Low-Latency Multi-Datacenter Databases using Replicated Commit

Hatem Mahmoud, Faisal Nawab, Alexander Pucher, Divyakant Agrawal, Amr El Abbadi

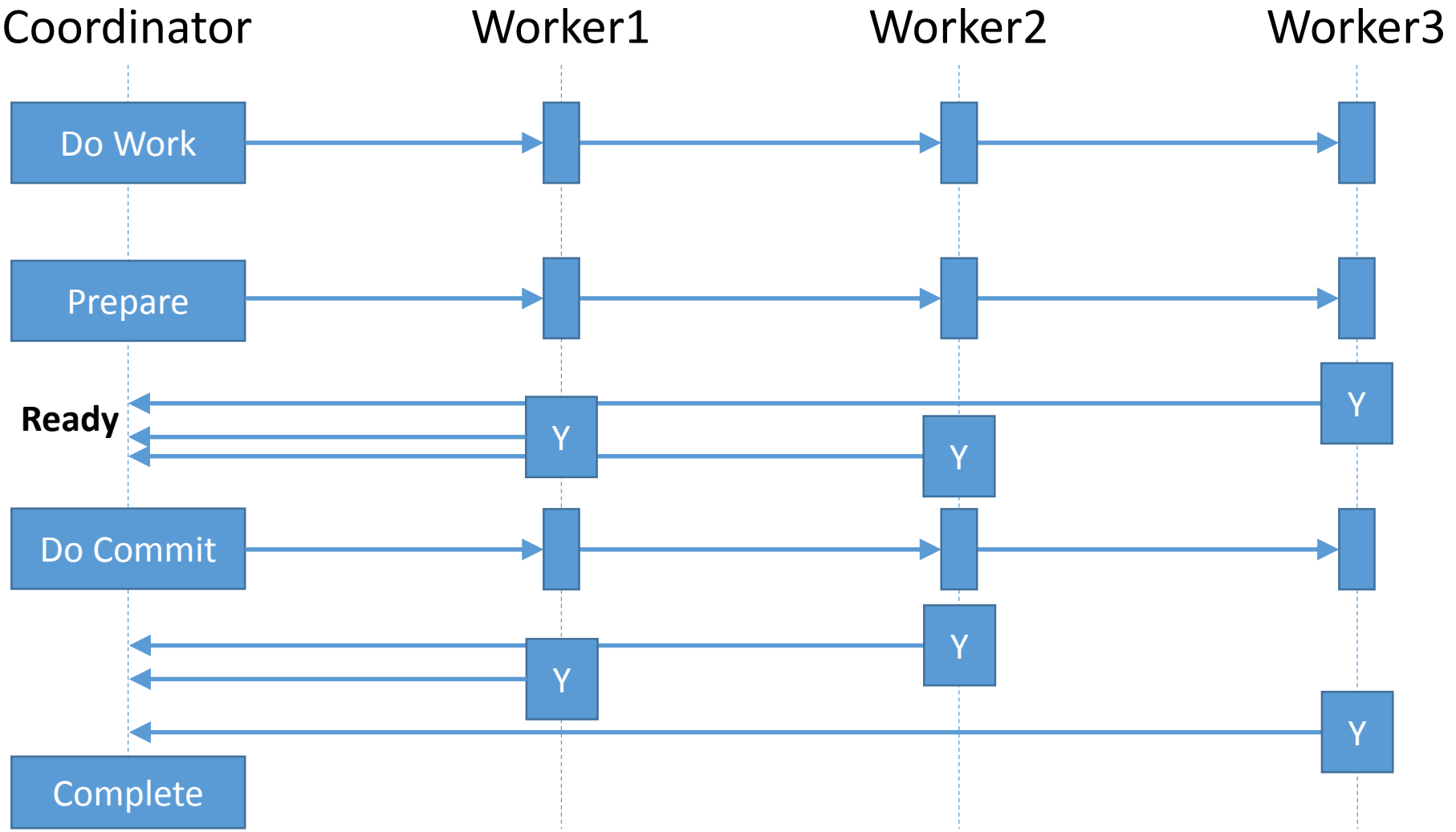
UCSB

Presented by Ashutosh Dhekne

Main Contributions

- Reduce cross data center communication trips
- Compare Replicated Commit with Replicated Log
- Extensive experimental study for evaluation of the approach
  - Number of data centers involved
  - Read vs Write operations
  - Number of operations per transaction
  - Data objects in the database
  - Effect of used shards

2 Phase Commit

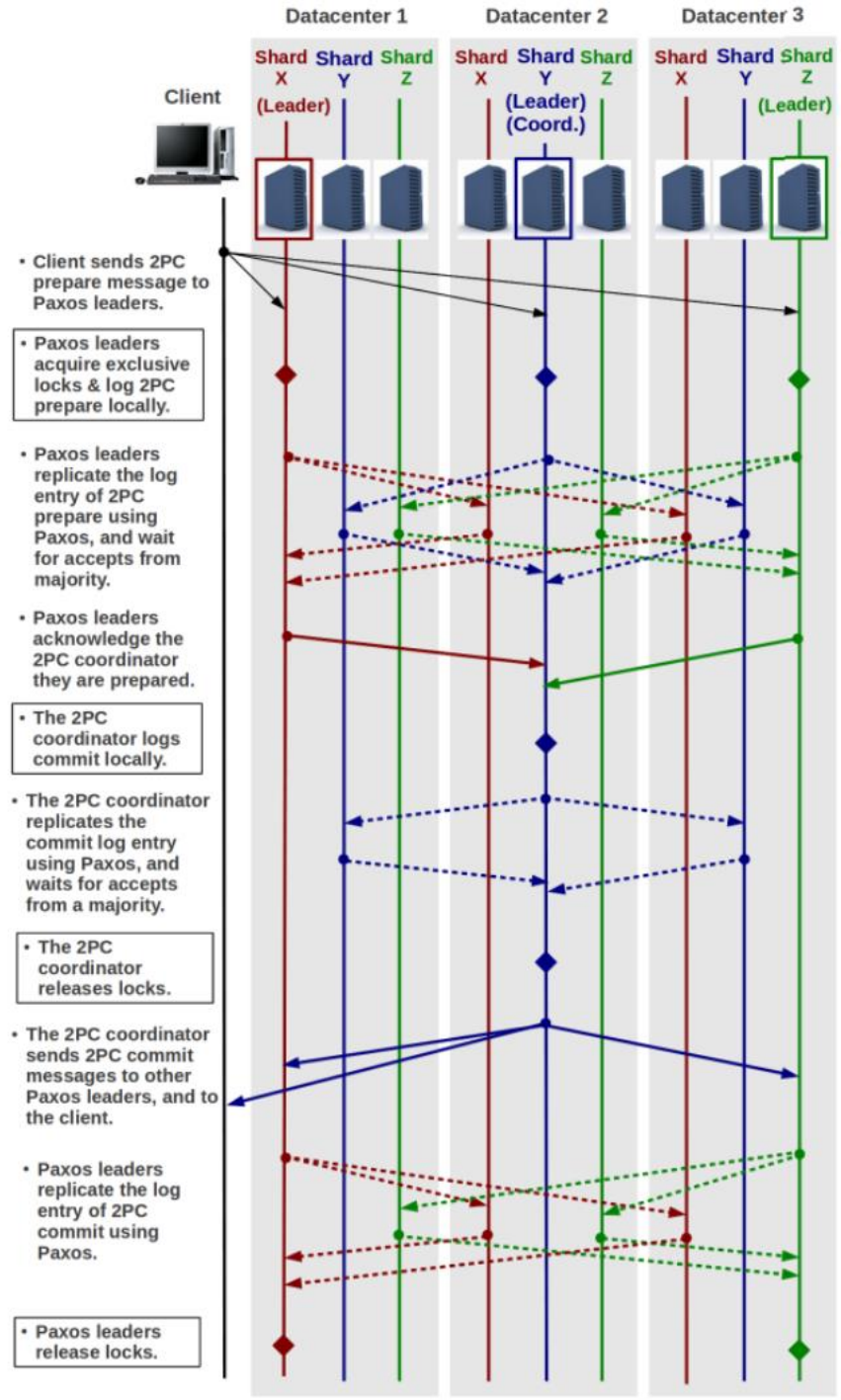


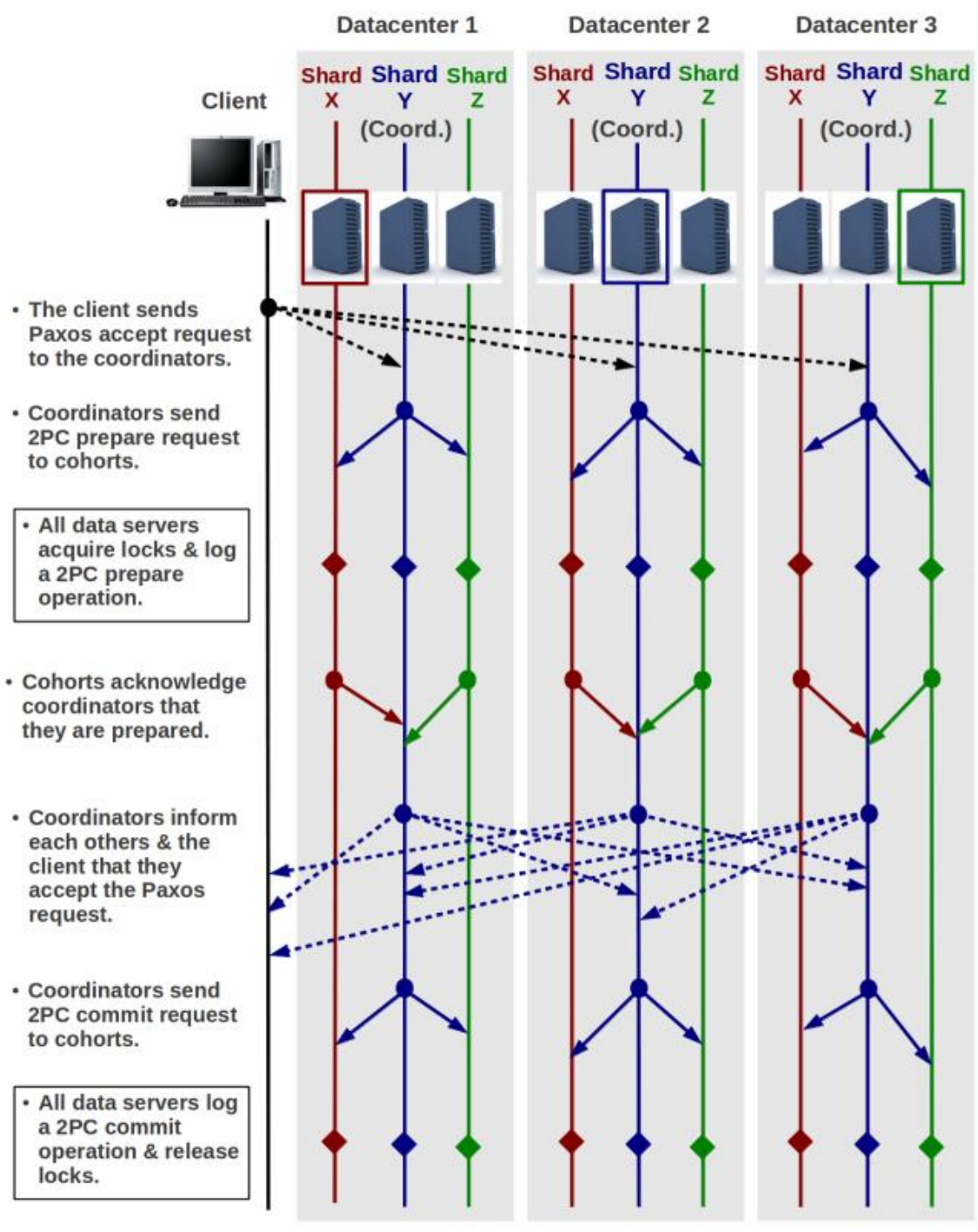
Why merge 2PC and Paxos?

- Two phase commit is very good for reaching consensus
- Agreement is guaranteed – only coordinator decides
- Validity is guaranteed – commit happens only if everyone is ready.
- Termination is guaranteed – no loops possible

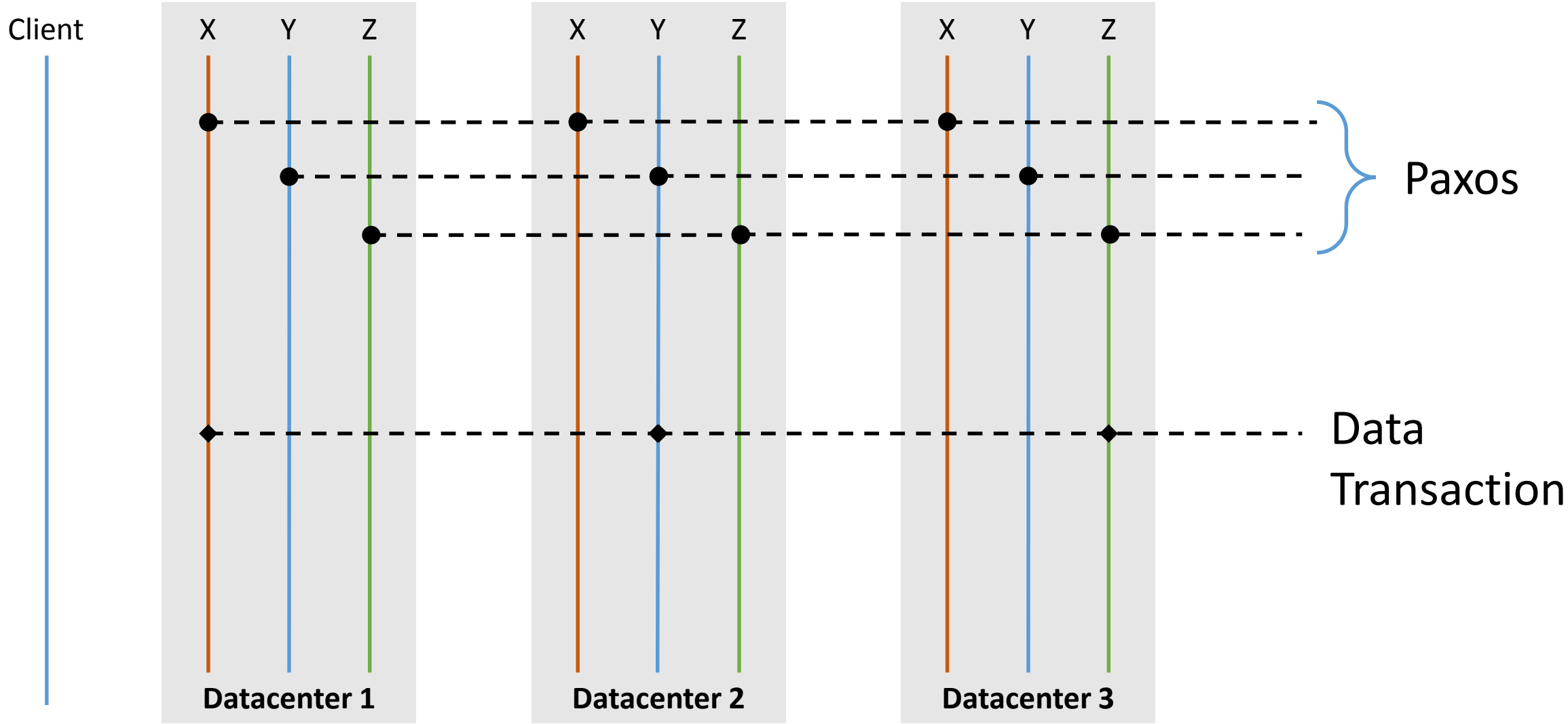
- But Failures can wreak havoc
- Paxos is good even if failures occur – quorum based, majority wins

Replicated Log vs Commits

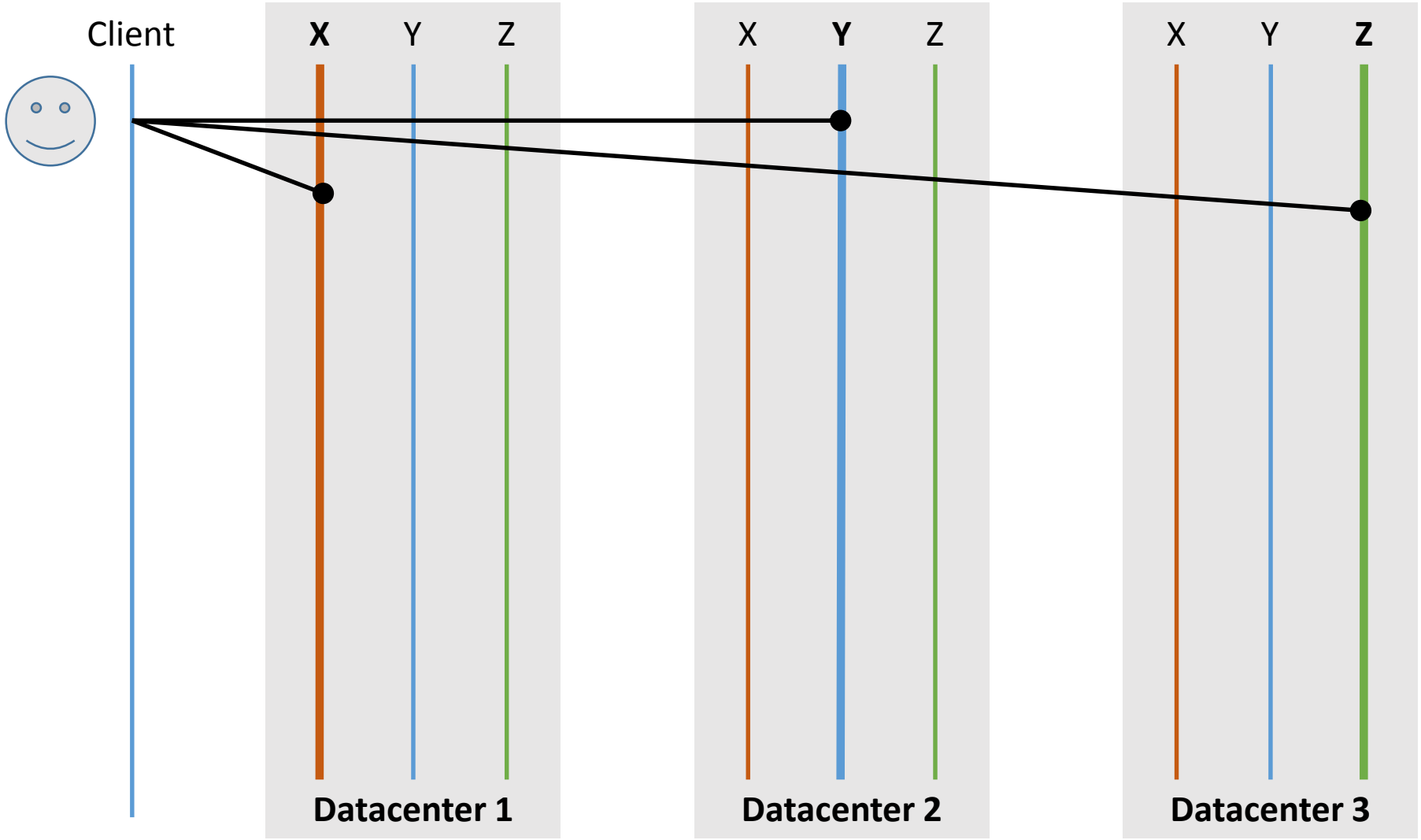




Replicated Log – Step by Step

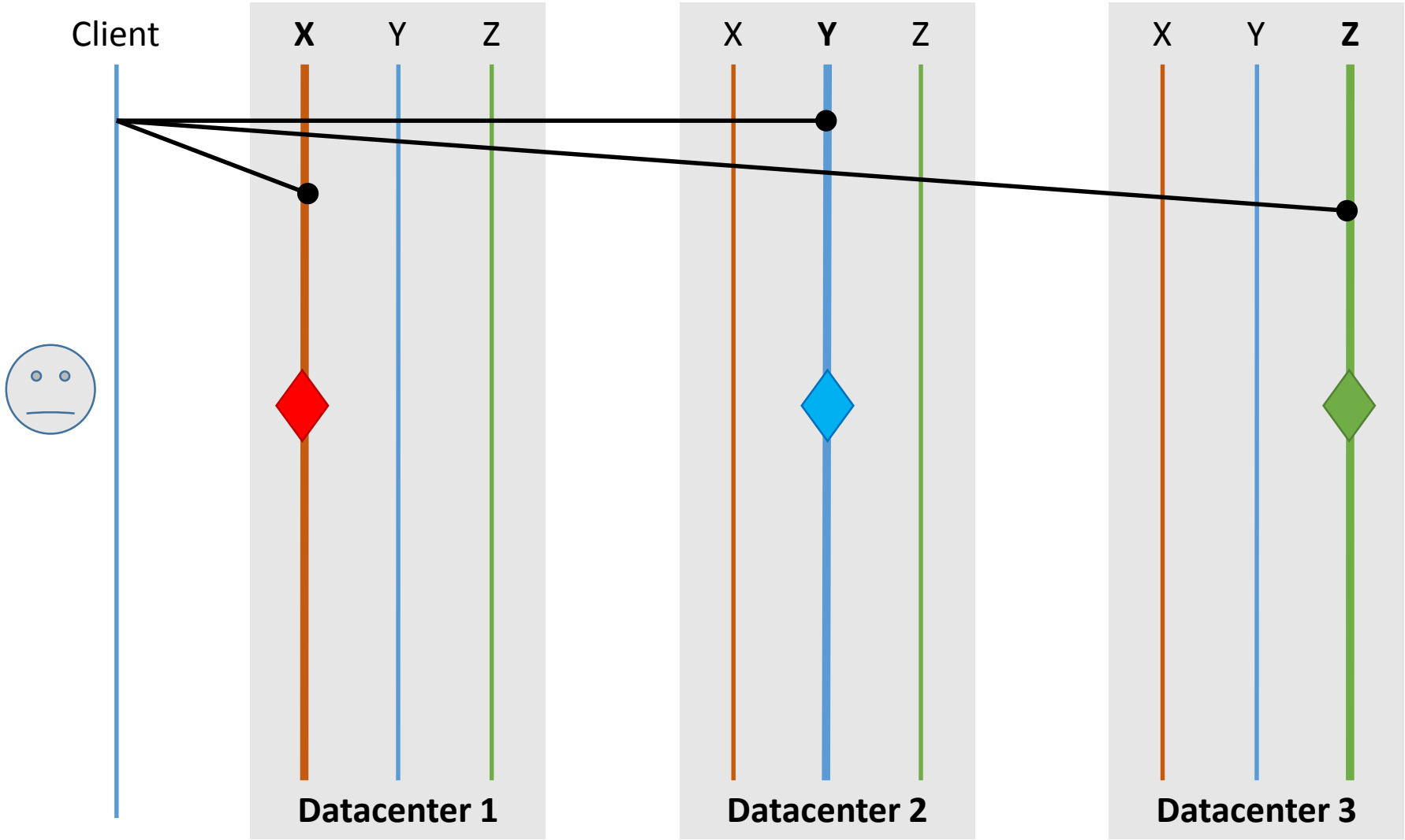


Replicated Log – Step by Step



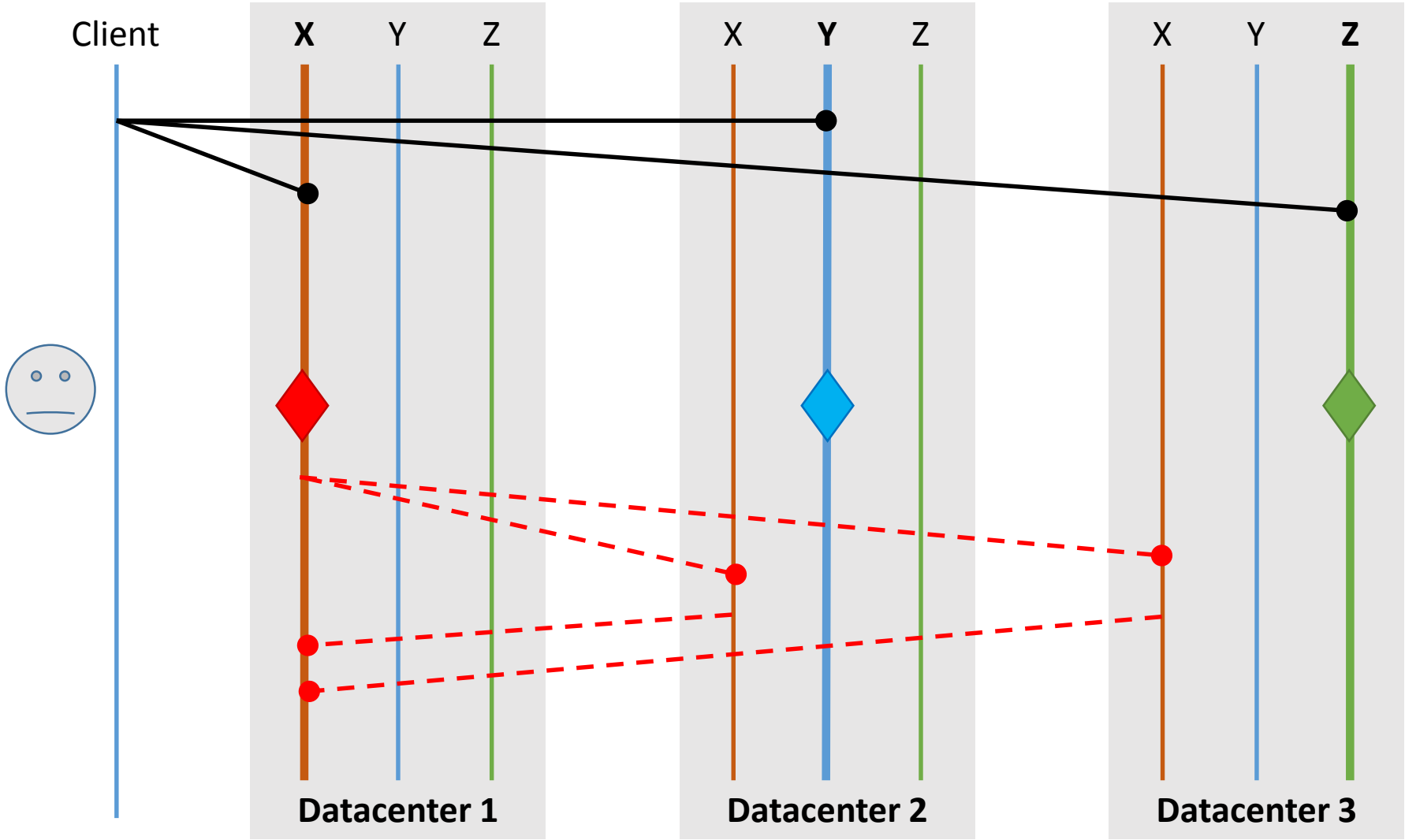
2PC **Prepare** message to Paxos Leaders

Replicated Log – Step by Step



Acquire exclusive locks and **log** 2PC Prepare locally

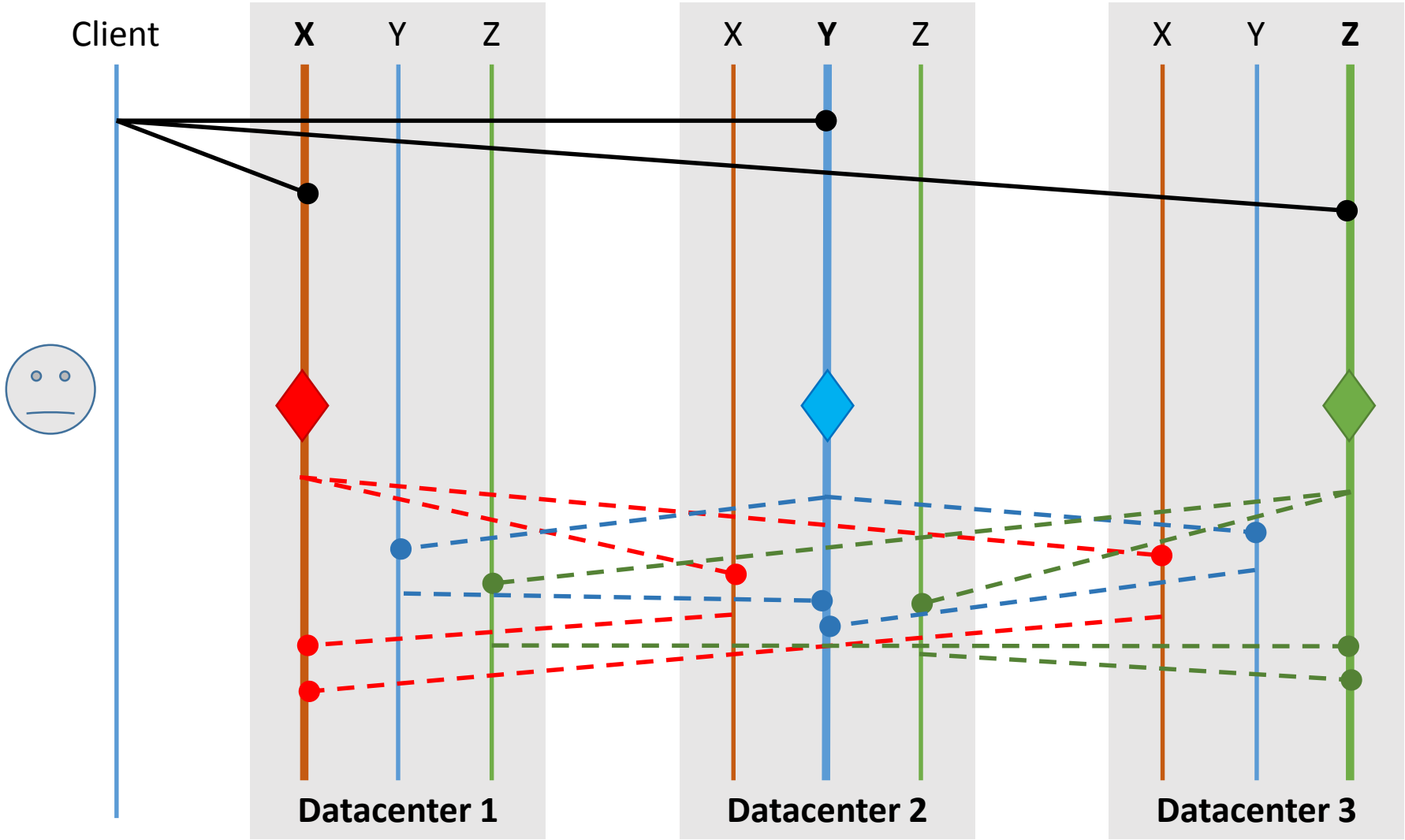
Replicated Log – Step by Step



Leader **informs** same shard in other datacenters

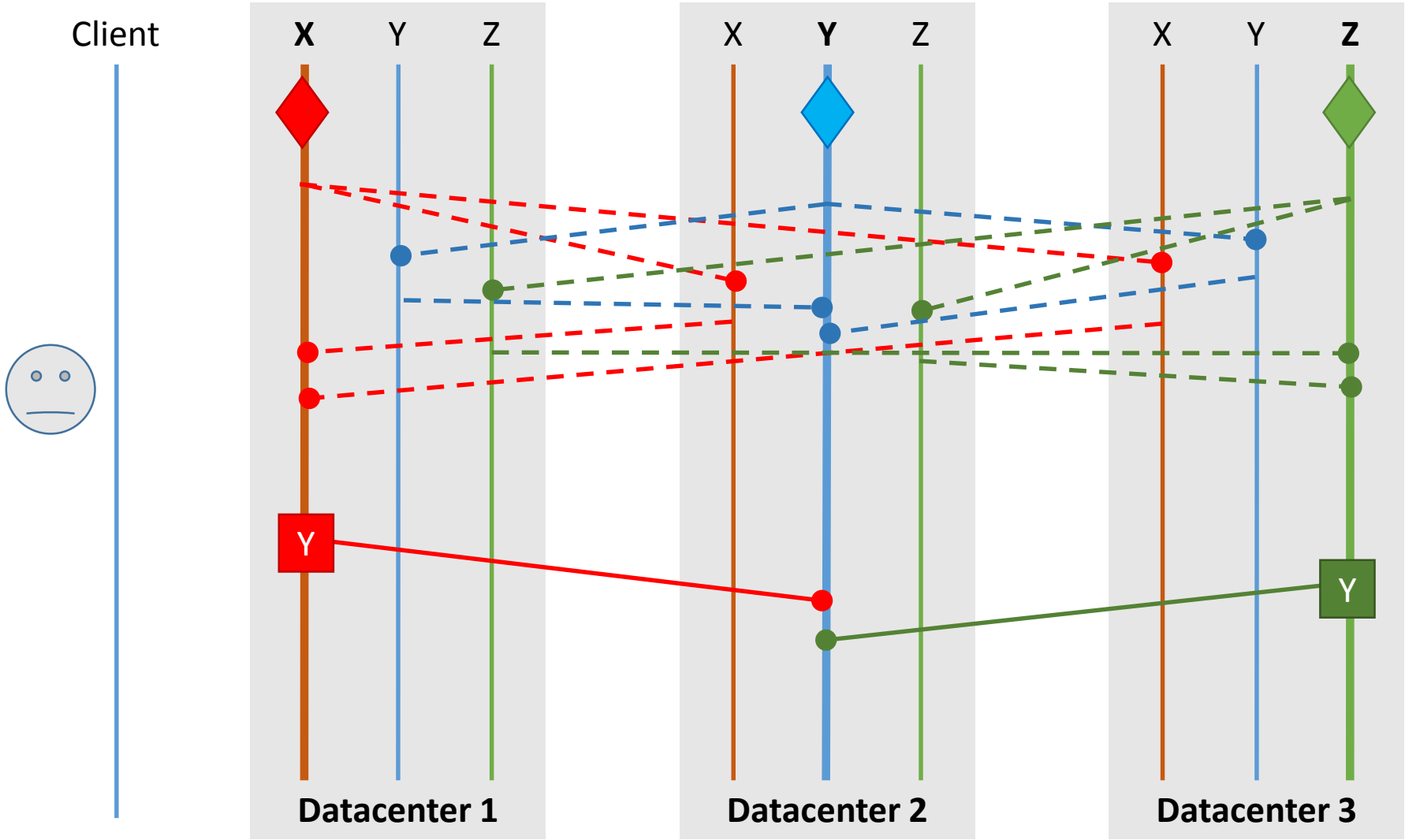
Those shards reply **accept**

Replicated Log – Step by Step



All of those messages together

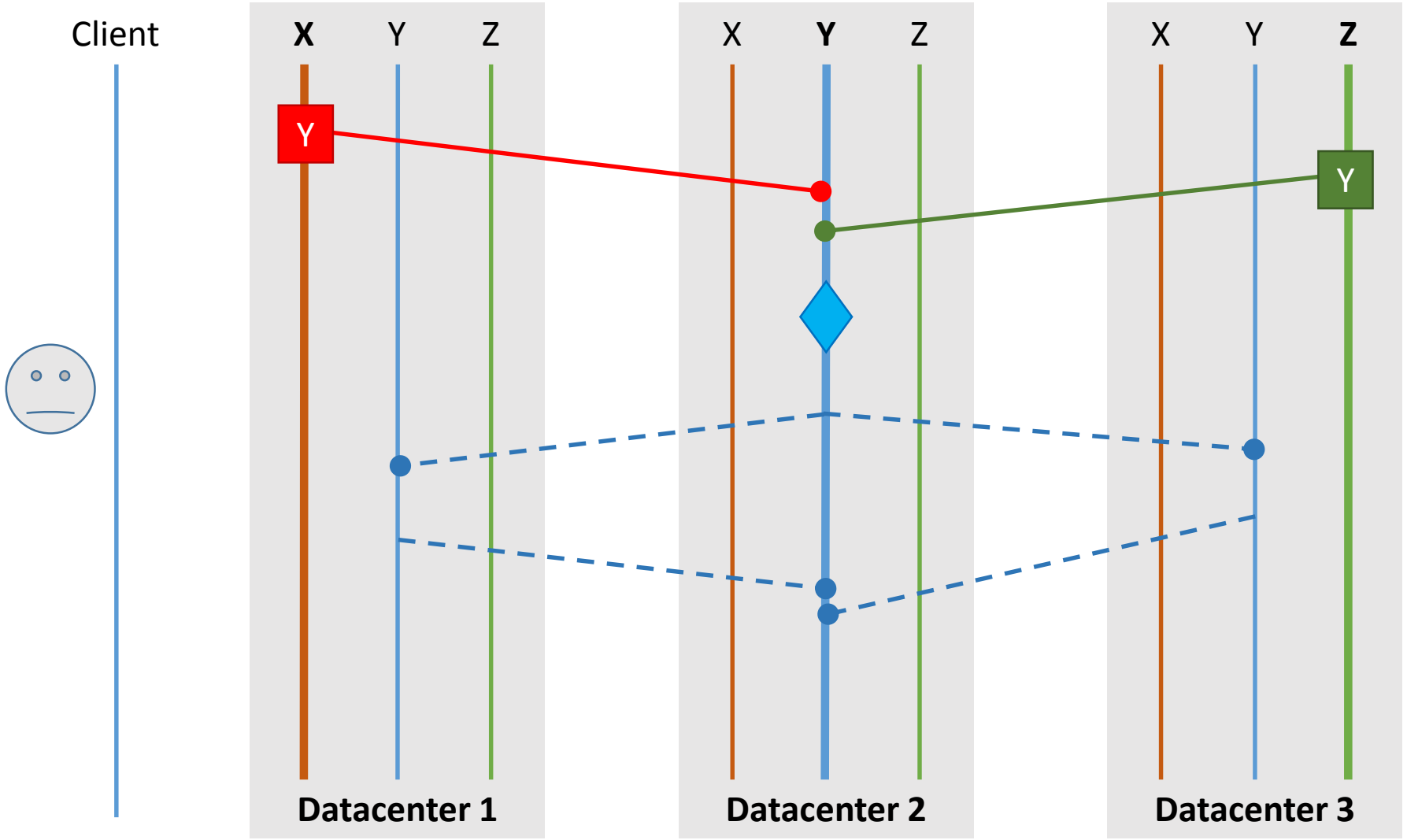
Replicated Log – Step by Step



Paxos leaders know status of their shards in other data centers

Paxos leaders ack the coordinator that they are **ready**

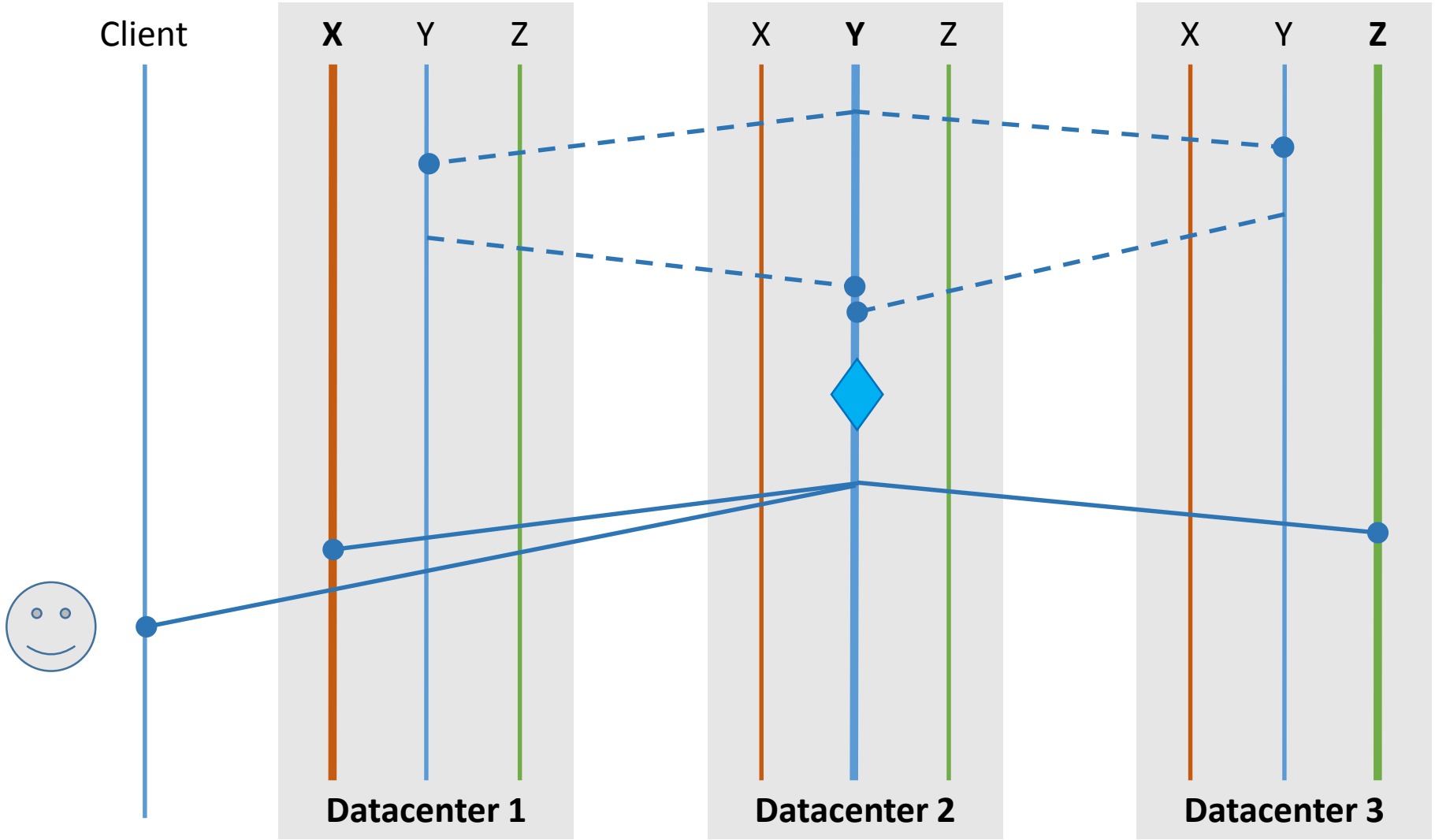
Replicated Log – Step by Step



Coordinator can now **log commit**

Then asks it's peer shards to commit and **receives acks**

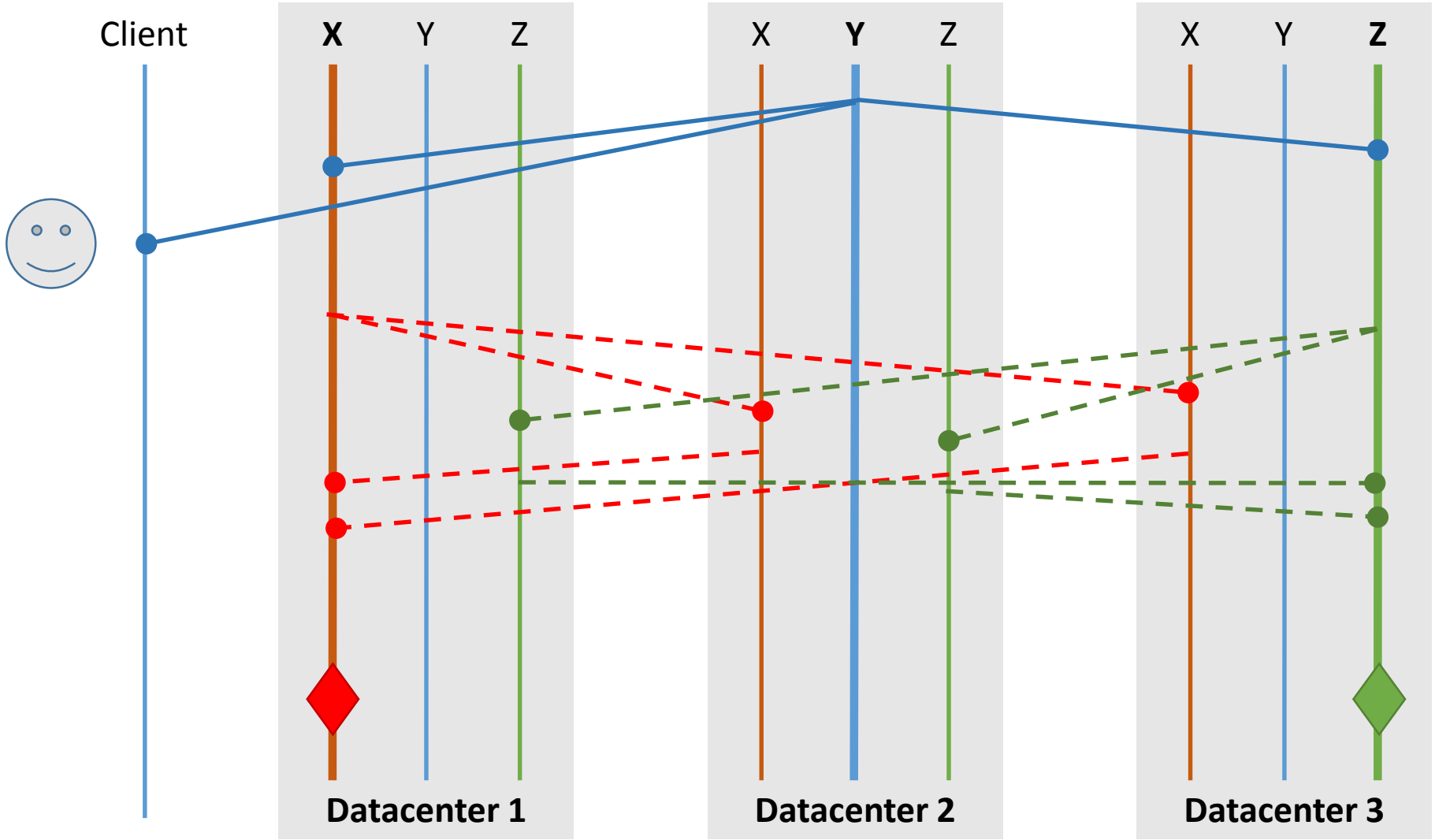
Replicated Log – Step by Step



Coordinator **releases lock**

Informs other leaders that **commit is made**

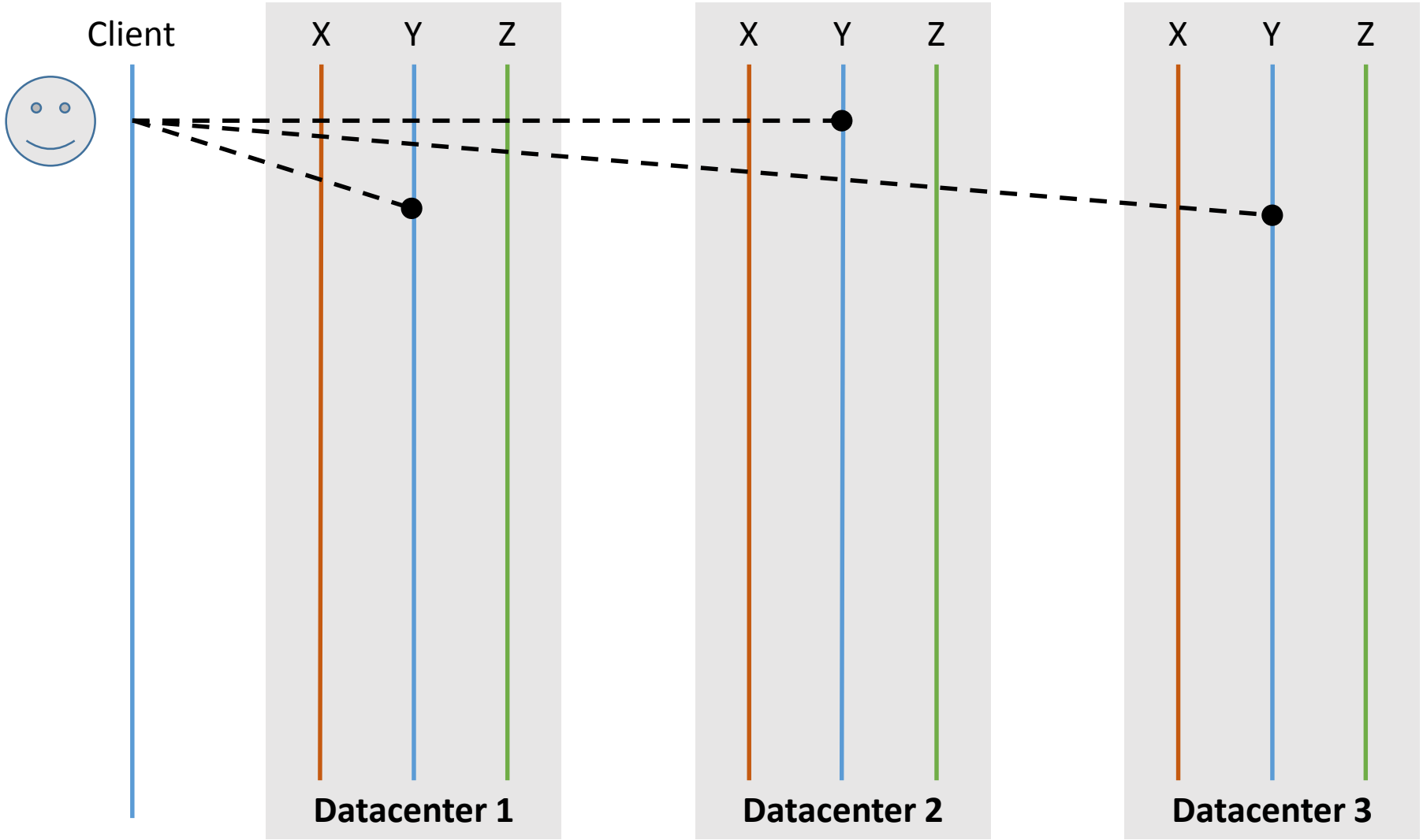
Replicated Log – Step by Step



Other leaders go in another frenzy to intimate their shards on other data centers

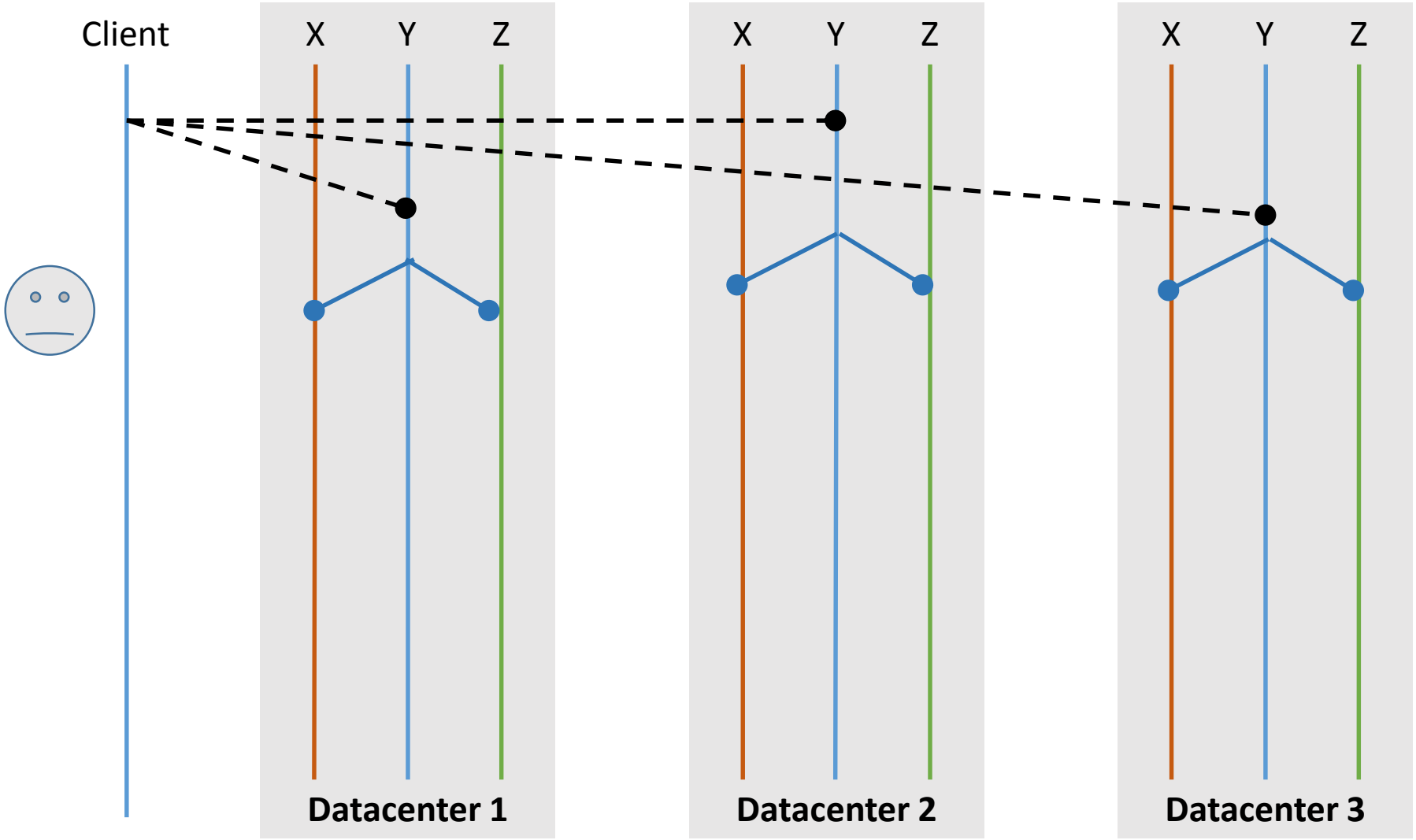
Finally, **locks are released**

Replicated Commit – Step by Step



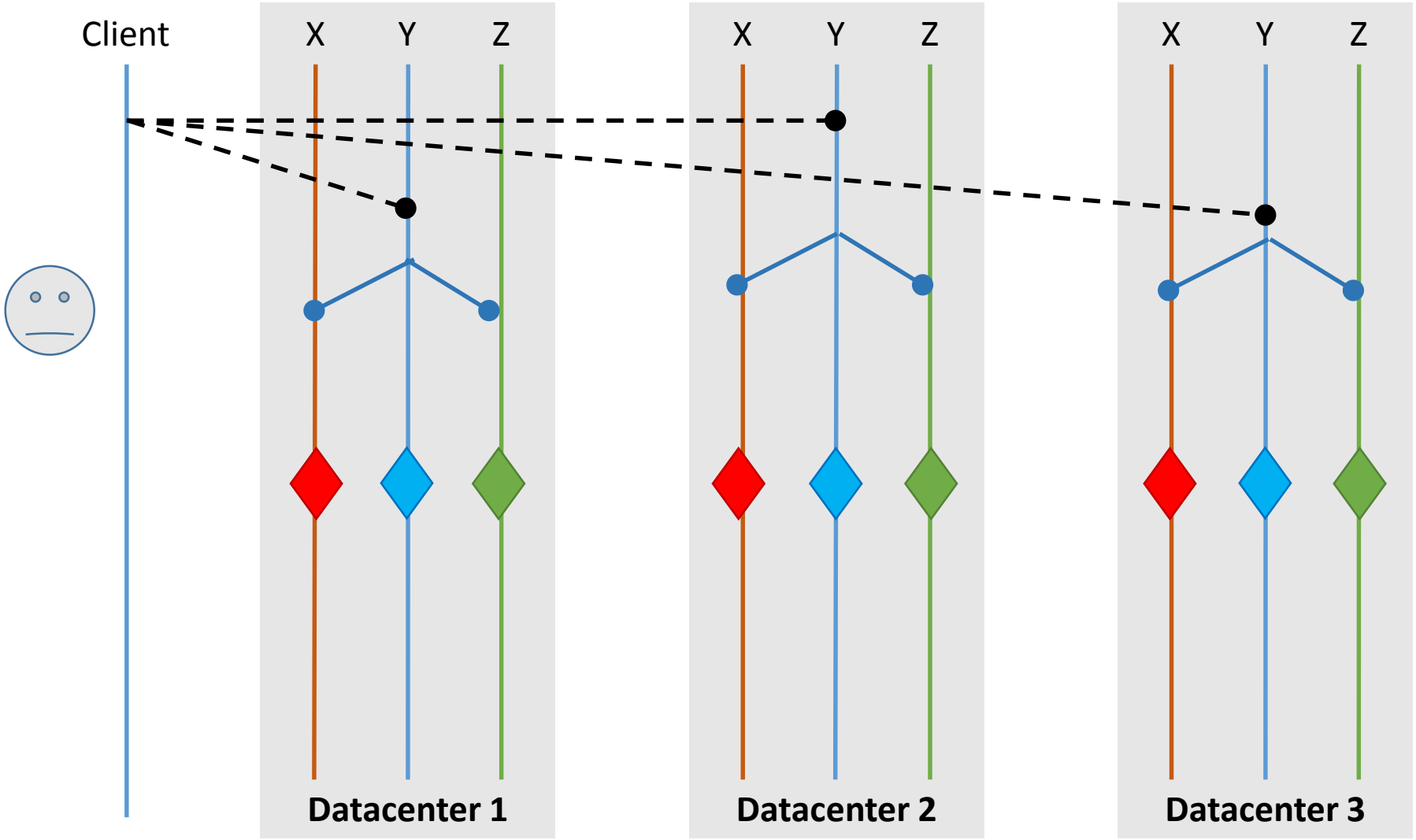
Paxos accept request to coordinators

Replicated Commit – Step by Step



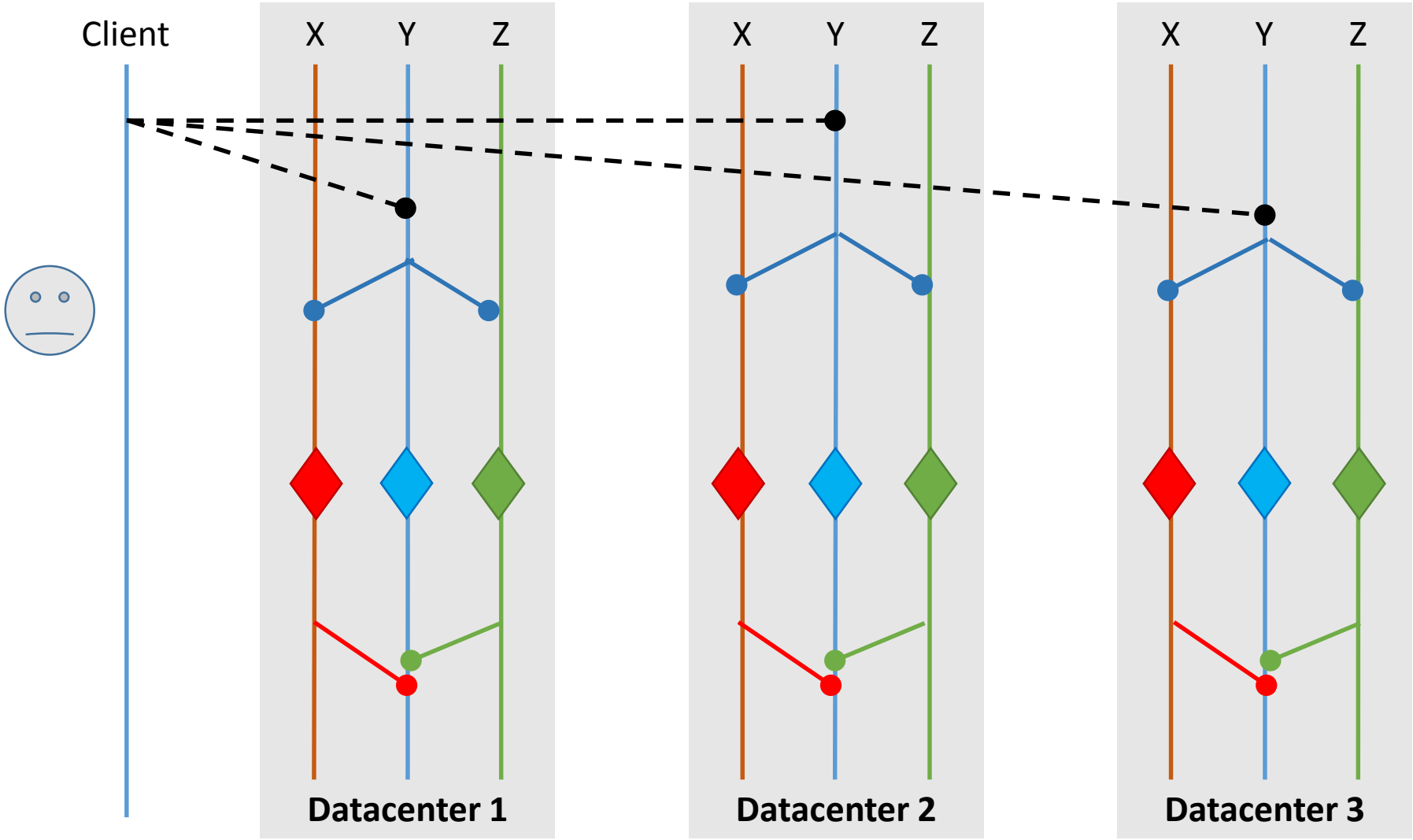
2PC **prepare** message is sent to other shards locally

Replicated Commit – Step by Step



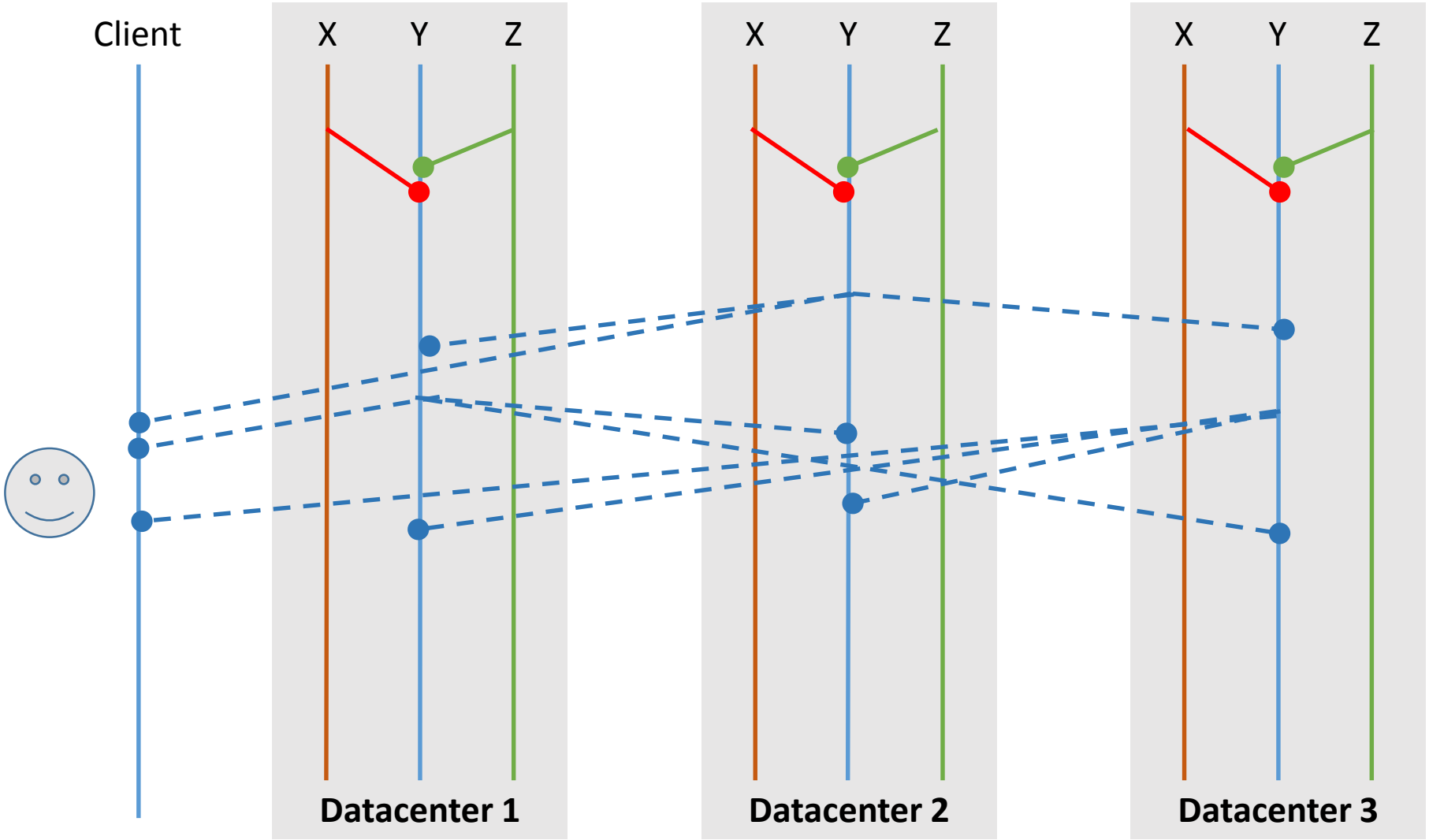
All acquire lock and **log 2PC prepare** message

Replicated Commit – Step by Step



All reply **ready** to the coordinator

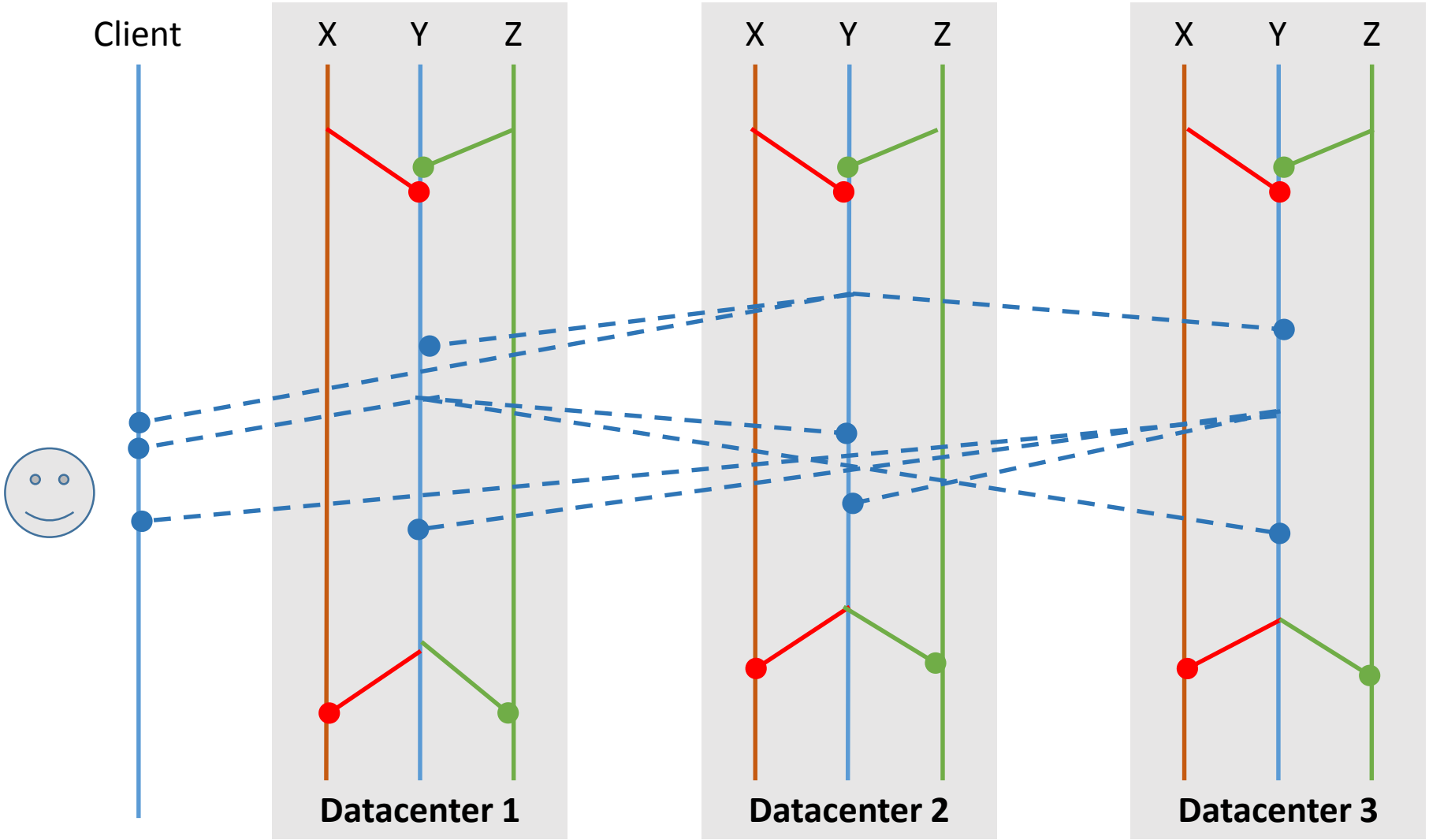
Replicated Commit – Step by Step



All coordinators inform each other and the client.

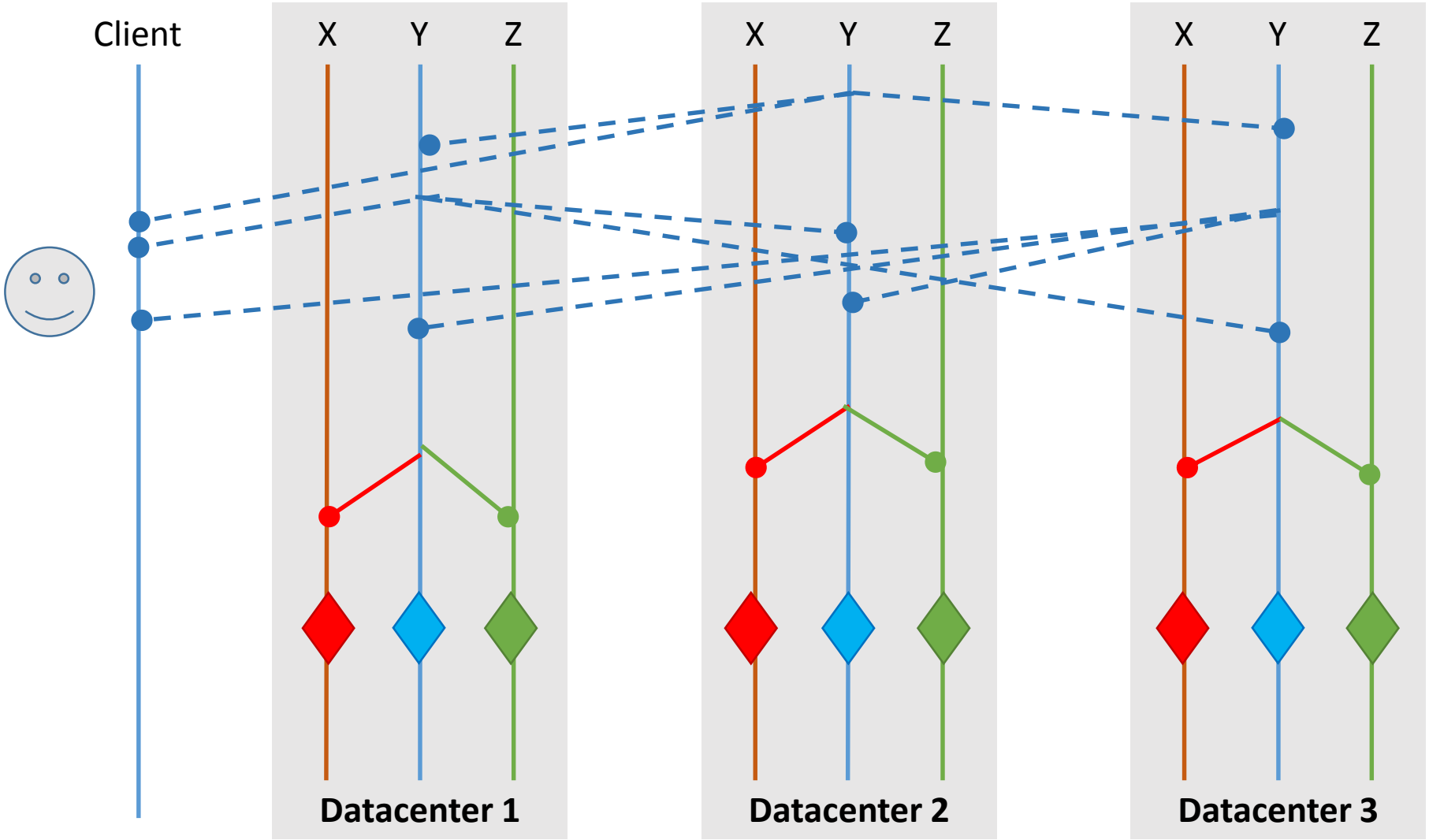
Client keeps quorum info

Replicated Commit – Step by Step



Commit is sent to all shards locally

Replicated Commit – Step by Step



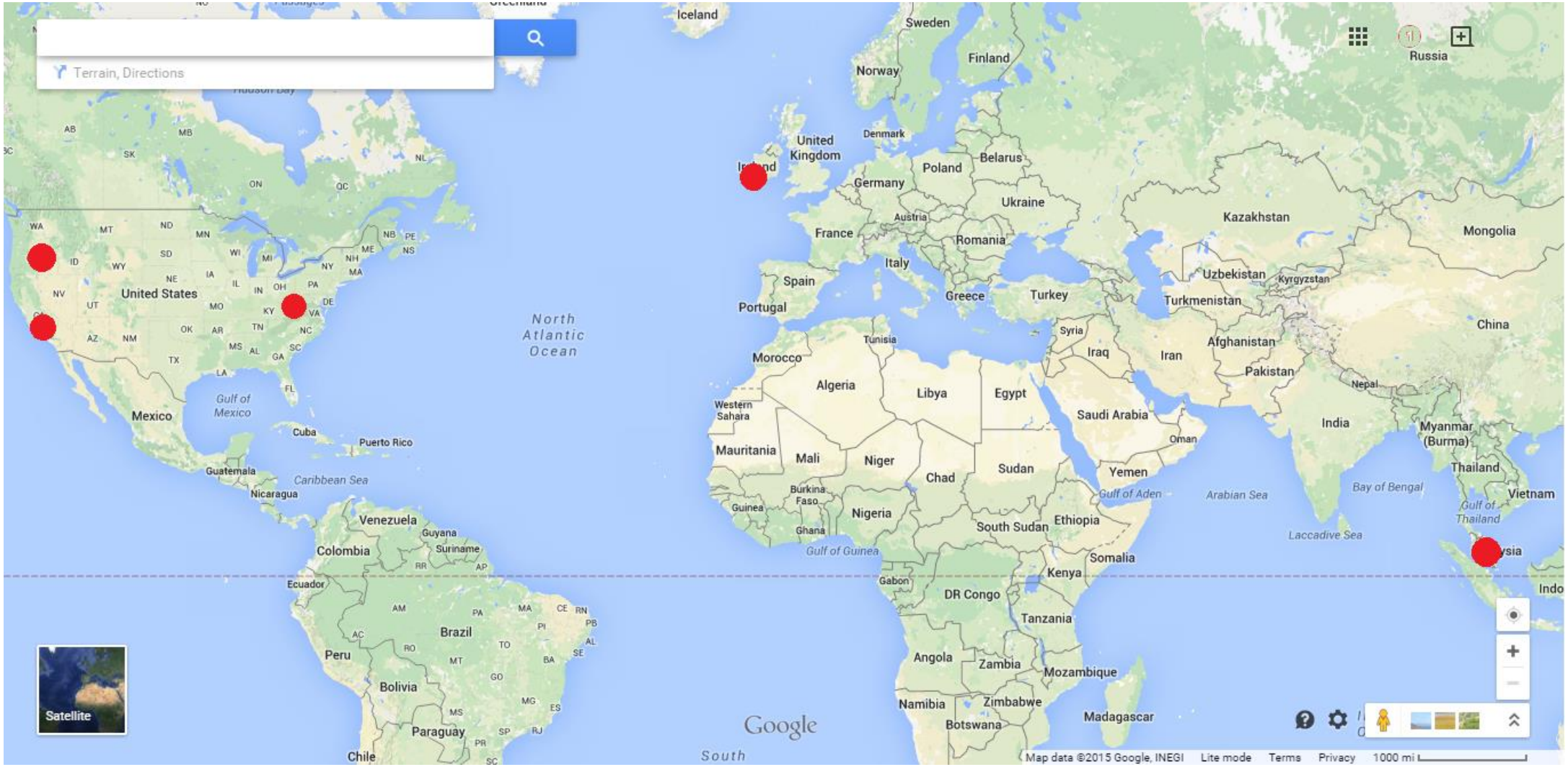
Finally, locks are released

Replicated Log Vs Replicated Commit

- Number of messages – Replicated Log requires a lot of inter-datacenter message transfers
- Latency – Network links across the continents are not at speed of light
- Access Locality – Messages exchanged locally are very fast

- A solution to higher latency – reduce the number of cross datacenter messages
- Replicated Commit achieves this

Experiments



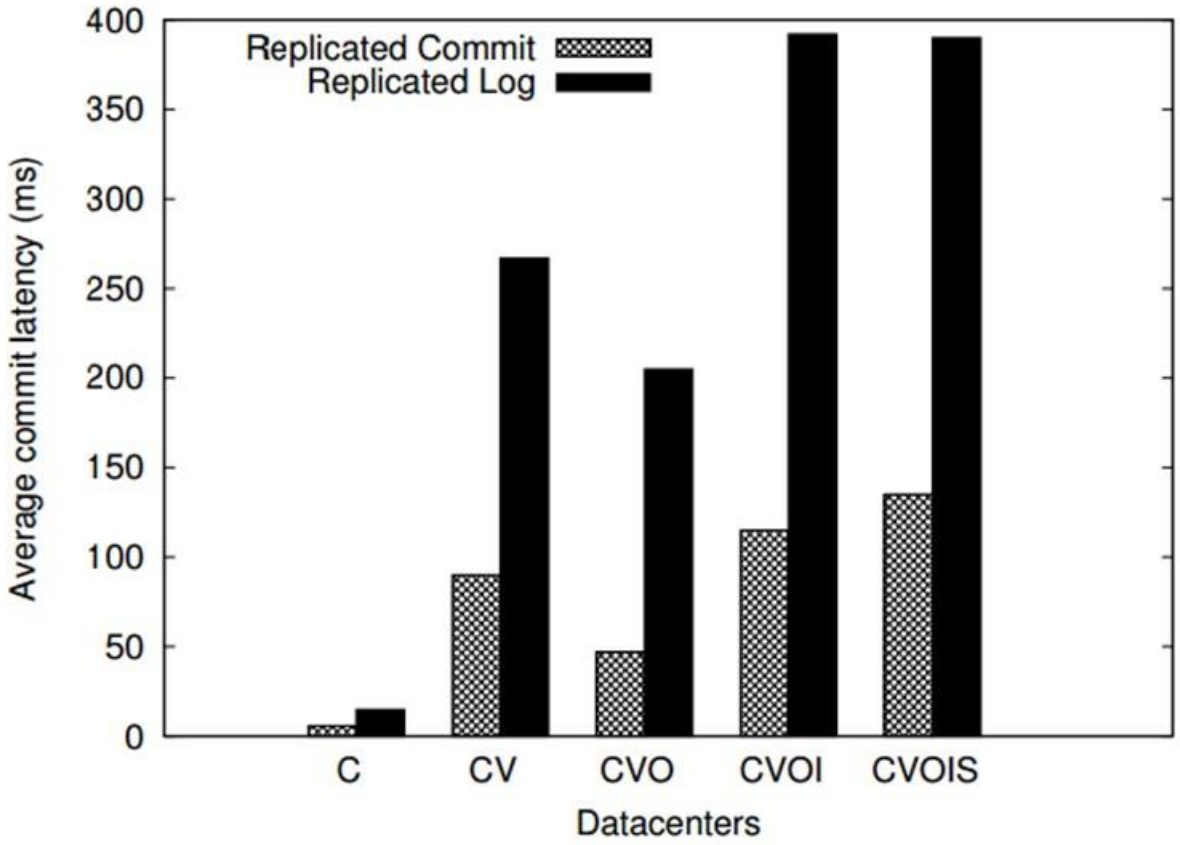
Experimental Setup

- 5 Data centers
- 2500 transactions
- 3000 items in the database, 1000 in each shard
- 3 shards per data center
- 50 operations per second in each datacenter

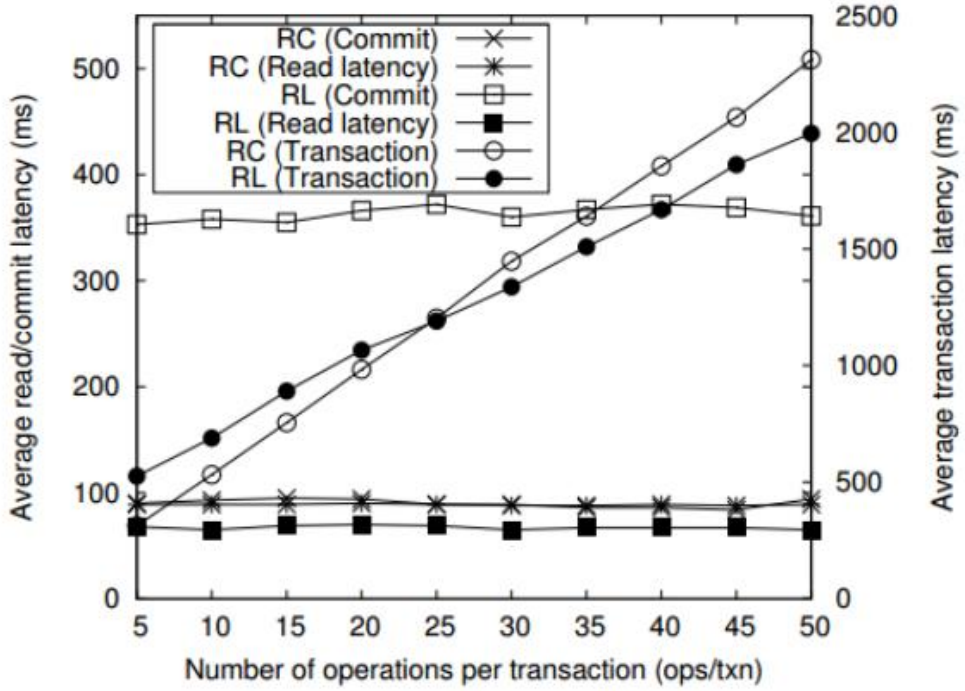
Average Commit Latency – various datacenters

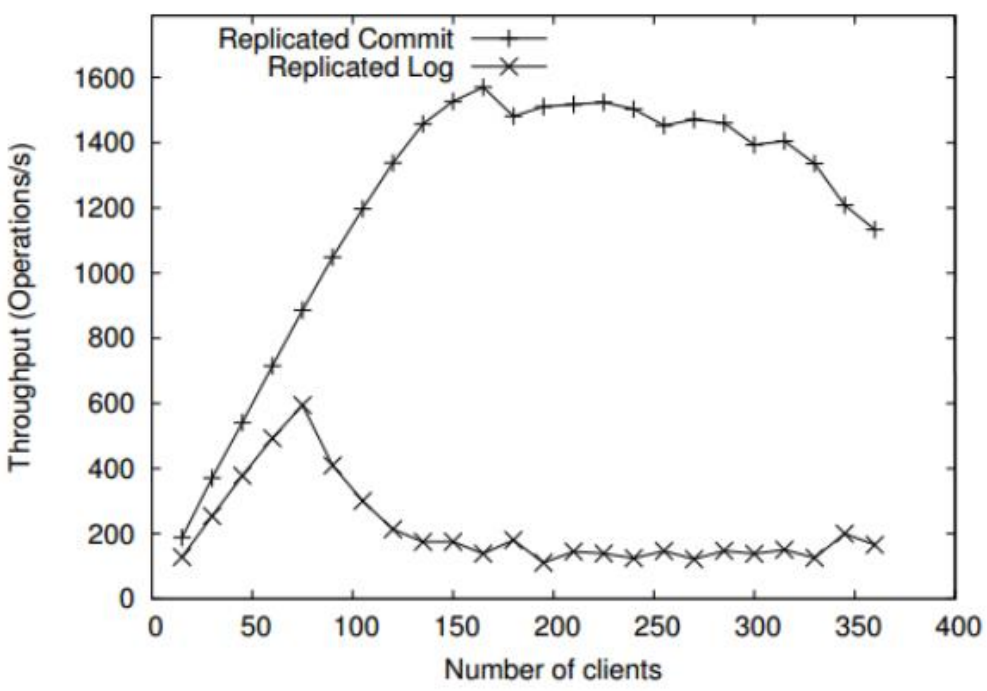
- Replicated Commit provides much faster responses than Replicated Logs

- Combination of servers from different regions versus combination of servers from same region has large impact

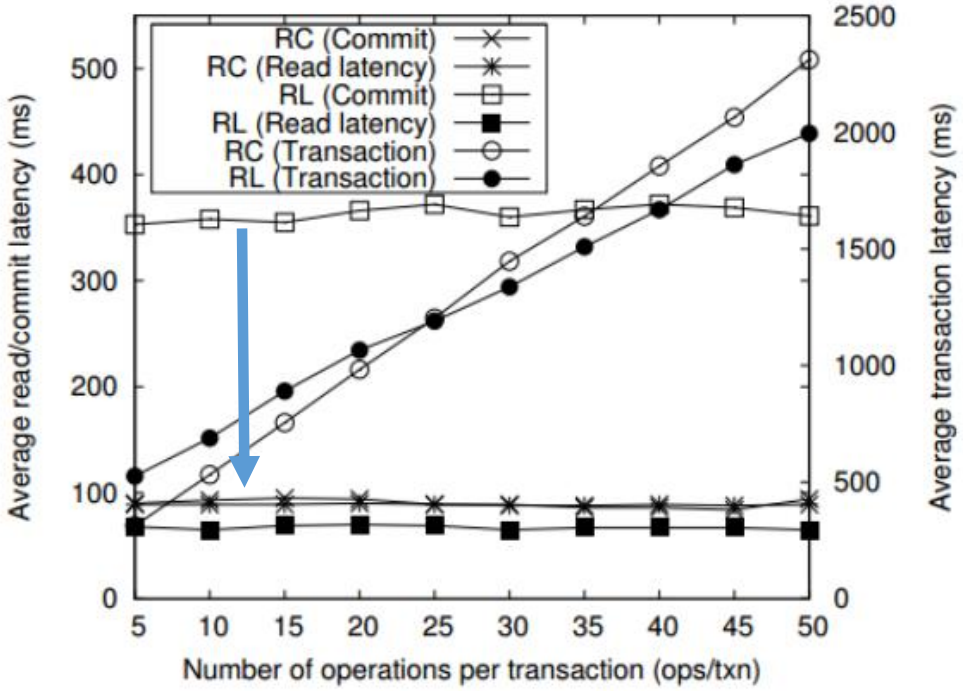


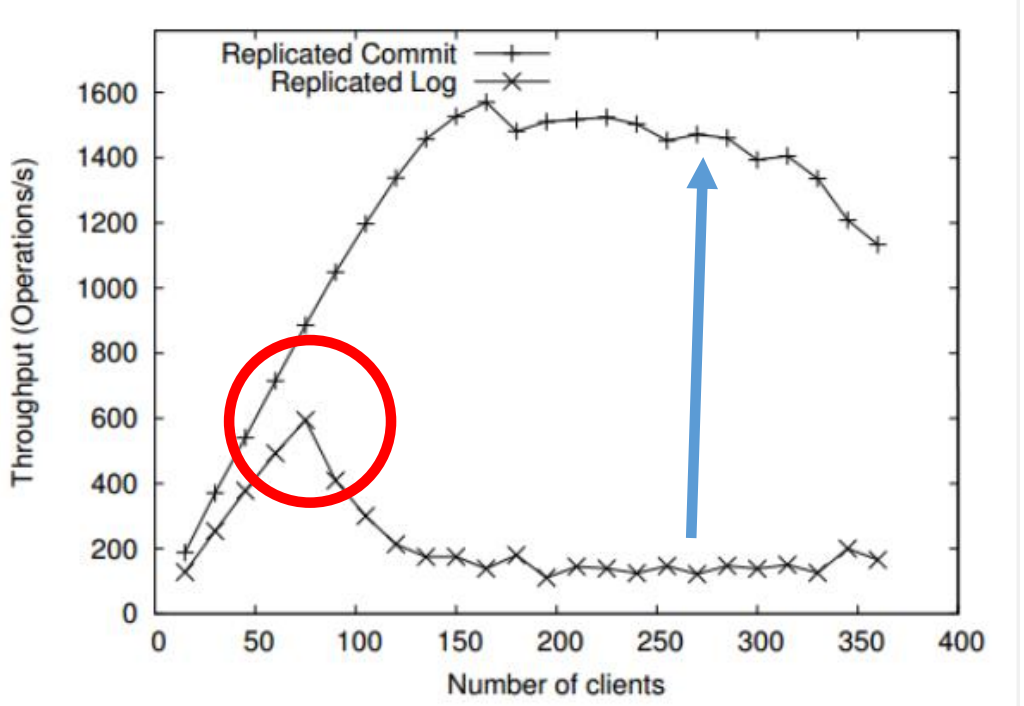
Scalability of Replicated Commit





Scalability of Replicated Commit

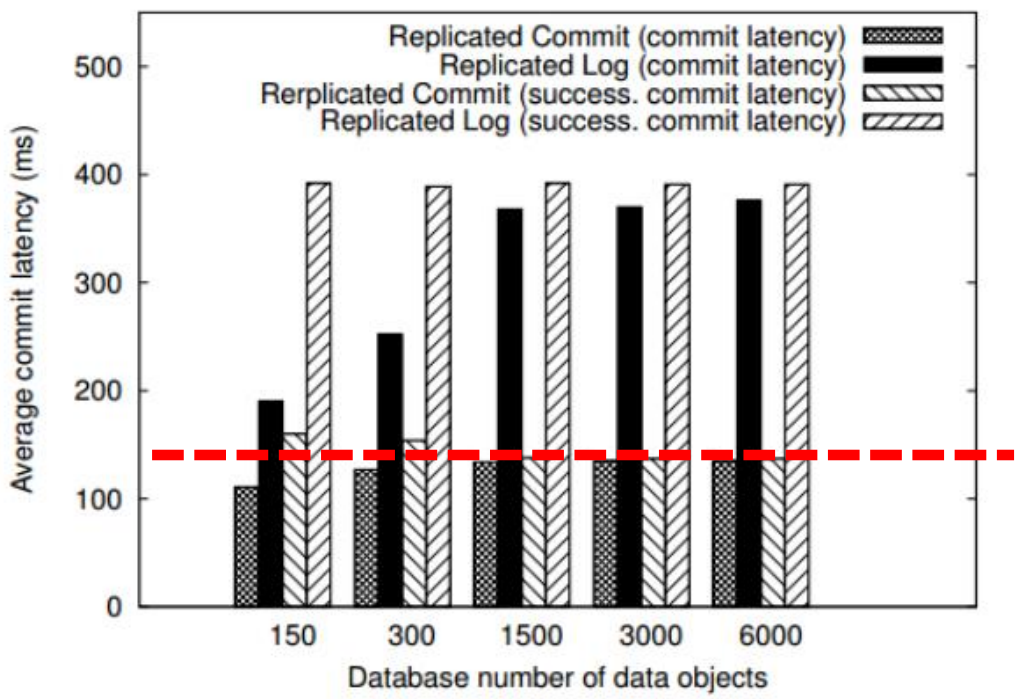


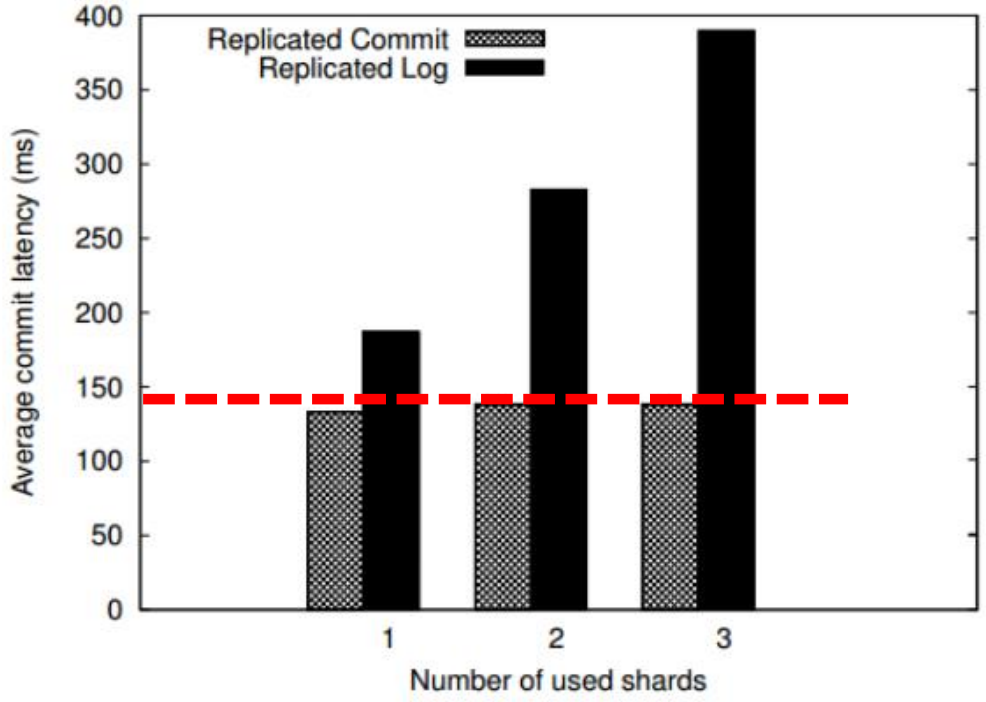


Replicated Commit has a very low read latency and comparable other latencies

Replicated Commit supports many more clients than Replicated Log

Changing Database Properties



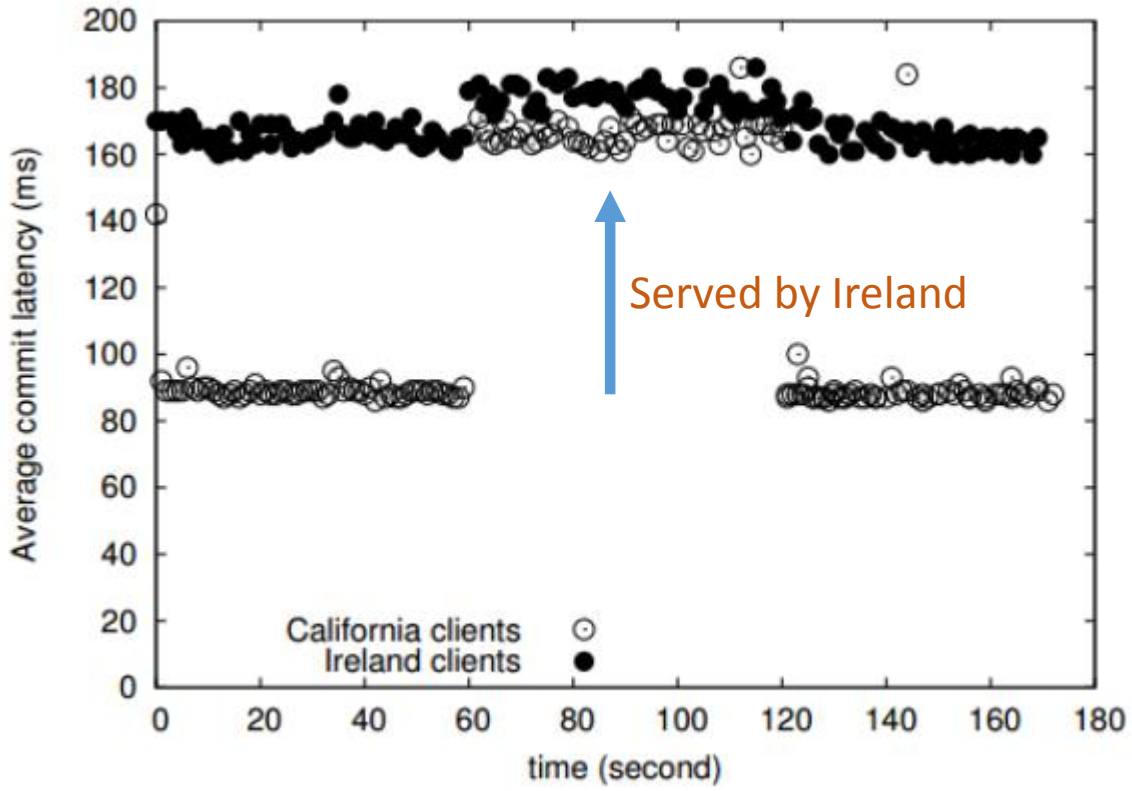


Increasing number of data objects does not affect Replicated Commit latency

Replicated Commit latency is agnostic of number of shards used

Fault Tolerance

- Sudden fault at one of the datacenters
- All clients are immediately served by another data center
- Latency increases in proportion
- Total latency for all clients gets affected possibly because of load



Comments

- Comparison between SQL and NoSQL is missing
- Effect of individual shards failing
- What is the tradeoff between Replicated Logs and Replicated Commit?
  - What are we losing if we adopt Replicated Commit?
  - Why does everyone not use Replicated Commit?
- Comparisons with other techniques discussed in the related work could have bolstered the paper even further.
- Intra-datacenter and inter-datacenter protocols are different (physical location knowledge helps the protocol – loss of abstraction)

Discussion

Conclusion

- A modified approach to achieving ACID properties in multi-datacenter settings is discussed
- Gains with respect to Replicated Logs are proportional to the latency between the servers
- More processing is done locally inside a data-center and then consensus is reached

Additional Material

Fault Tolerant Logs