Distributed Systems

Replication

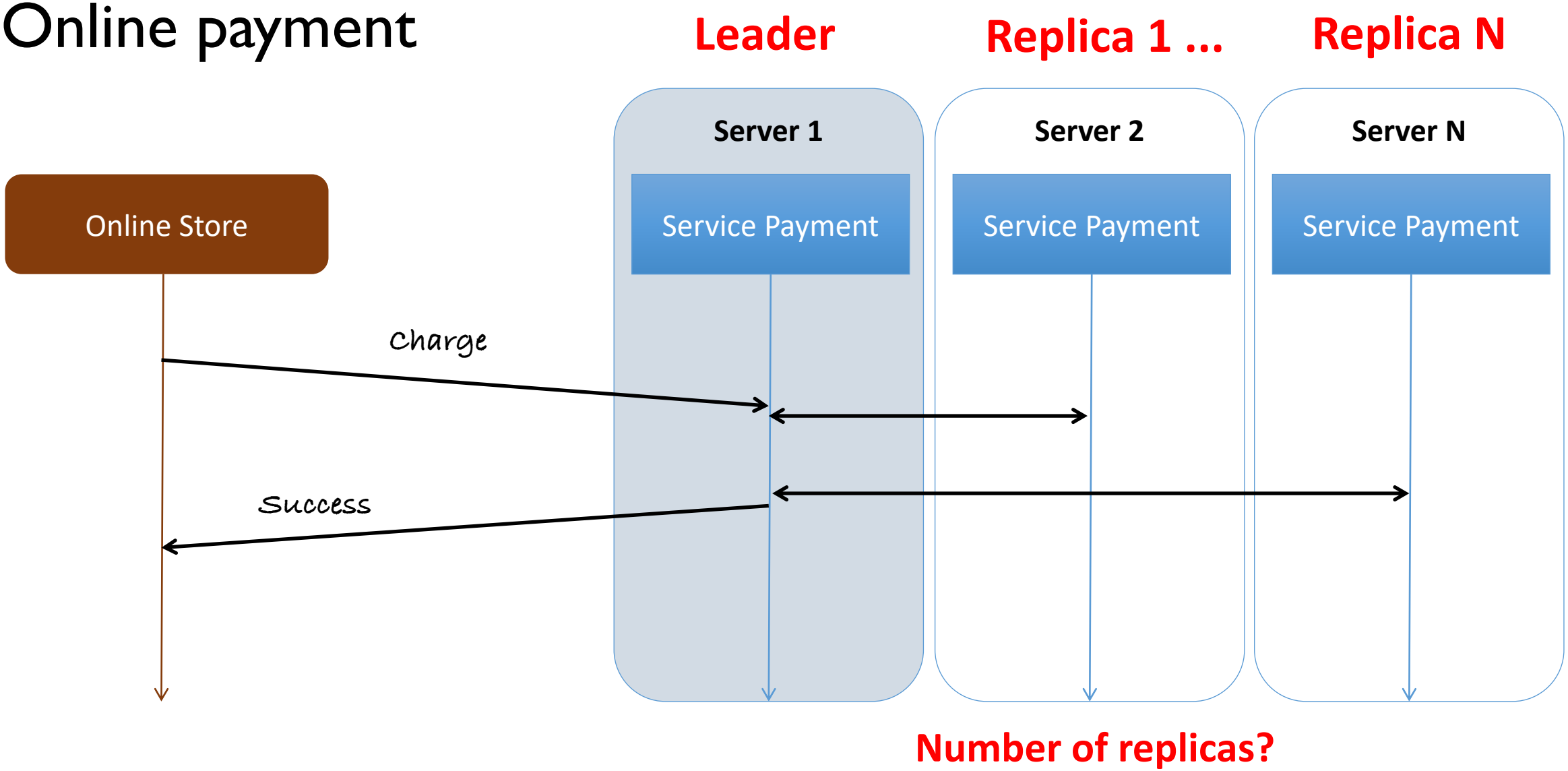
Thoai Nam

High Performance Computing Lab (HPC Lab)

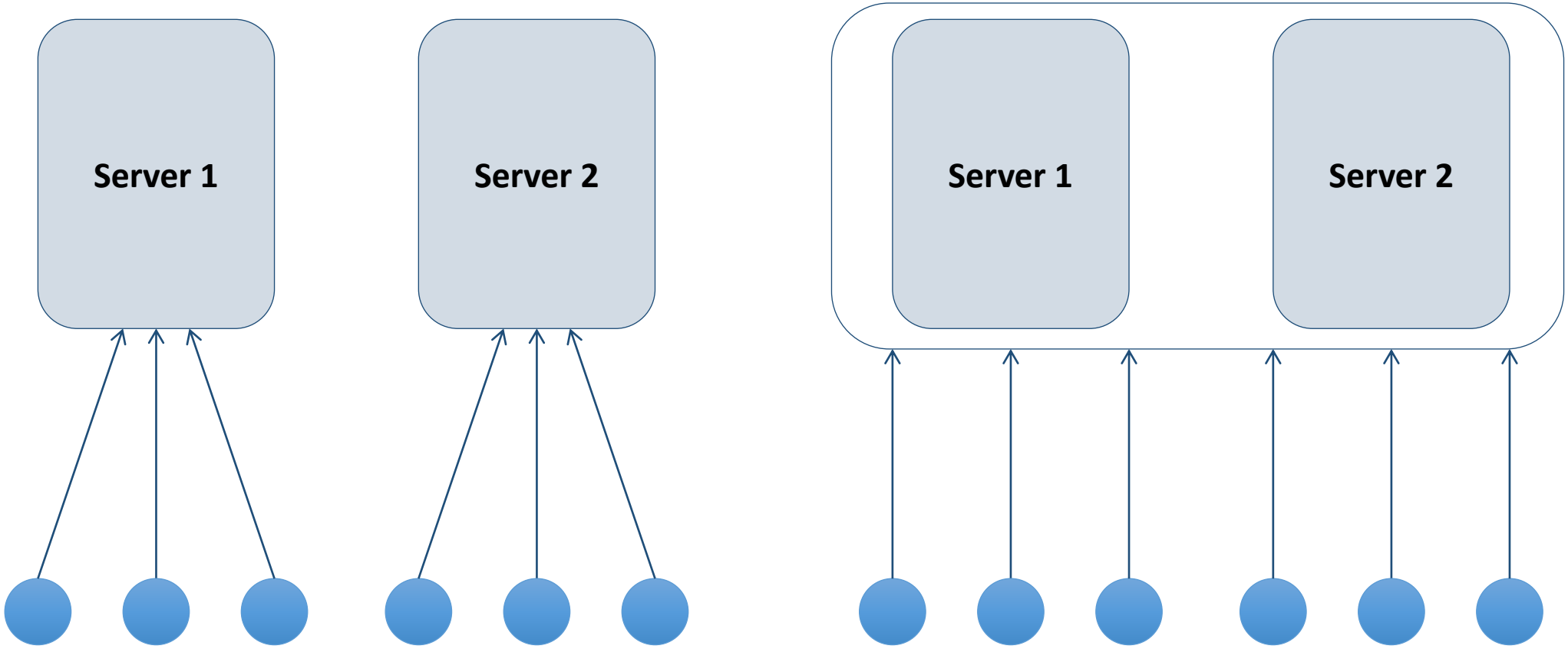
Faculty of Computer Science and Engineering

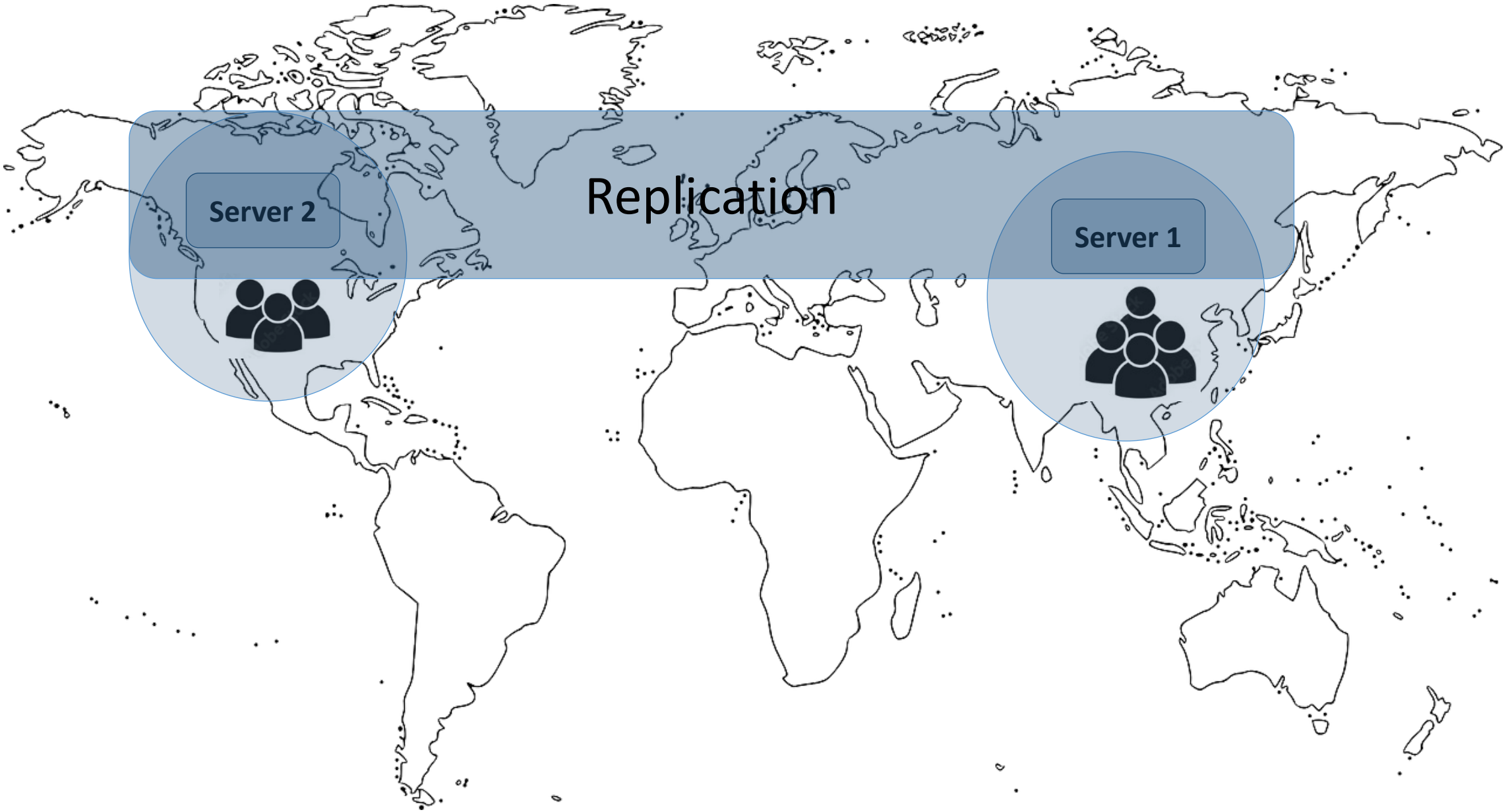
HCMC University of Technology

Online payment



Load balancing: replication




Server 2
Replication
Server 1

Reasons for replication

- Copy of the same data in multiple nodes
  - Databases (Apache Cassandra), file systems (NFS – Network file sharing)...
- A node that has a copy of the data is called replica
  - If some replicas are faulty, others are still accessible
  - Spread load across replicas
  - If data does not change, data distribution becomes relatively easy.

Reasons for replication

- Enhancing reliability
  - Protection against hardware crashes and corruption of data
- Improving performance
  - Scaling in terms of numbers and geographical area
  - Caching is a special form of replication
  - Trade-off between performance and scalability.

Challenges (Keep replicas consistent)

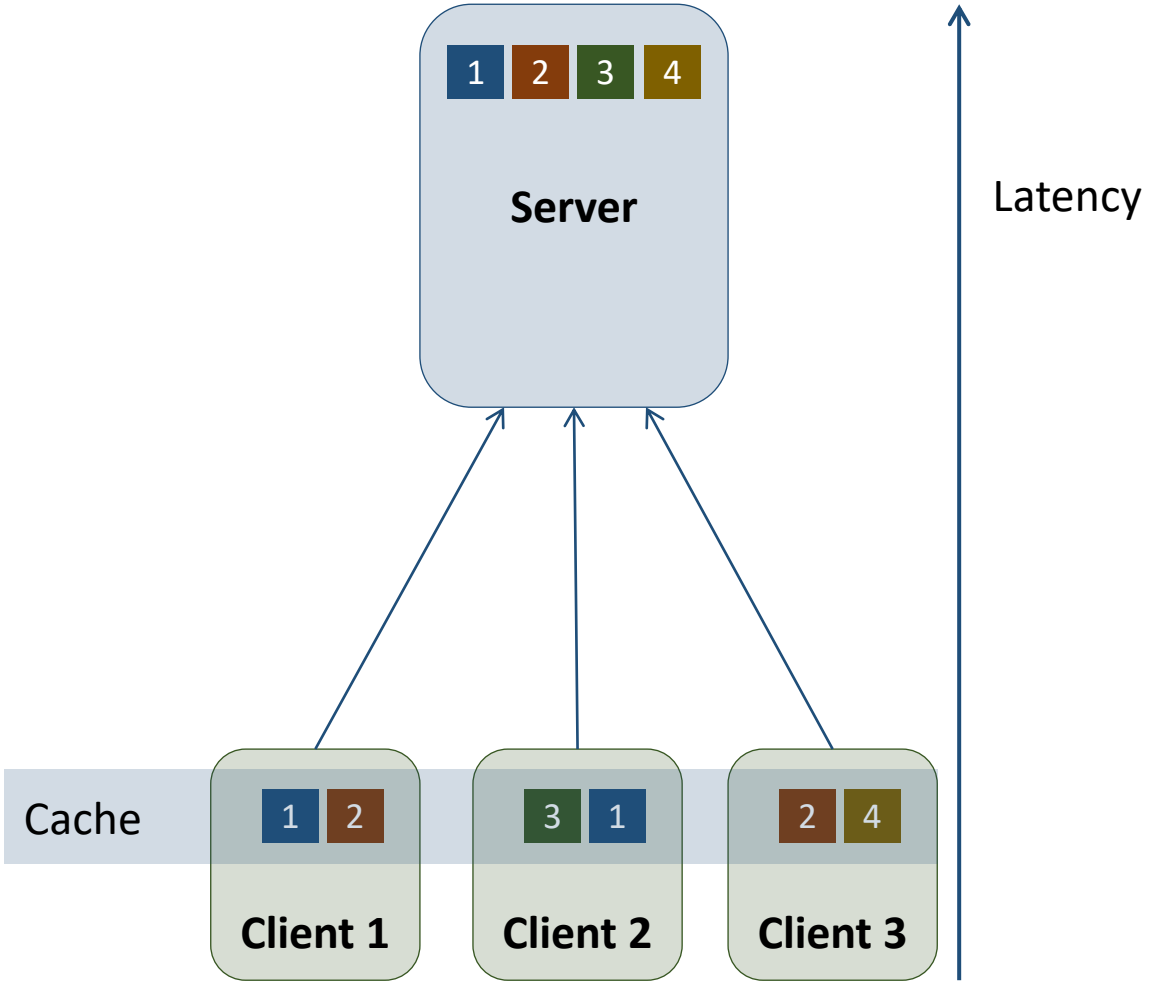
- When one copy is updated, other copies need to be updated, as well
- A number of consistency models
- Protocols for distribution of updates.

System performance and replication

- Replication improves server performance, particularly reliability and availability
- However, if data is not replicated properly, this can lead to a system that cannot be utilized in practice.

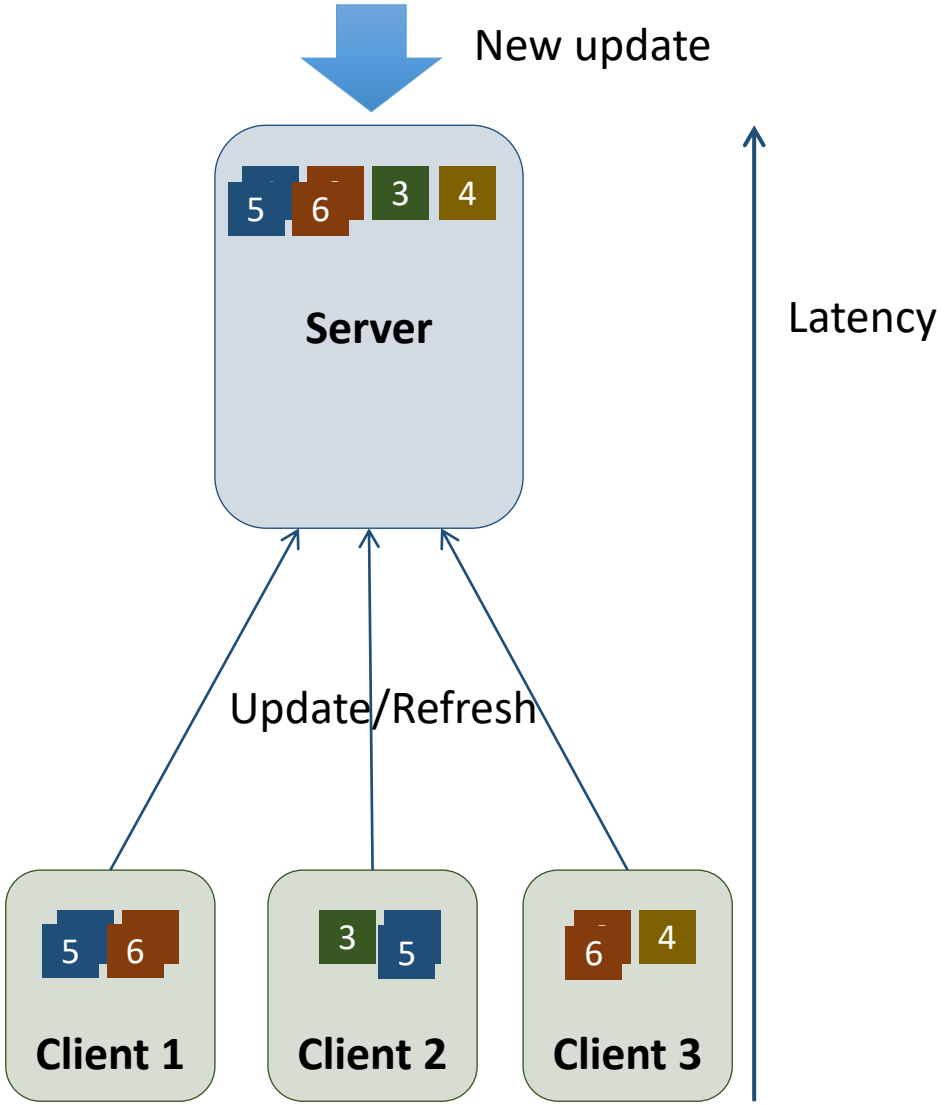
Replication caching (1)

The idea of keeping around pieces of information that you utilize frequently



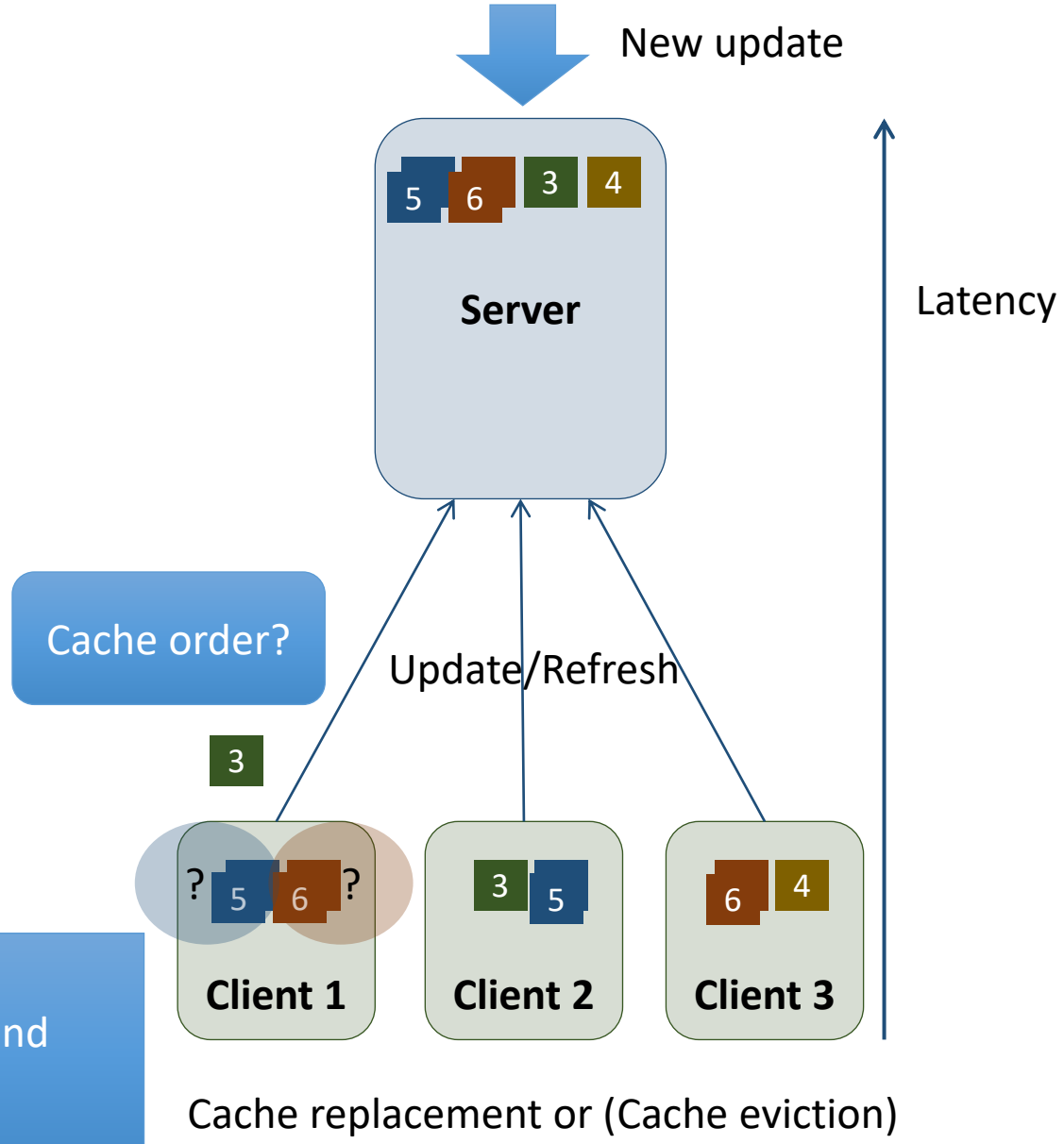
Replication caching (2)

Publish and subscribe architecture, e.g., trigger on events



Replication caching (3)

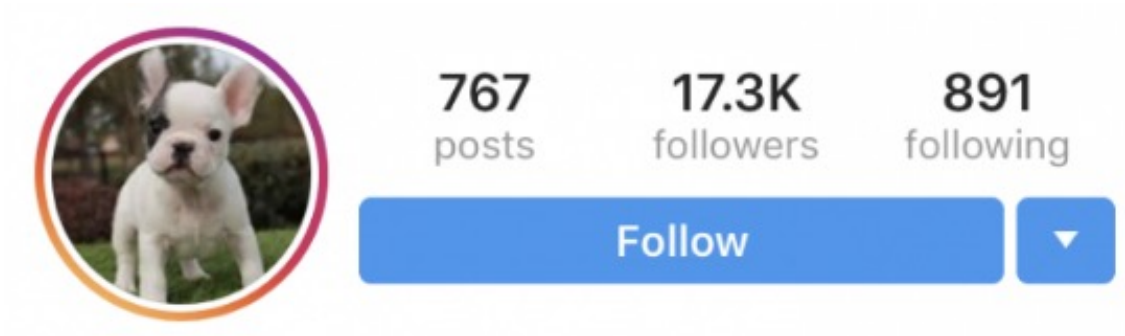
- Cache eviction policies
  - First-in, First out (FIFO) – evict or overwrite whatever has been sitting in the cache the longest
  - Random eviction – adding new data to the cache and overwriting old data at random
  - Least recently used (LRU) – evicting the data item that has gone the longest untouched
- In the context of distributed systems, Content Distributed Networks (CDNs)
  - Computer around the world that contain copies of popular websites (in cache)

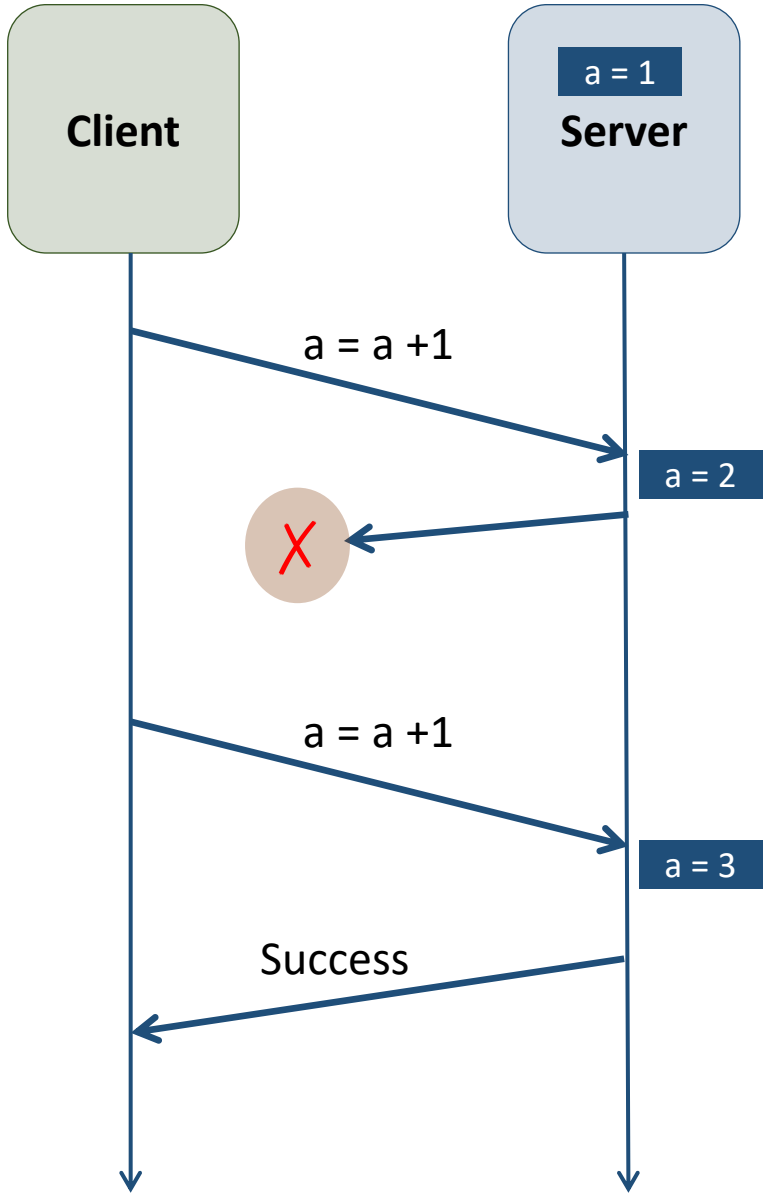


Self-organizing lists:
Principle that suggests that chaos is one of the most well-designed and efficient structures available
- Analogy: your pile of papers in your desk.

Data changes => Consistency

- Tracking and managing changes on the data is a key problem in replication.
- The system must identified which requests have been already seen.

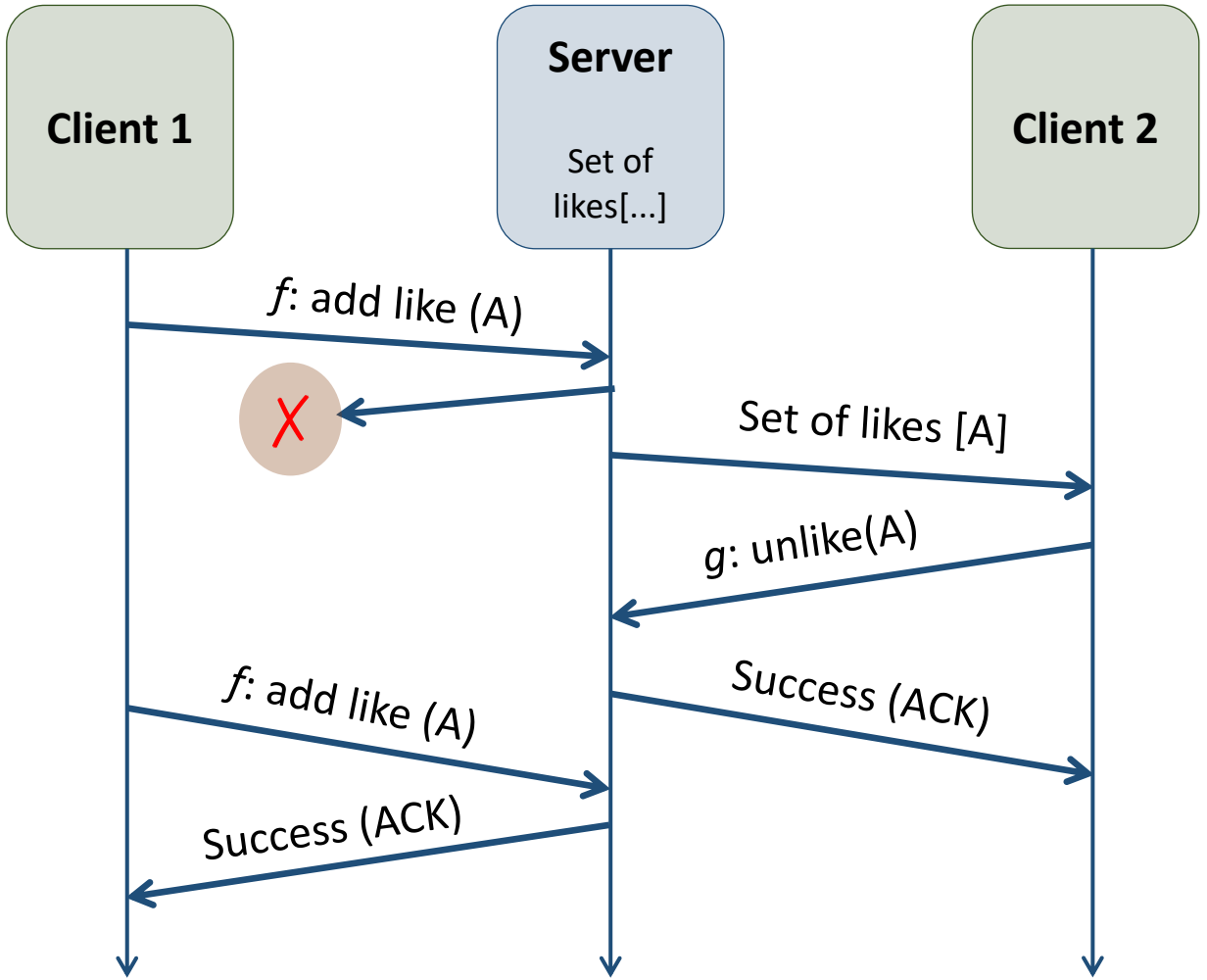




Idempotence

- Idempotence: an operation that has not additional effect in a final result even after applied multiple times.
  - Requests can be retried without deduplication
  - A function f is idempotent if $f(x) = f(f(x))$
  - Not idempotent: f(likeCount) = likeCount + 1
  - Idempotent: f(likeSet) = likeSet ∪ {userID}
- Choice of retry behavior:
  - At-most-once semantics: send request, don't retry, update may not happen
  - At-least-once semantics: retry request until acknowledged, many repeat update
  - Exactly-once semantics: retry + idempotence or deduplication.

Adding and then removing again (1)

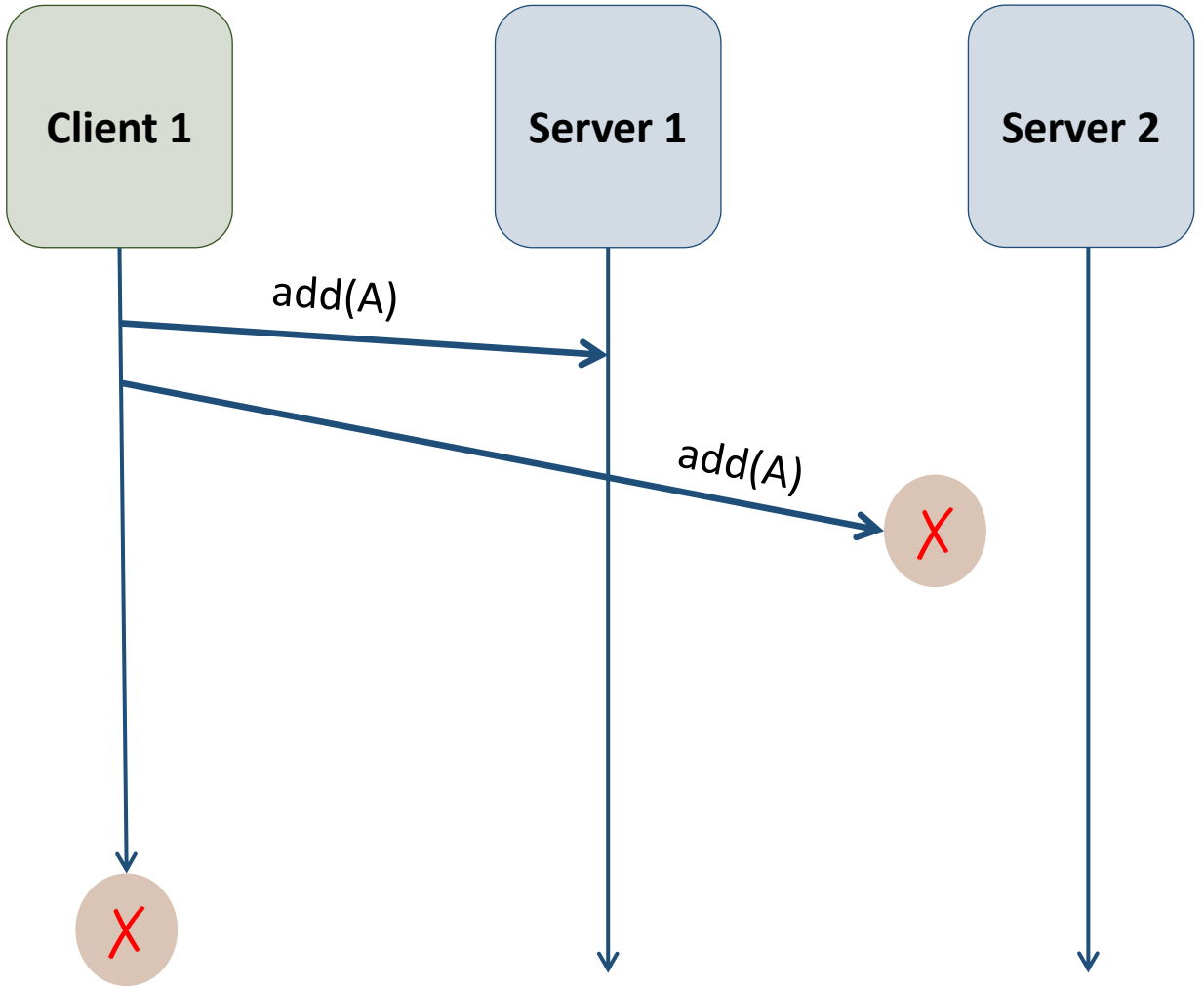


$f(A)$ = Set of likes $\cup$ {A};

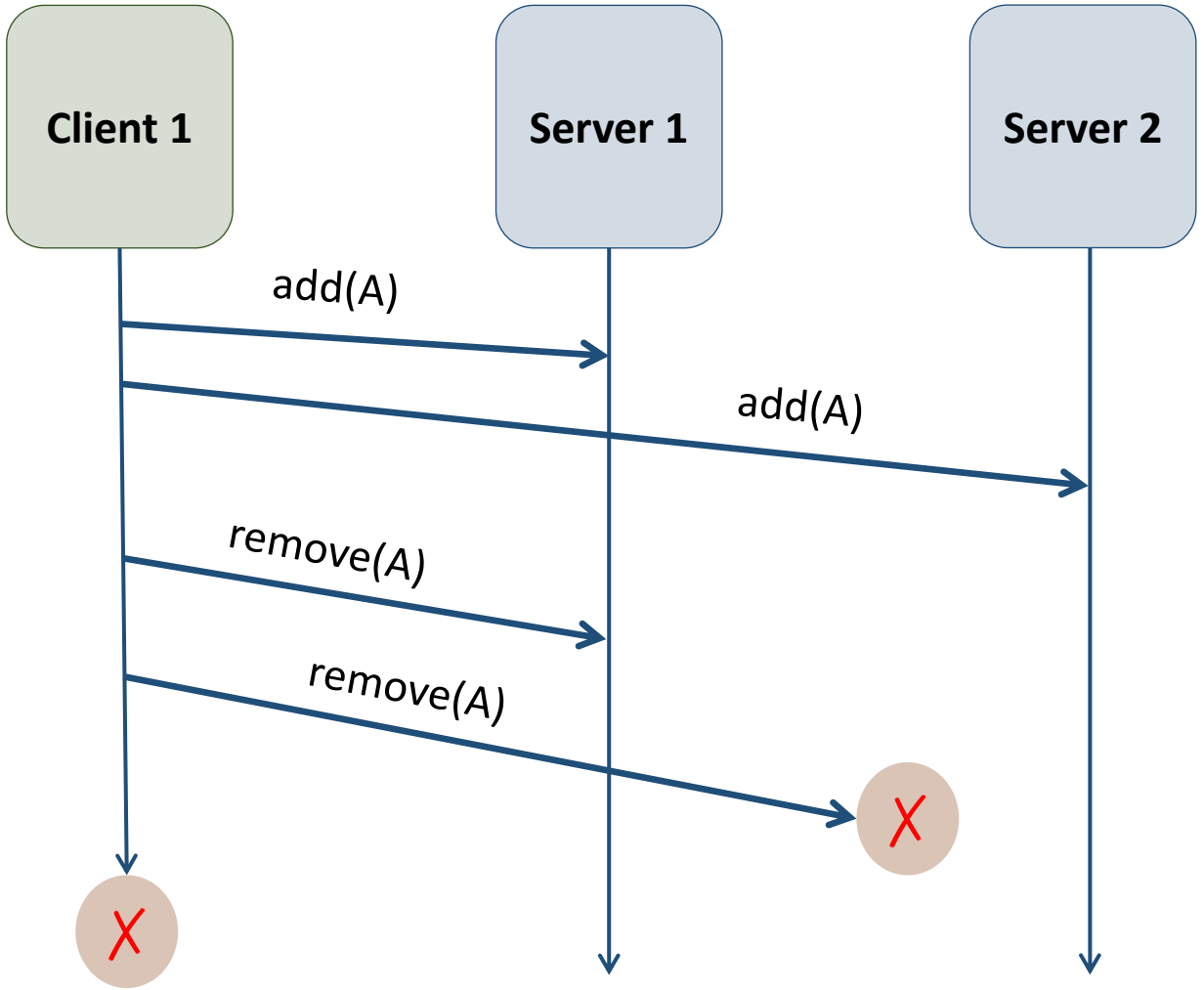
$g(A)$ = Set of likes $\setminus$ {A};

idempotent?

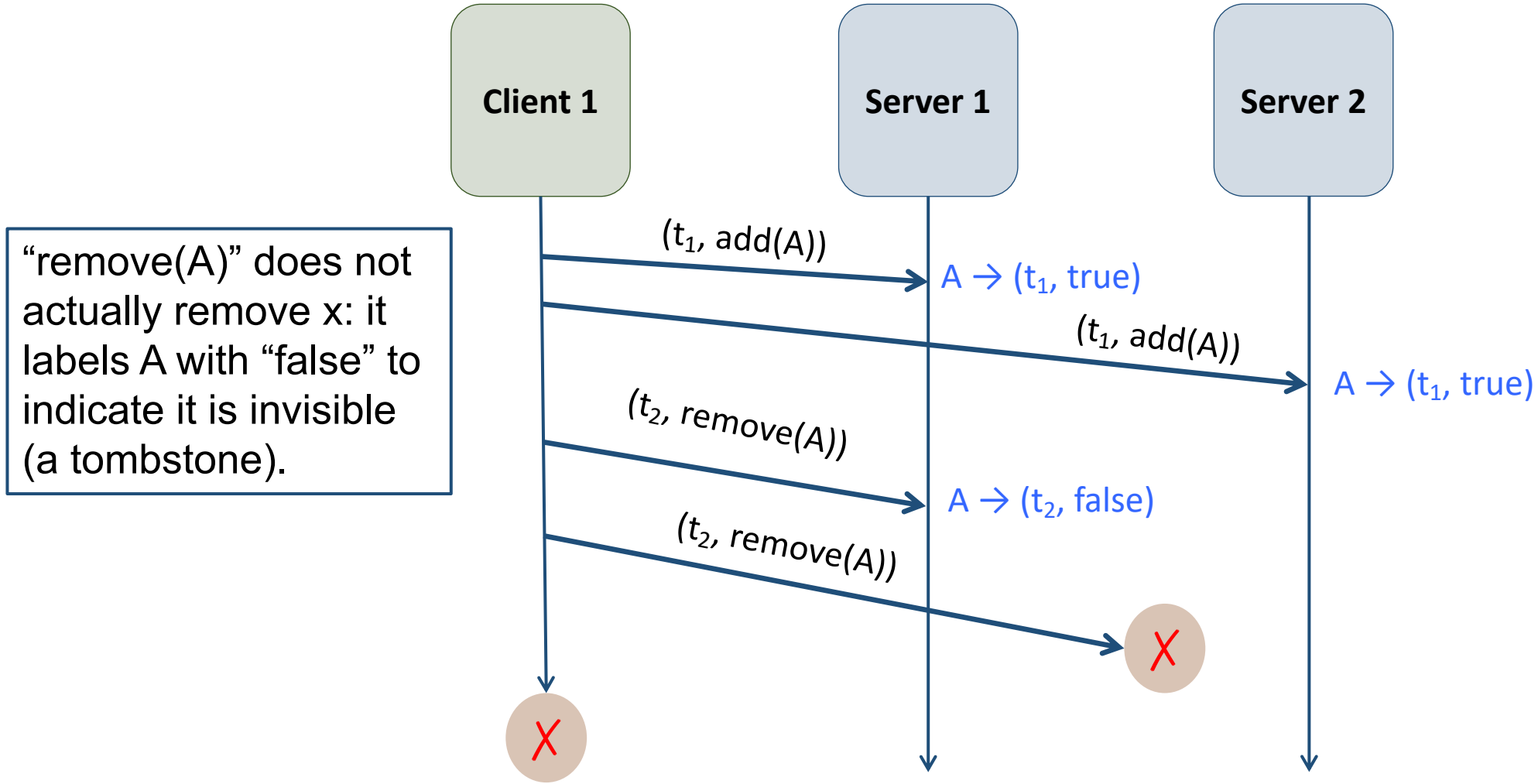
Adding and then removing again (2)



Adding and then removing again (3)



Adding and then removing again (4)



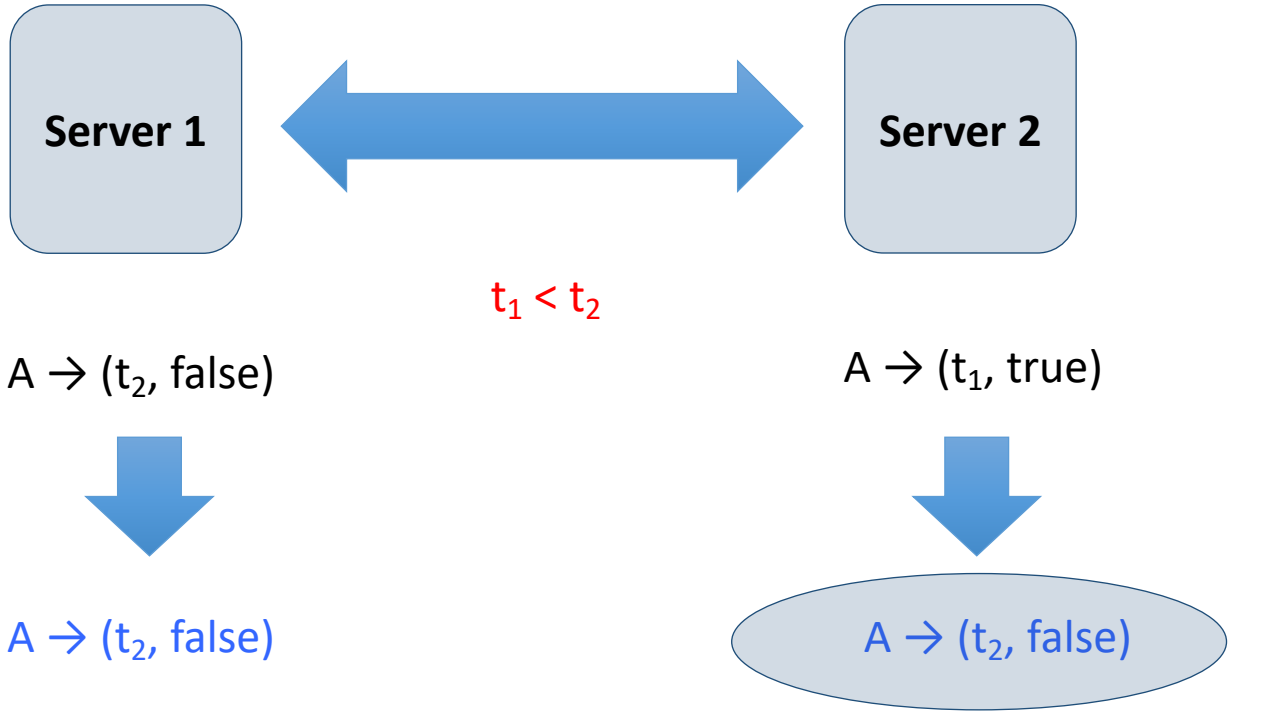
"remove(A)" does not actually remove x: it labels A with "false" to indicate it is invisible (a tombstone).

Replication issues

- Main issue
  - To keep replicas consistent, we generally need to ensure that all conflicting operations are done in the same order everywhere
- Conflict operations
  - Read–write conflict: a read operation and a write operation act concurrently
  - Write–write conflict: two concurrent write operations
- Issue
  - Guaranteeing global ordering on conflicting operations may be a costly operation, downgrading scalability
  - Solution: weaken consistency requirements so that hopefully global synchronization can be avoided.

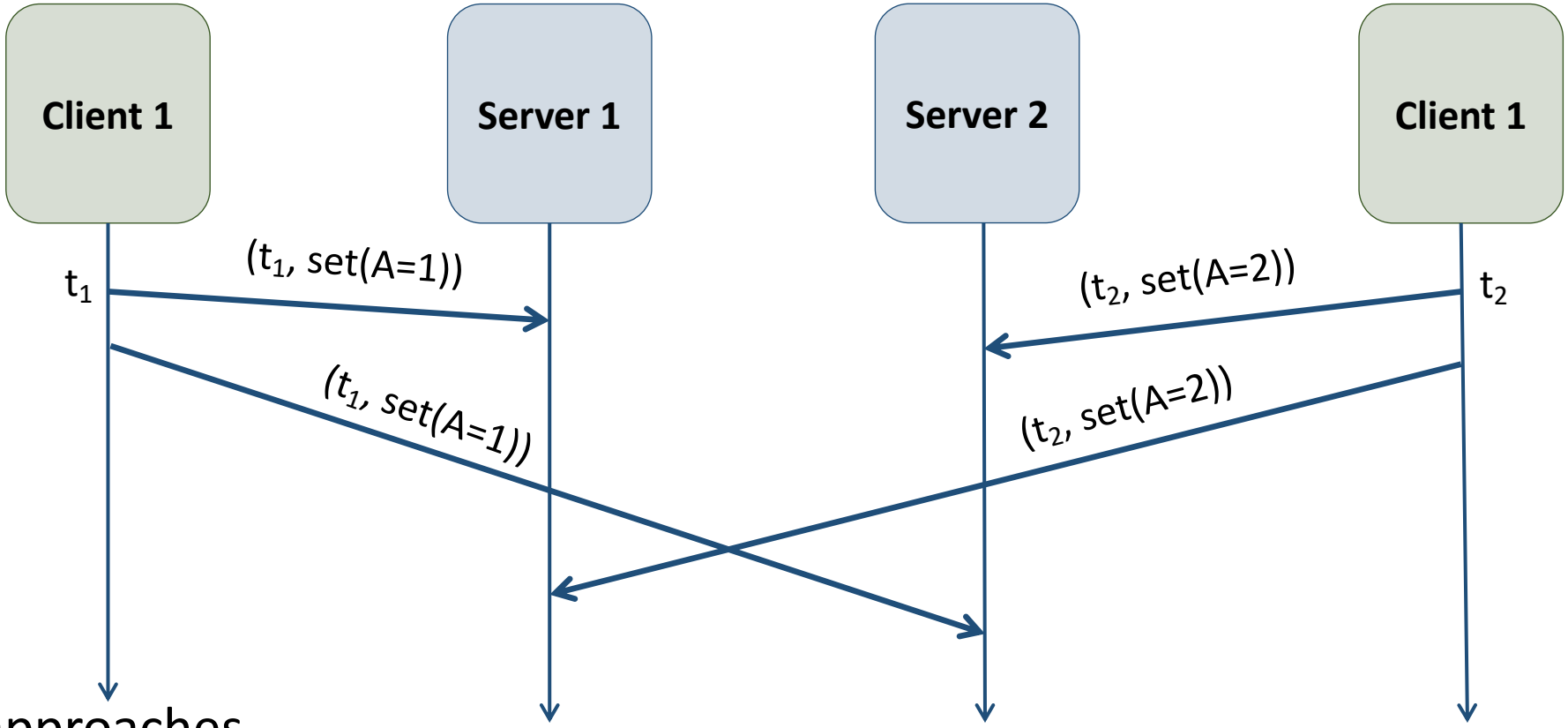
Reconciling replicas

- Replicas periodically communicate among themselves to check for any inconsistencies. (Anti-entropy protocol)



Update the record (data item) with latest timestamps.

Concurrent writes by different clients



Two common approaches

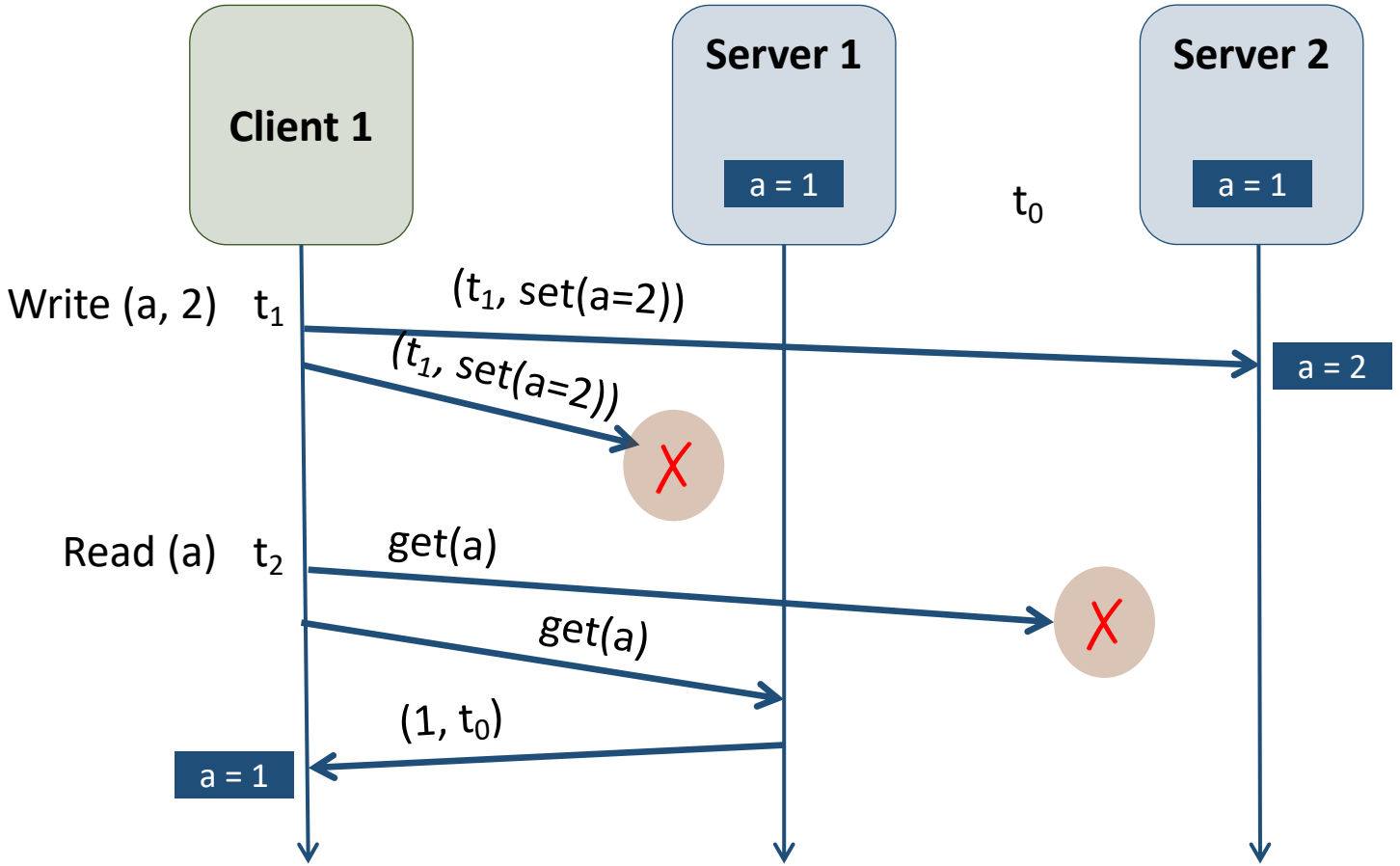
- Last writer wins (LWW)
  - Use timestamps with total order (e.g., Lamport clock)
- Multi-value register
  - Use timestamps with partial order (e.g., Vector clock).

Replica failure

- Observation
  - A replica may be unavailable due to network partition or node fault, etc.
- Assume
  - Each replica has a probability p of being unavailable (faulty)
  - Probability of all *n* replicas being faulty: p^n
  - Probability of >= 1 out of *n* replicas being faulty: $1 – (1 – p)^n$
- Example with p=0.01

Replicas n	P (>=1 faulty)	P (>= (n+1)/2 faulty)	P (all n faulty)
1	0.01	0.01	0.01
3	0.03	3×10^{-4}	10^{-6}
5	0.049	1×10^{-5}	10^{-10}
100	0.63	6×10^{-74}	10^{-200}

Read-after-write consistency

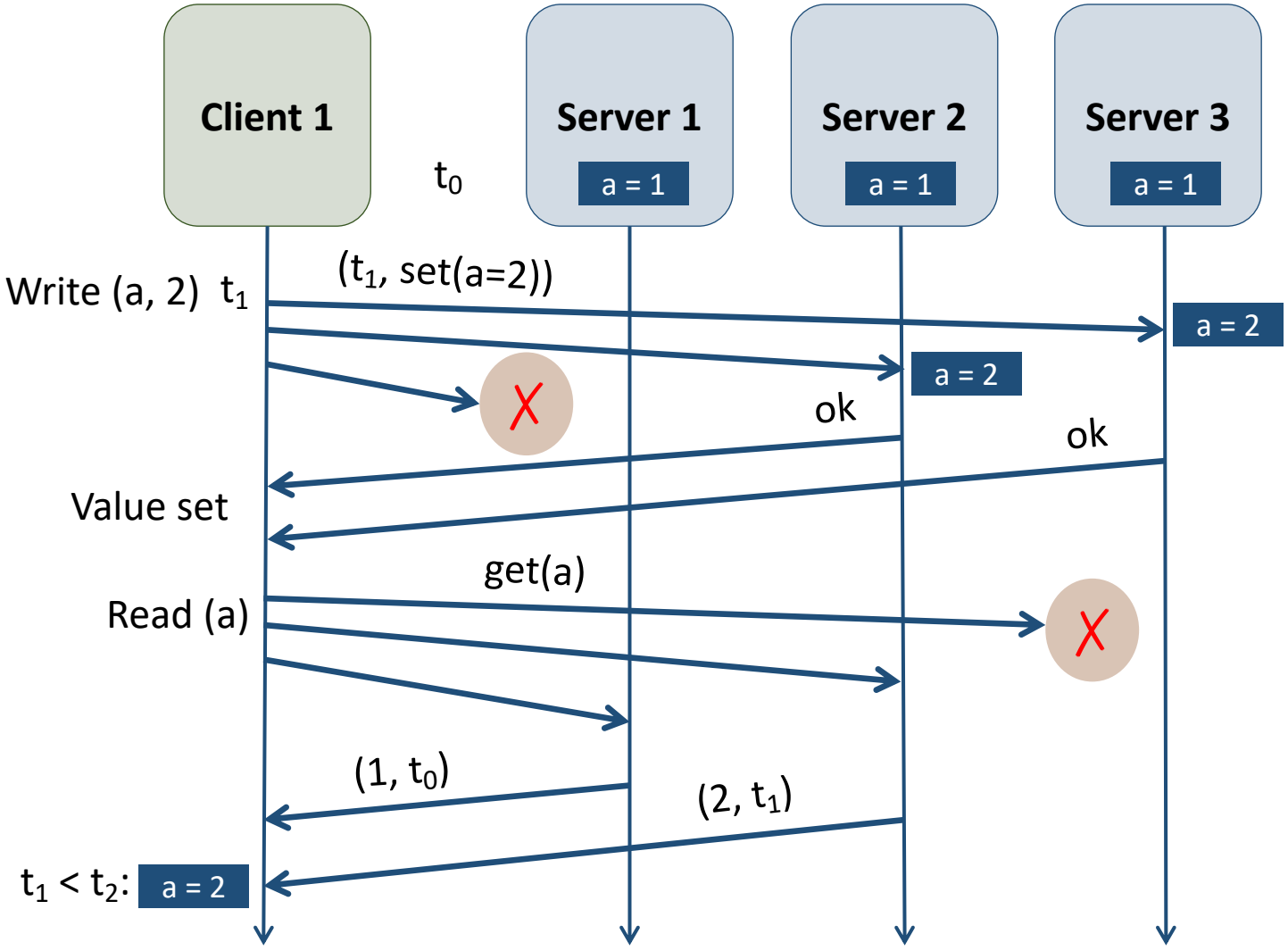


Replicated-write protocols (1)

- Write operations can be carried out at multiple replicas
- Active replication
  - Each replica has an associated process carrying out update operations
  - Drawback: operations need to be carried out in same order everywhere
    - ✧ Lamport timestamps (does not scale well in large systems)
    - ✧ Central coordinator (sequencer)
  - Problem with central coordinator: replicated invocations
    - ✧ Coordinator in each replica for managing invocations and replies.

Replicated-write protocols (2)

- Quorum-based protocols
  - Majority (among the replicas) based mechanisms
  - Ex: 2 out of 3 quorum
- Client looks at the timestamp to figure out latest value.

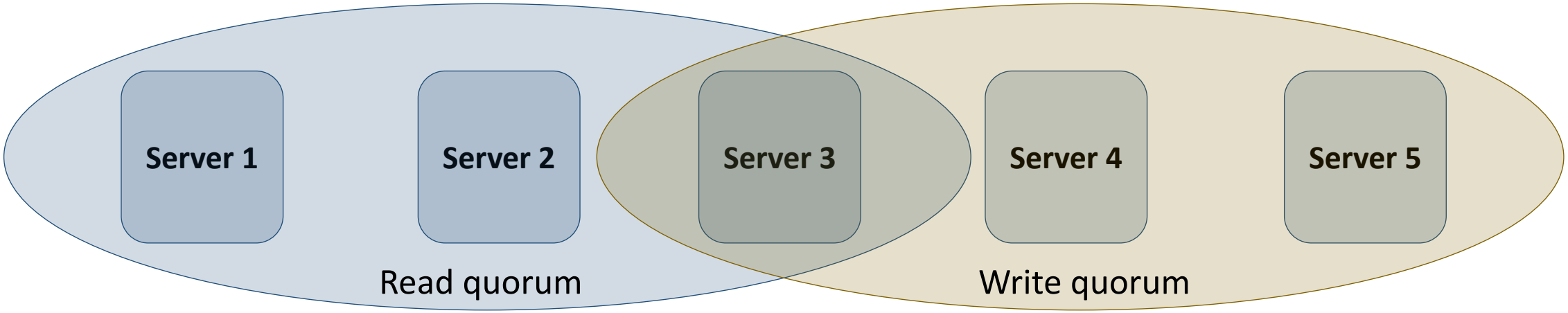


Quorum-based protocols (1)

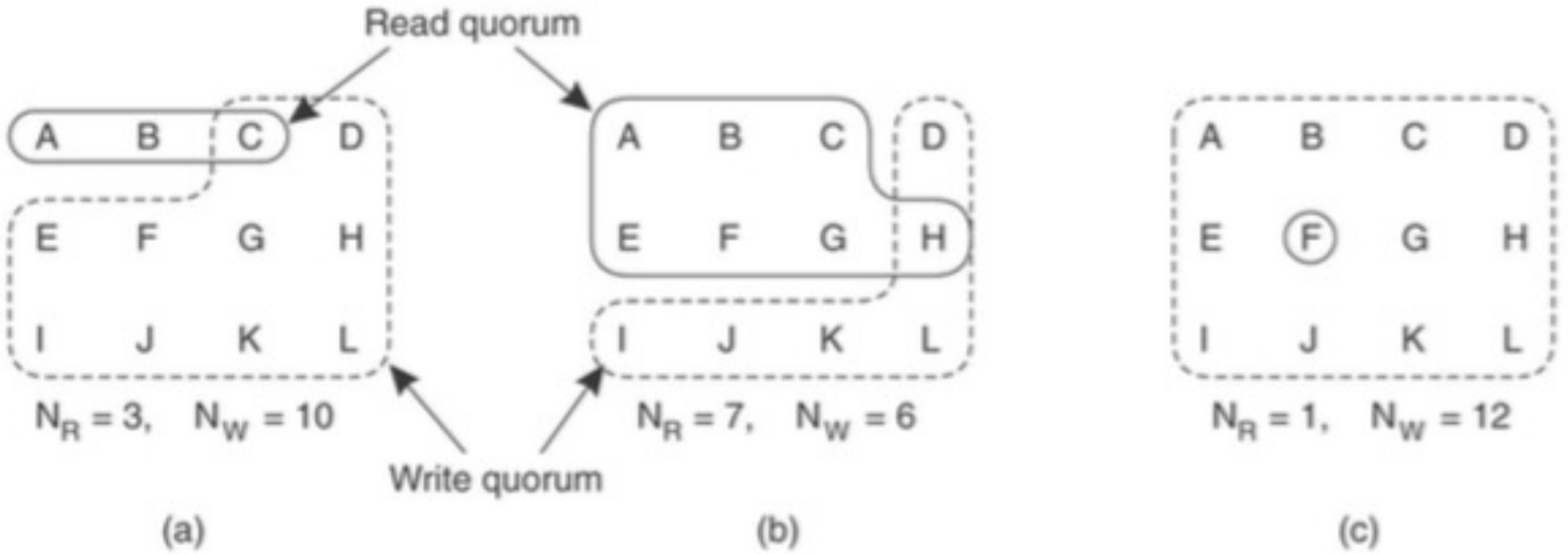
- Clients request and acquire permission of multiple servers before either reading or writing replicated data item
- Gifford's scheme for N replicas
  - Read quorum: client needs permission from arbitrary N_R servers
    - ✧ Prevents read-write conflicts
  - Write quorum: client needs permission from arbitrary N_W servers
    - ✧ Prevents write-write conflicts
  - Constraints:
    - $N_R+N_W > N$
    - $N_W > N/2$

Quorum-based protocols (2)

Read quorum and write quorum share >= 1 replica

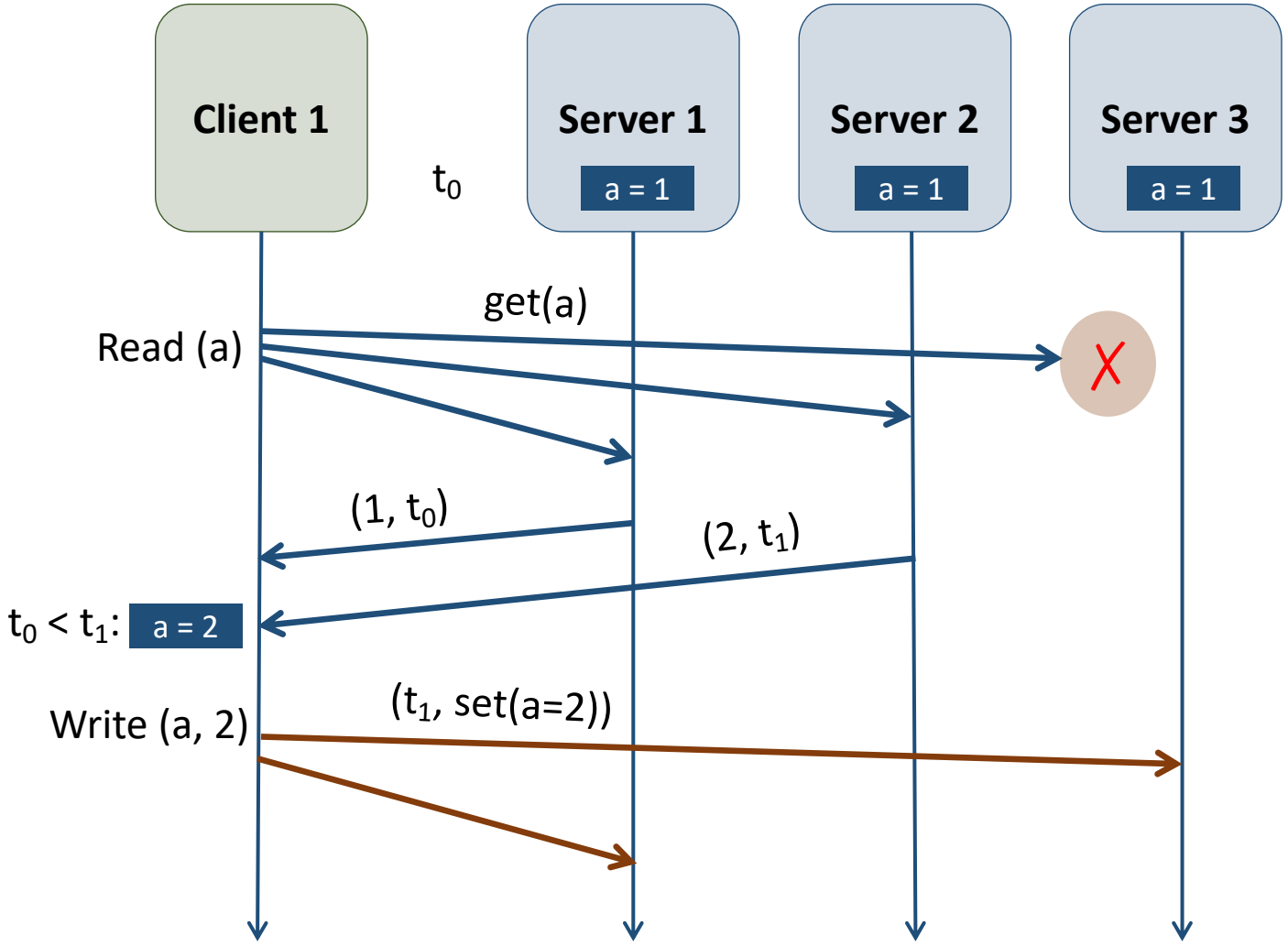


Quorum-based protocols (3)



a) Correct choice of read and write set
b) May lead to write-write conflicts
c) Correct choice known as ROWA (read one, write all)

Read repair



- Client-support
  - Update (t_1, 2) is more recent than (t_0, 1) since $t_0 < t_1$
  - Client helps propagate (t_1, 2) to other replicas.