey certain rules, data store will work correctly
- Normally one would like: “read returns the result of most recent write”
- However: No global clock! What is most recent (last) write?
- Conflict: Two operations in the same interval on the same data item and at least one is a write.

Data centric ordering

Strong ordering

- Write performed in order
- Strict, Sequential, Causal

Weak ordering

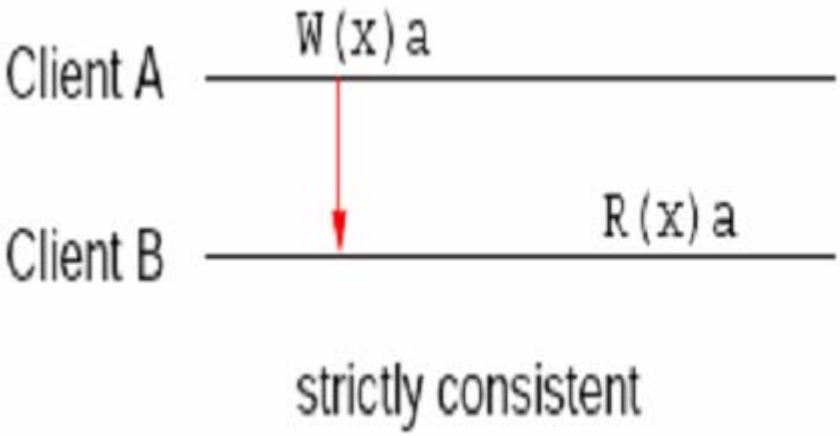
- Series of write grouped on single replica
- Results of grouped writes propagated
- Release

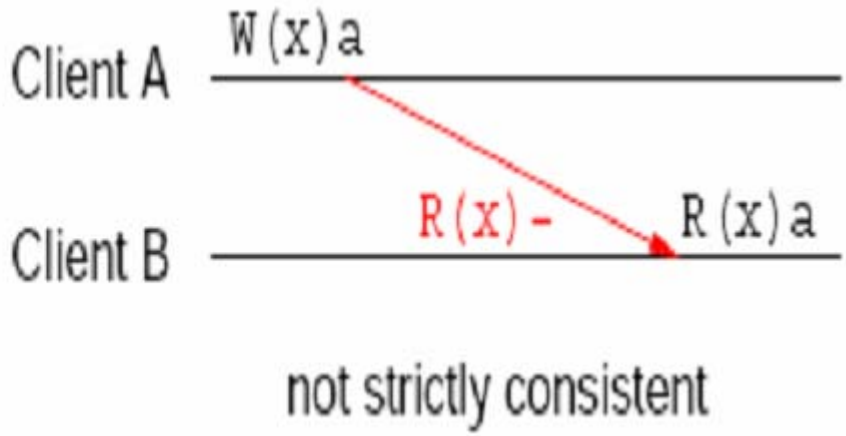
Simbol

- W(x) a, berarti client tertentu menulis sesuatu bernilai a
- Tanda panah merah, berarti client tertentu menulis di tempat client lain, sehingga konsisten
- R(x) a, berarti client tertentu membaca nilai a

Strict consistency

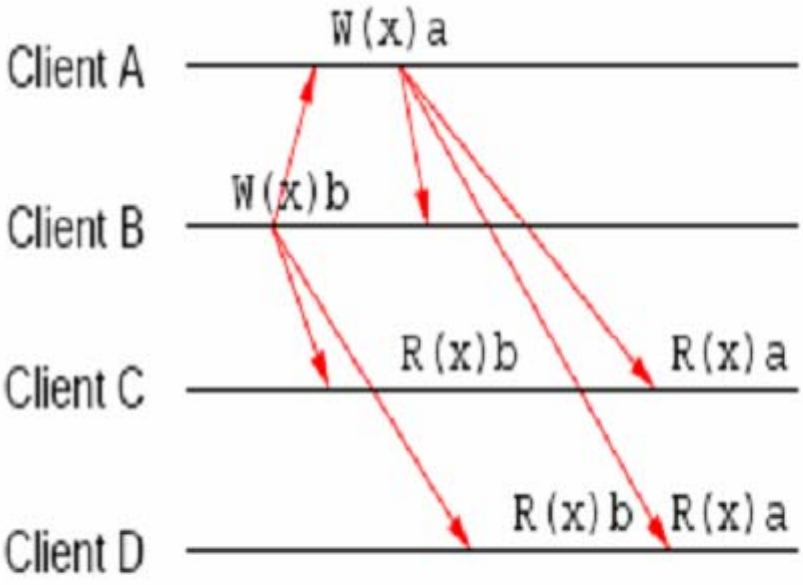
- Any read on data item x returns a value corresponding to the results of the most recent write on x
- Hard to implement on distributed system because of “most recent” due to network delays



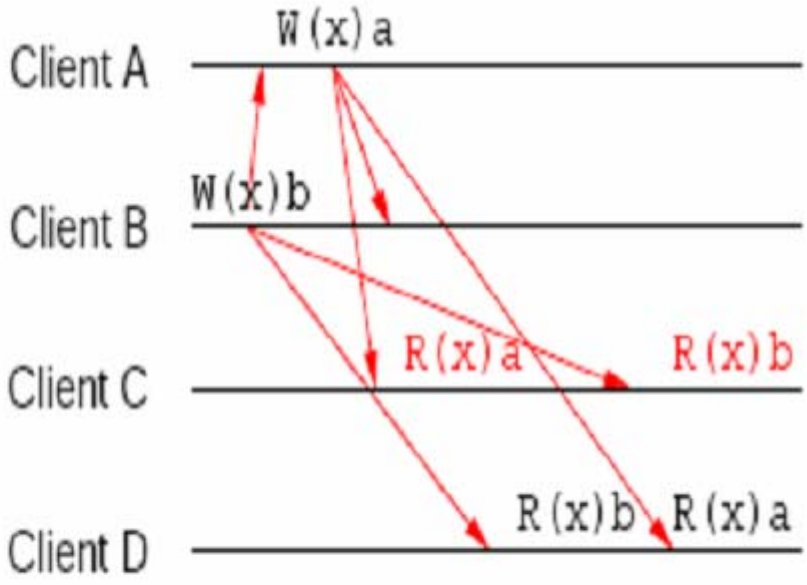


Sequential Consistency

- All operation are done sequentially
- Not ordered according to time



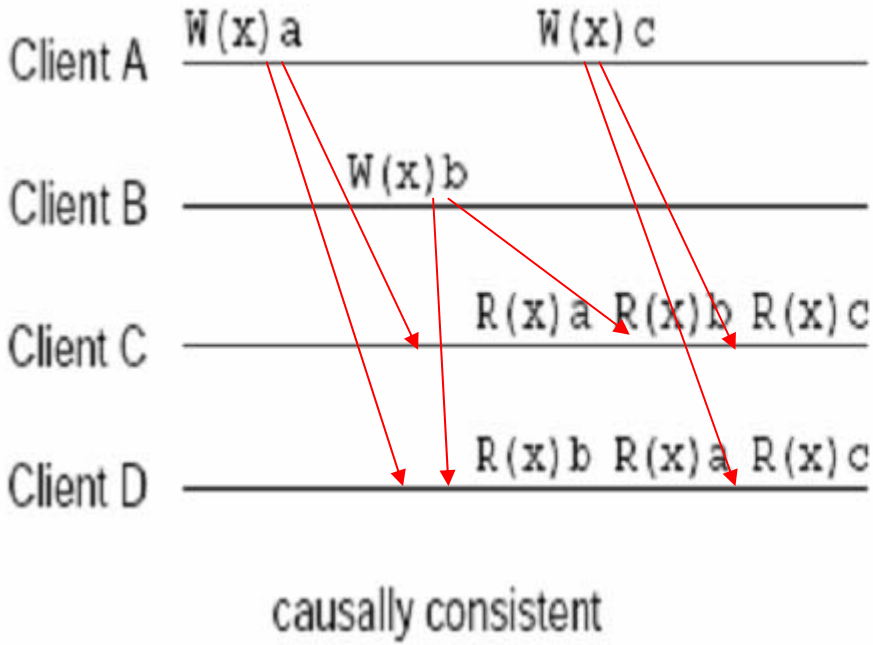
sequential



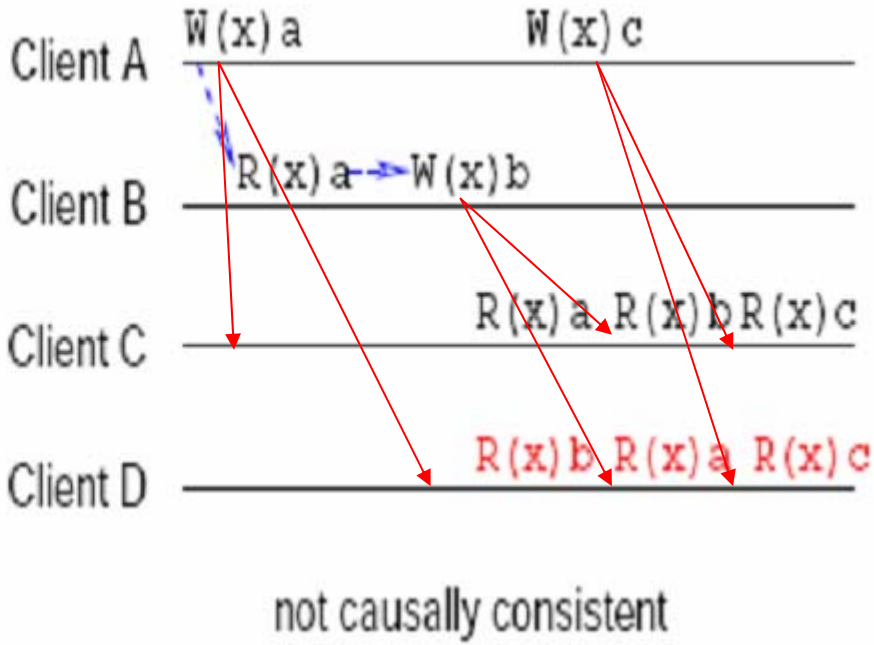
not sequential

Causally related

- Bersifat tidak independen
- Suatu operasi disebabkan oleh operasi lain
  - Read -> write (same client)
  - W(x) -> R(x) (same/different client)



causally consistent

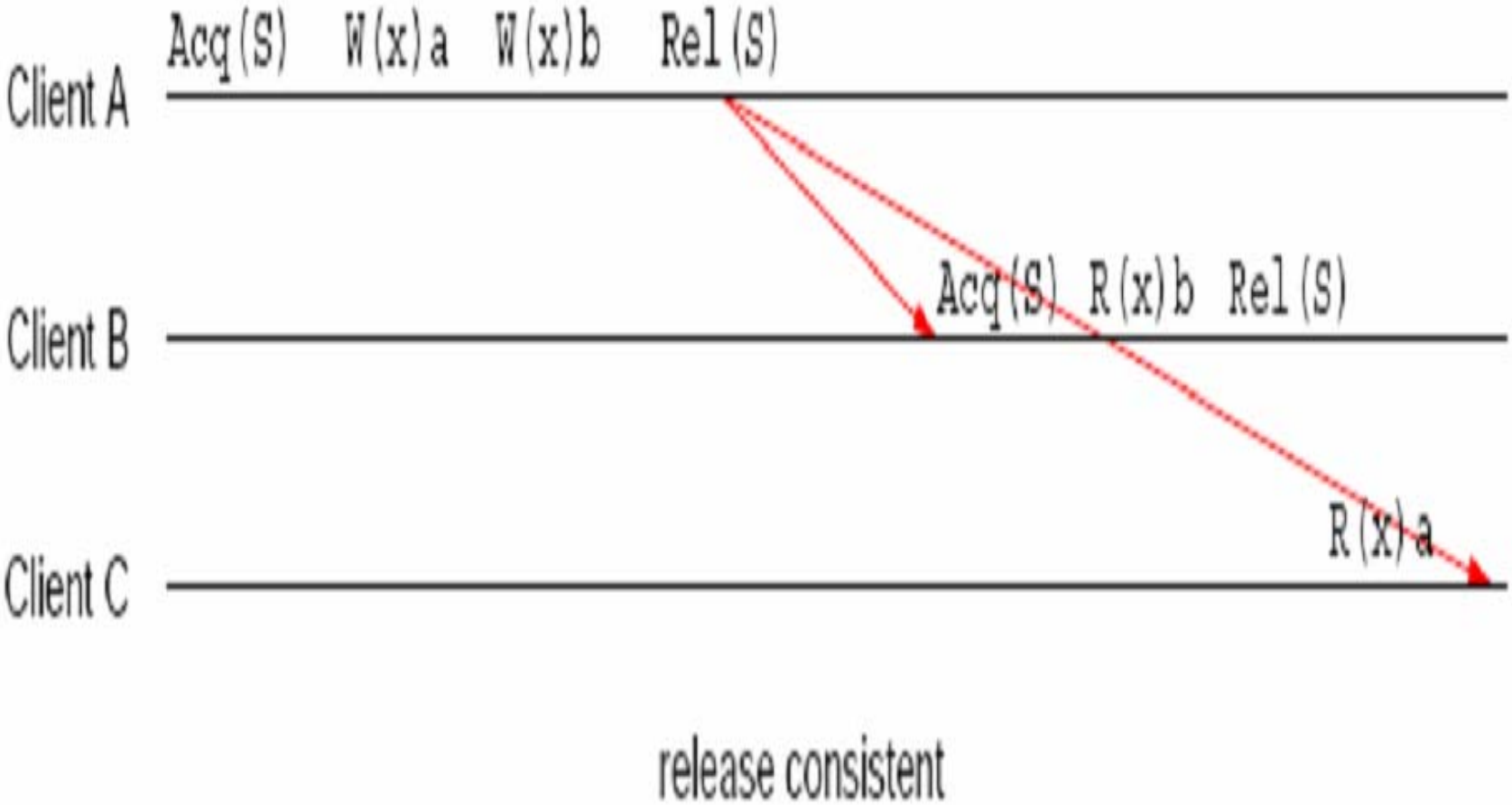


not causally consistent

Release consistency

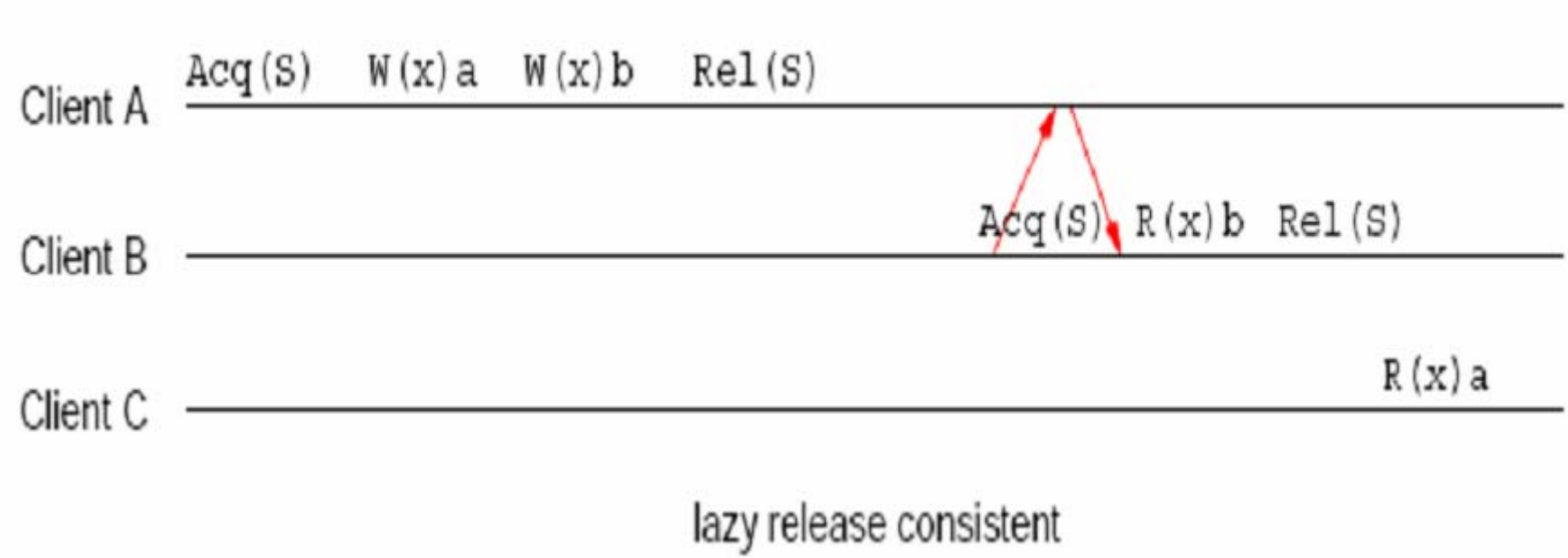
- Explicit separation of synchonization task
  - **Acquire** -> bring local state up to date
  - **Release** -> propagate all local updates
- Orders are FIFO consistent
- Release only after all read/write by client is completed
- Read/write only after all acquire by client is completed

Release consistency



Lazy release consistency

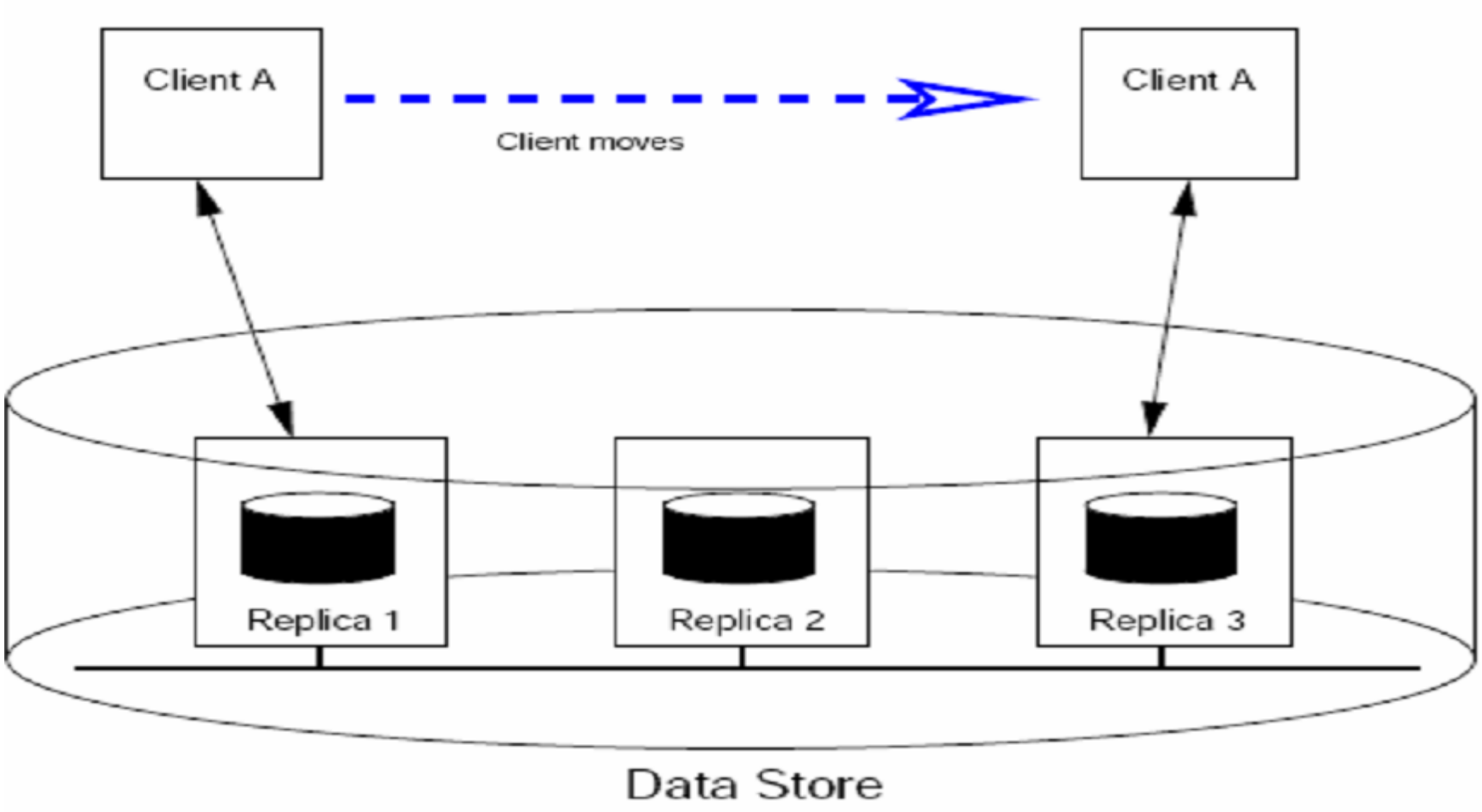
- Don't send updates on release
- Acquire cause clients get newest state



Client centric consistency model

- Provides guarantees about ordering of operation for single client
- Single client access data store
- Client accesses different replicas
- Data isn't shared by clients
- Each client will see different orderings

Client centric

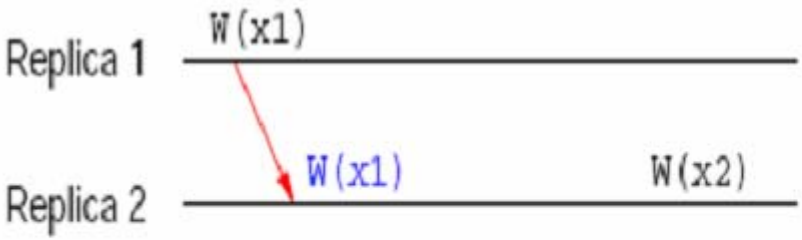


Monotonic Read

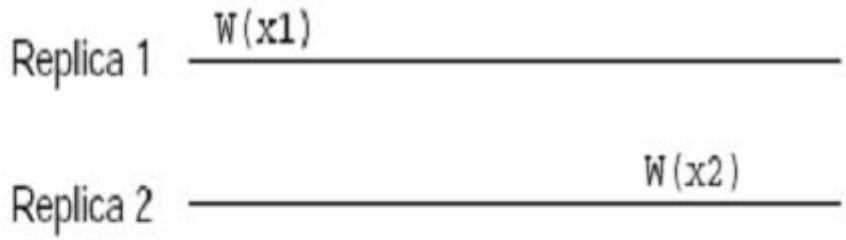
- If a client has seen a value of x at a time t, it will never see older version of x in the future
- Eg.: Reading incoming email messages will fetches the latest updates

Monotonic Write

- A write operation on data item x is completed before any successive write on x by the same client
- All writes by single client are sequentially ordered
- Eg.: maintaining version of replicated files in correct order everywhere



monotonic-write consistent



not monotonic-write consistent

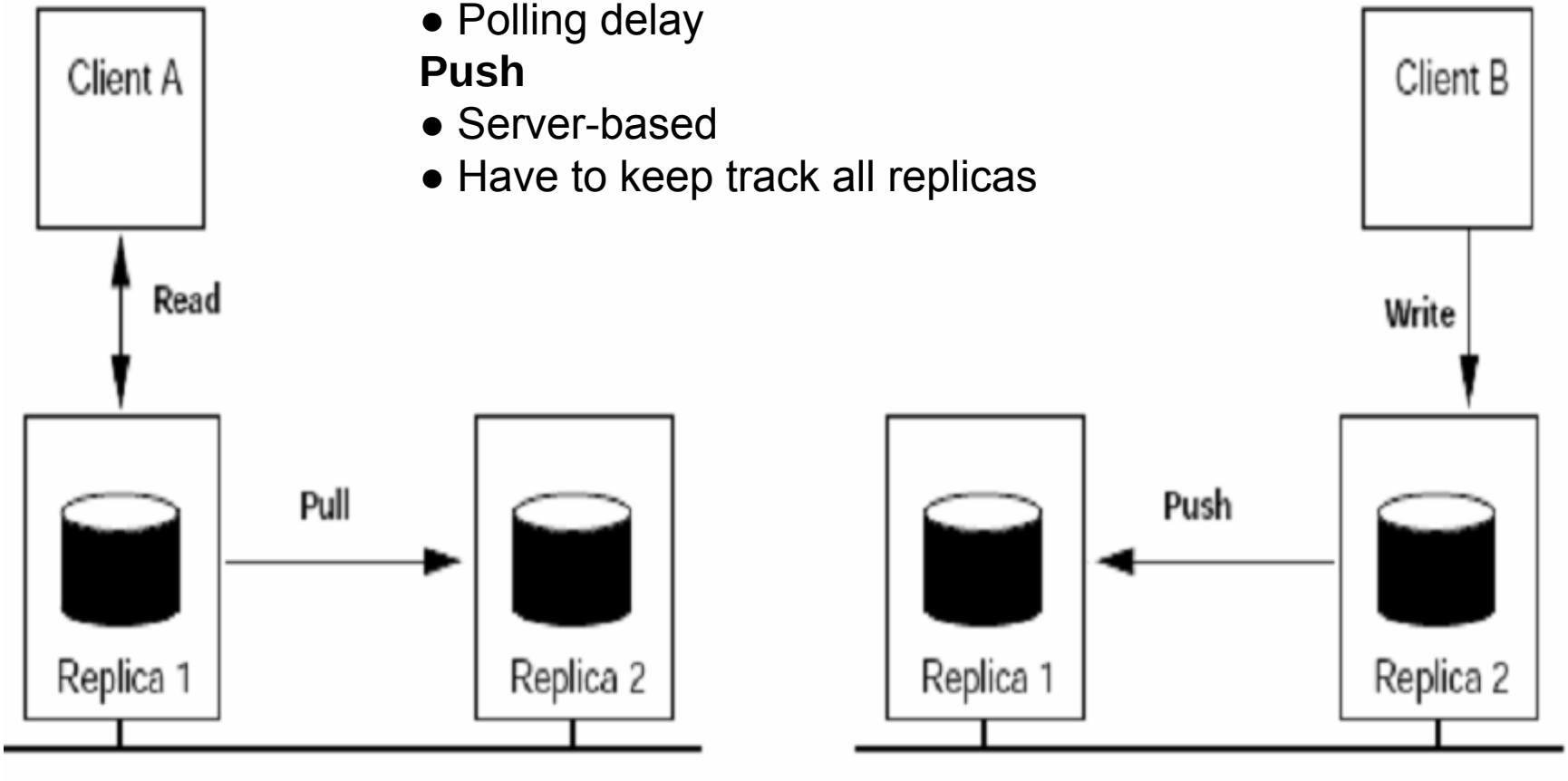
Read your writes

- Effect of a write on x will always be seen by a successive read of x by same client
- Eg.: update your web page and guaranteeing that your web browser shows newest version instead of cached version

Update Replica

Pull
- On request
- Client-based
- Polling delay

Push
- Server-based
- Have to keep track all replicas



NEXT

- Security in Distributed System