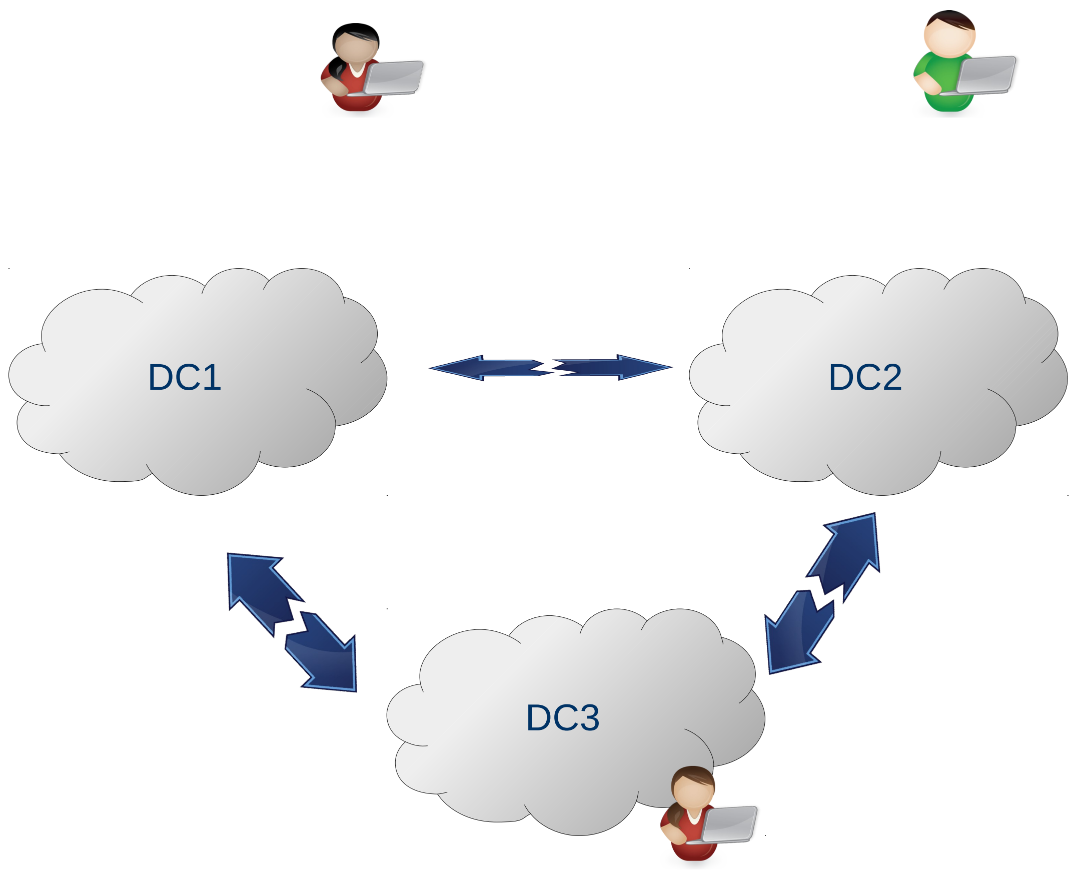
Causal consistency in Geo-replicated Systems

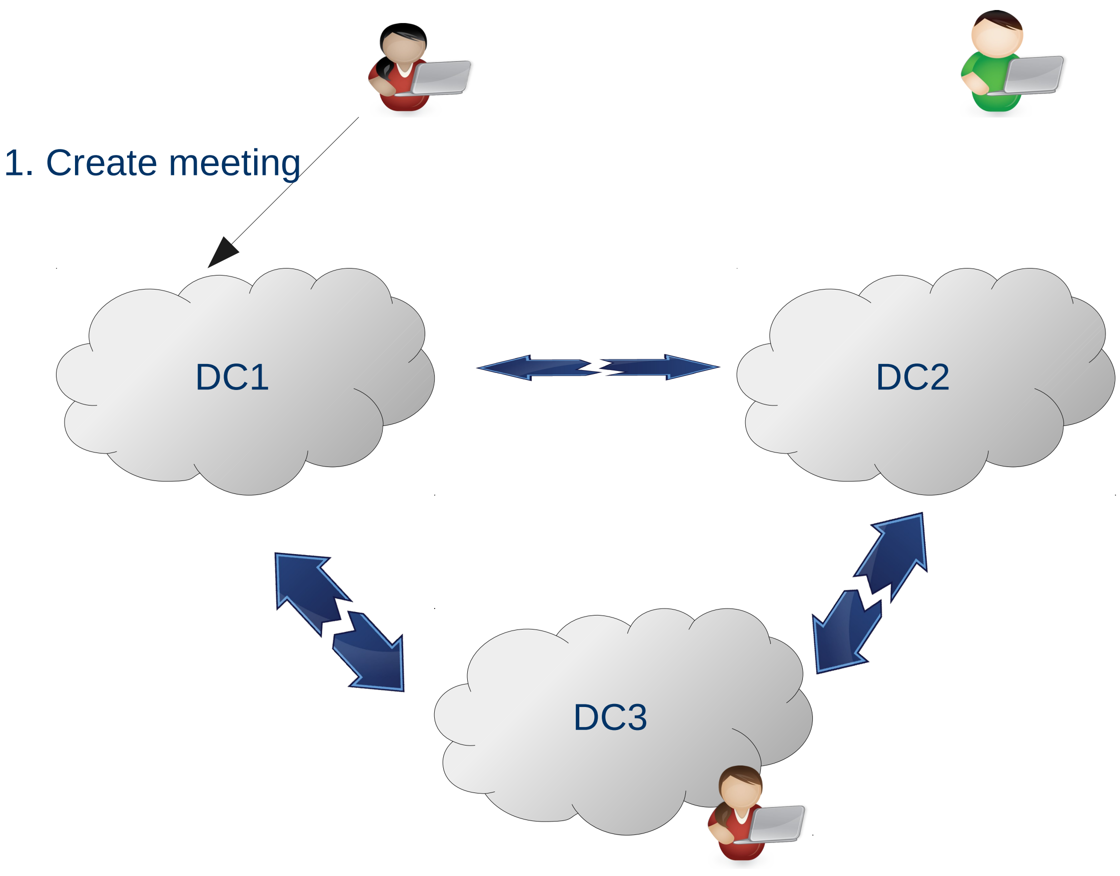
Deepthi Akkoorath

AG SoftTech

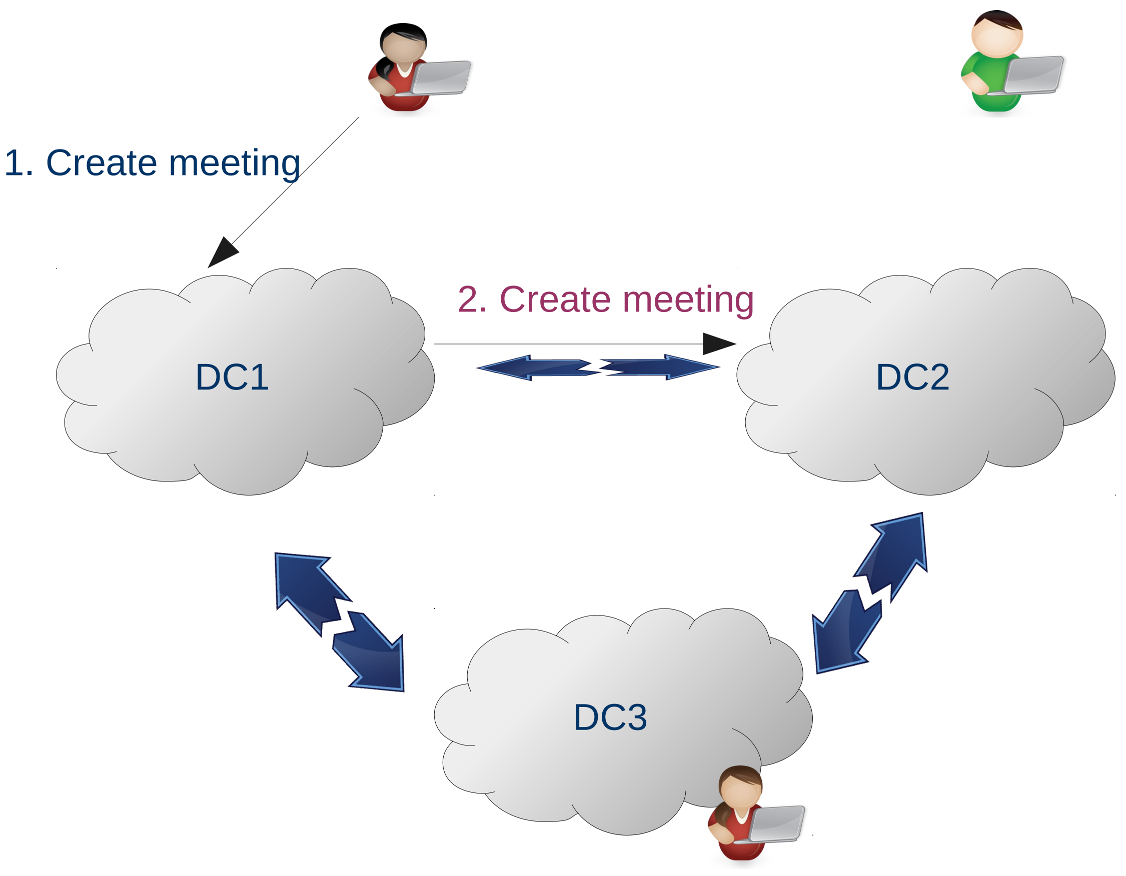
Geo-Replicated Distributed Application



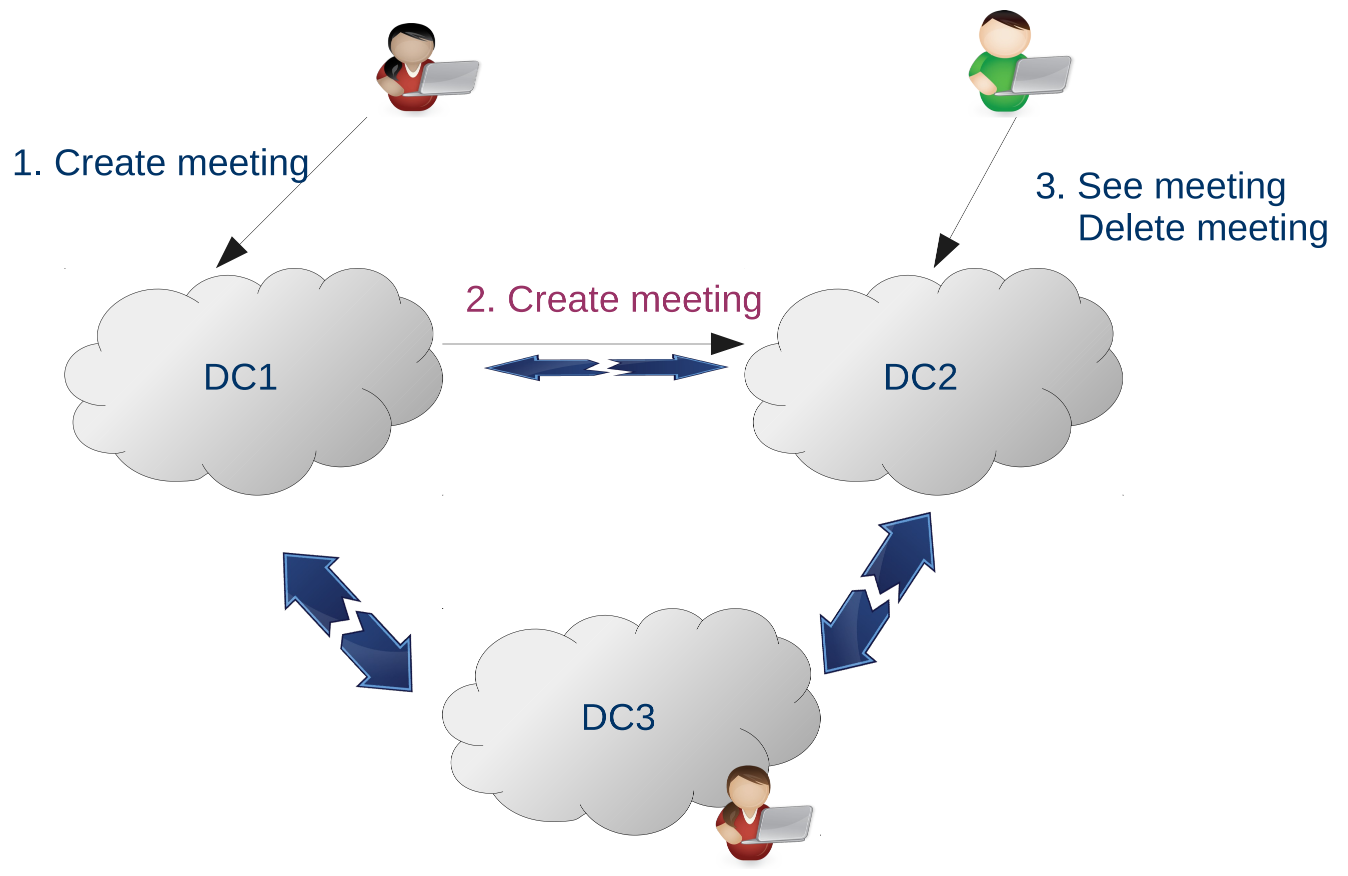
Geo-Replicated Distributed Application



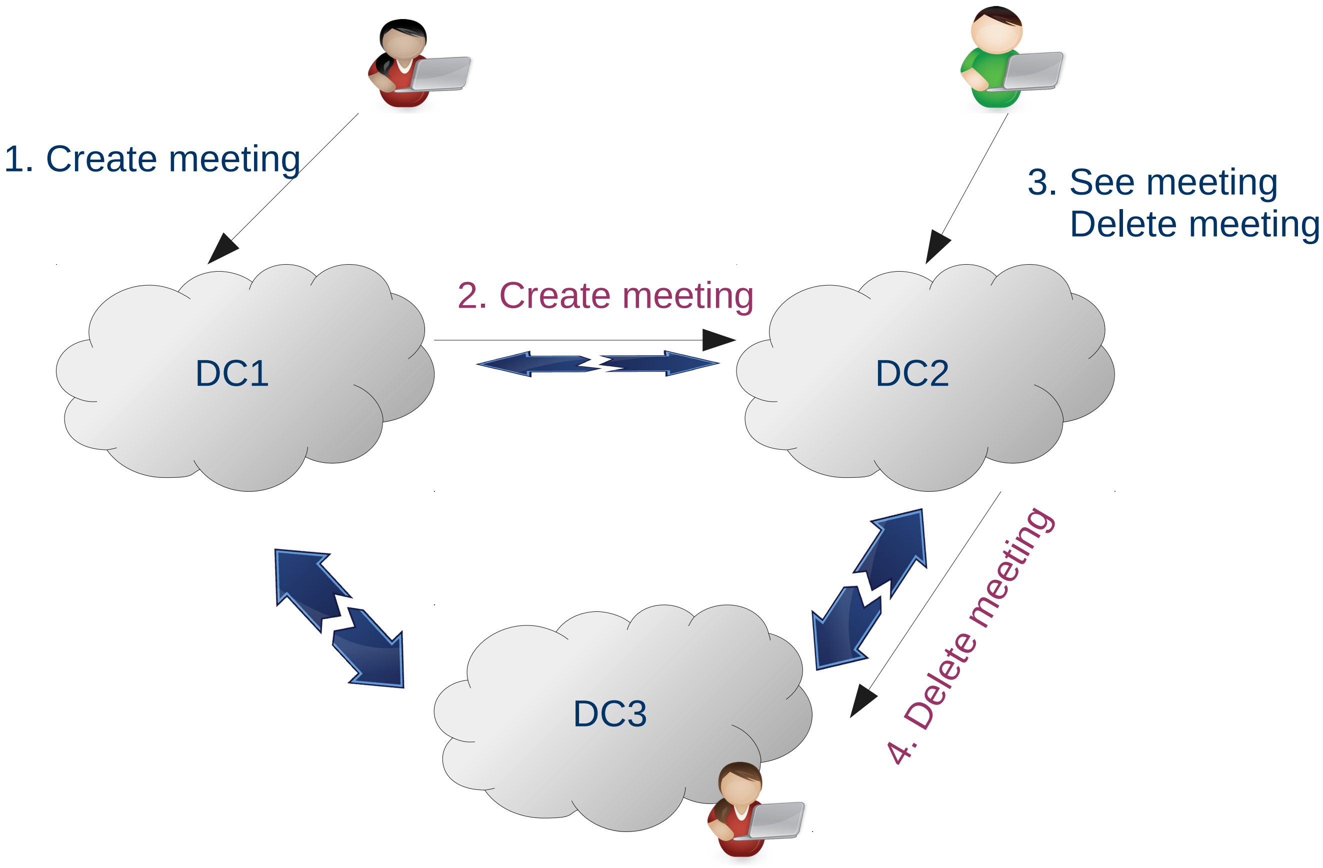
Geo-Replicated Distributed Application



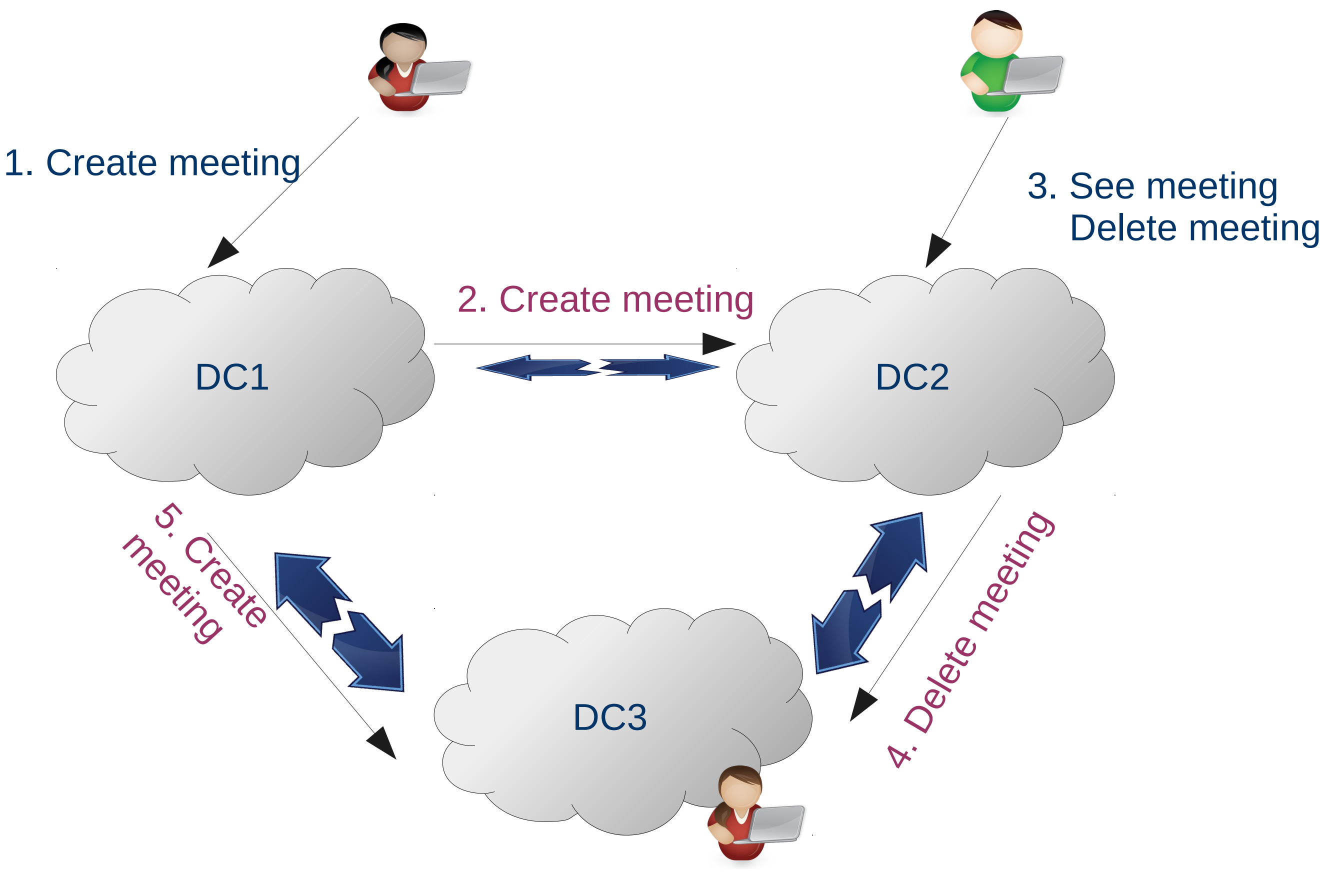
Geo-Replicated Distributed Application



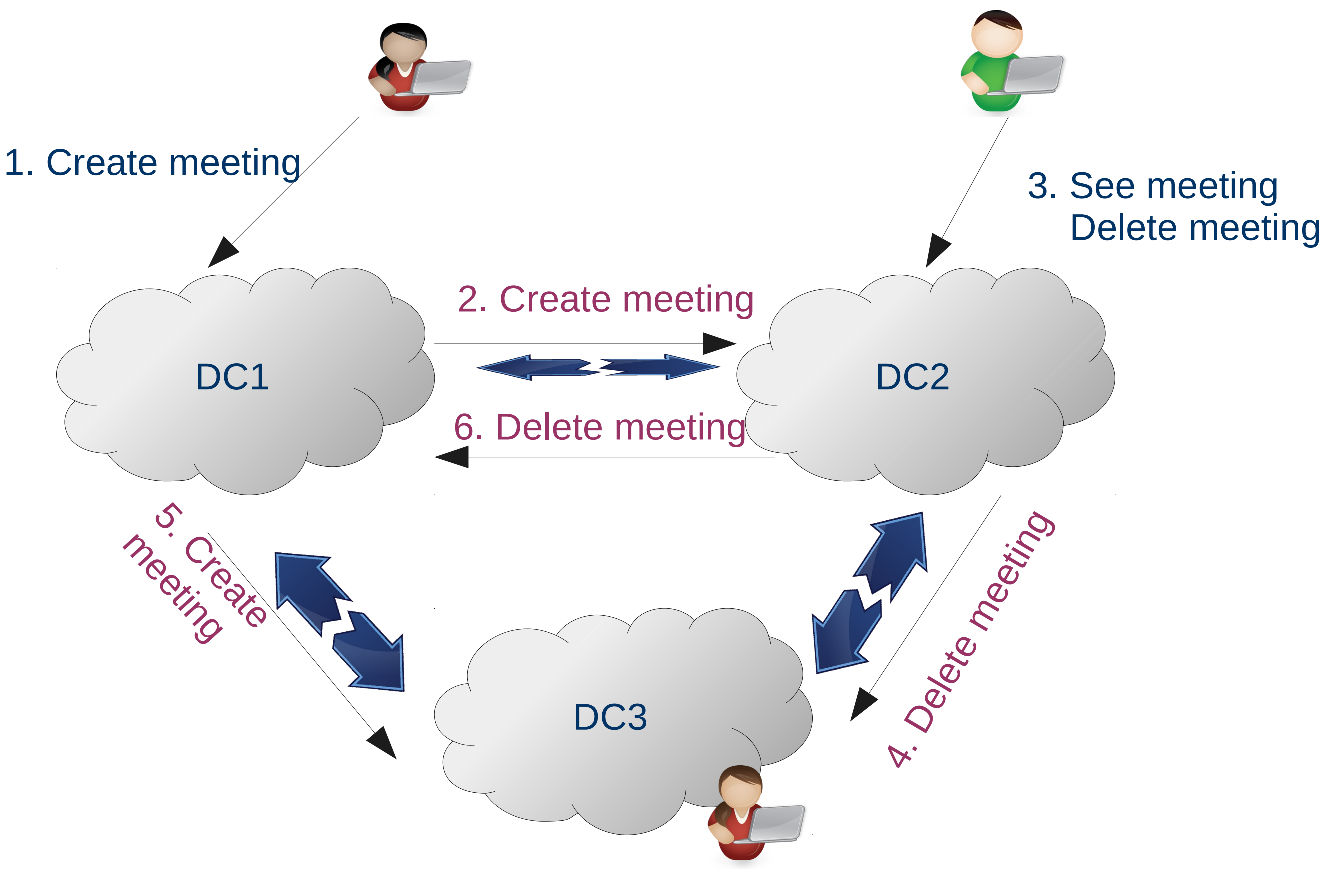
Geo-Replicated Distributed Application



Geo-Replicated Distributed Application



Geo-Replicated Distributed Application



Lamport's Timestamps

- Total order of events satisfying Happened-before relation
- Each process has a Logical clock
- A process increments its clock for each event
- Sends clock with each message it sends
- On receiving a message
  - Sets clock = max(own clock, received clock)

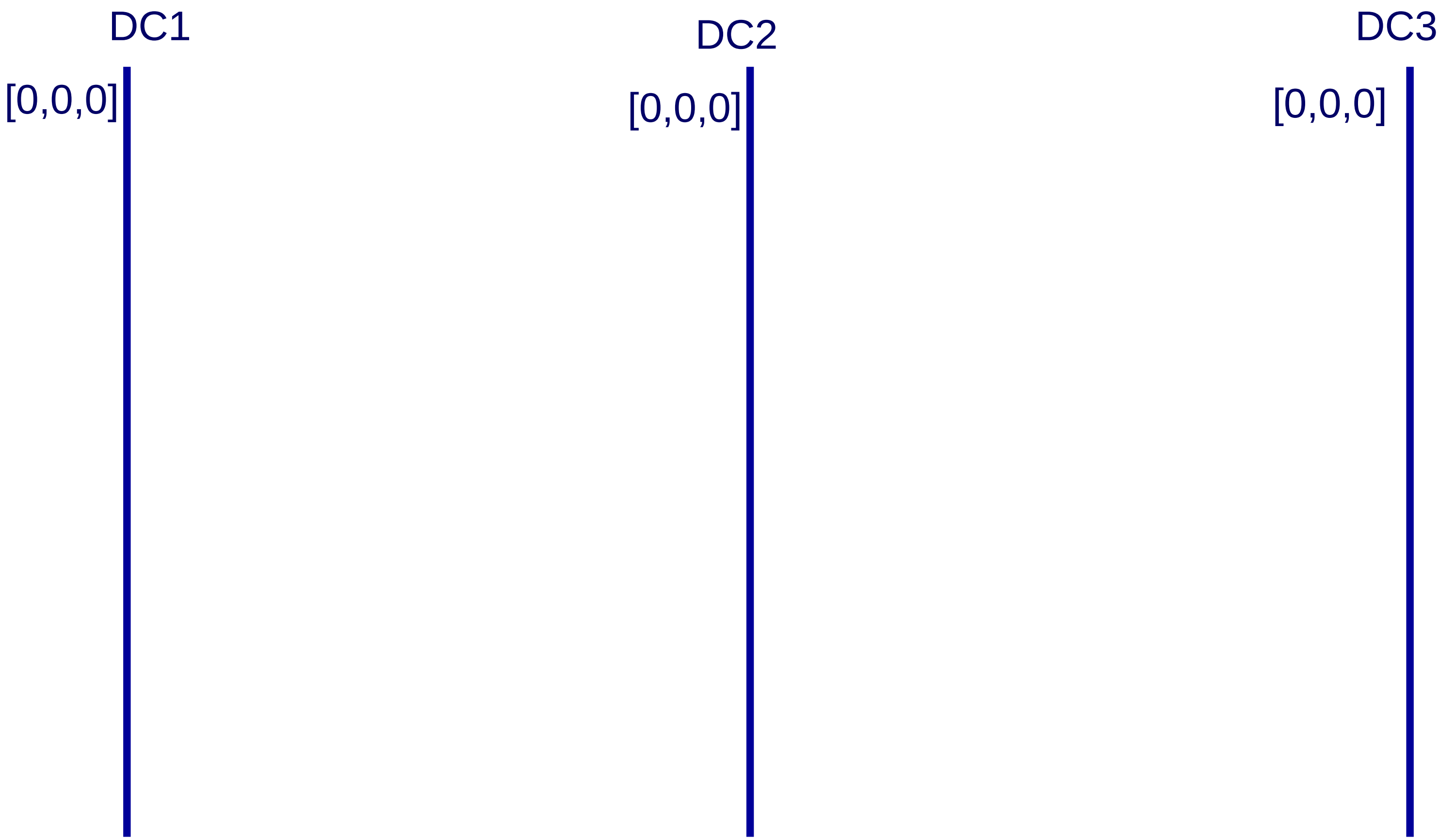
Lamport's Timestamps

- Total order of events satisfying Happened-before relation
- Each process has a Logical clock
- A process increments its clock for each event
- Sends clock with each message it sends
- On receiving a message
  - Sets clock = max(own clock, received clock)

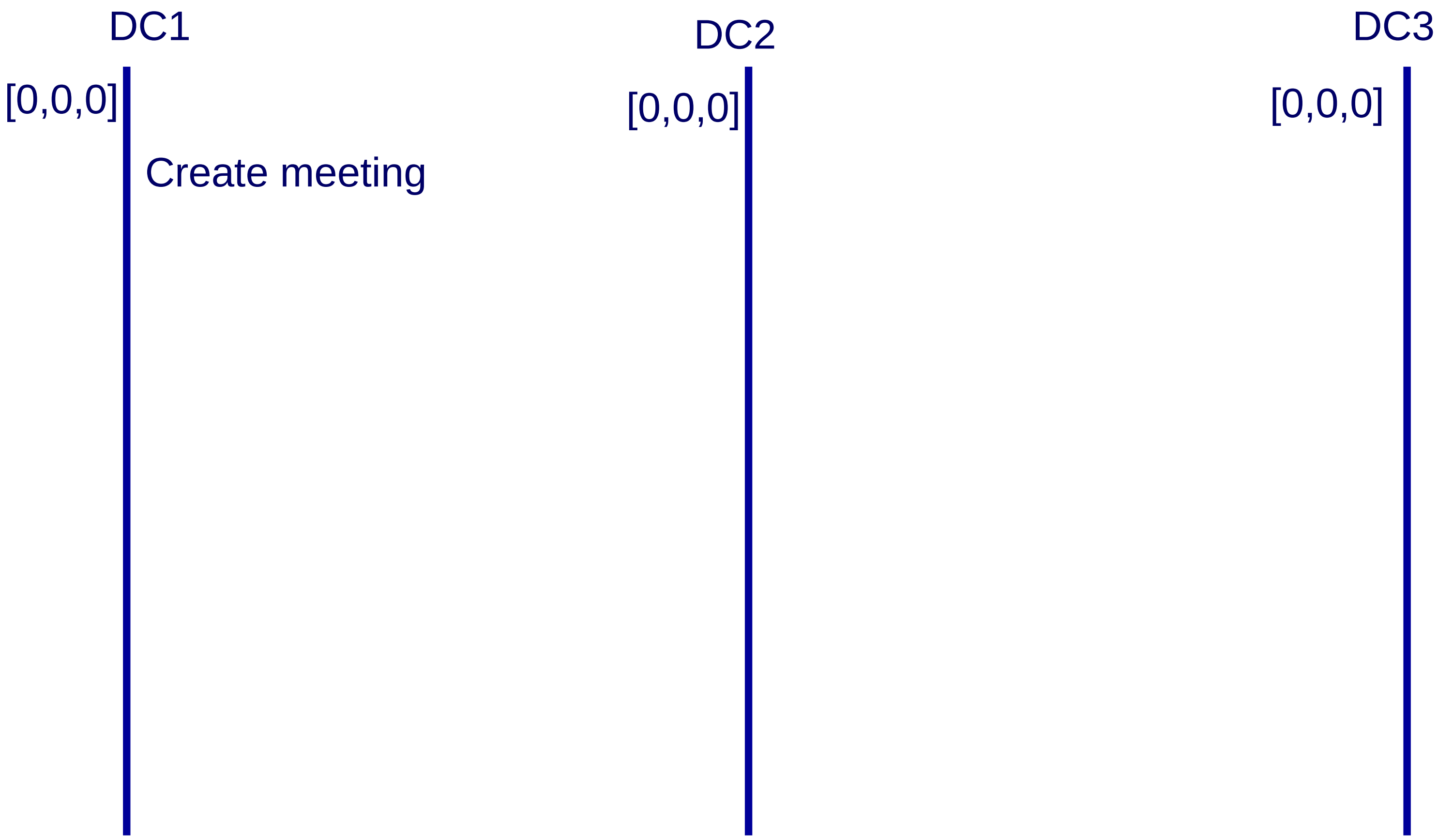
Vectorclocks

- Similar to Lamport's timestamp
- Partial order and detect causality violations
- A system on N process
  - Vectorclock = array of N logical clocks
  - Each process has a vectorclock
  - Increment its own logical clock for each event
  - On receiving a message
    - Set each entry in vc to be max(local entry, corresponding entry in received vc)

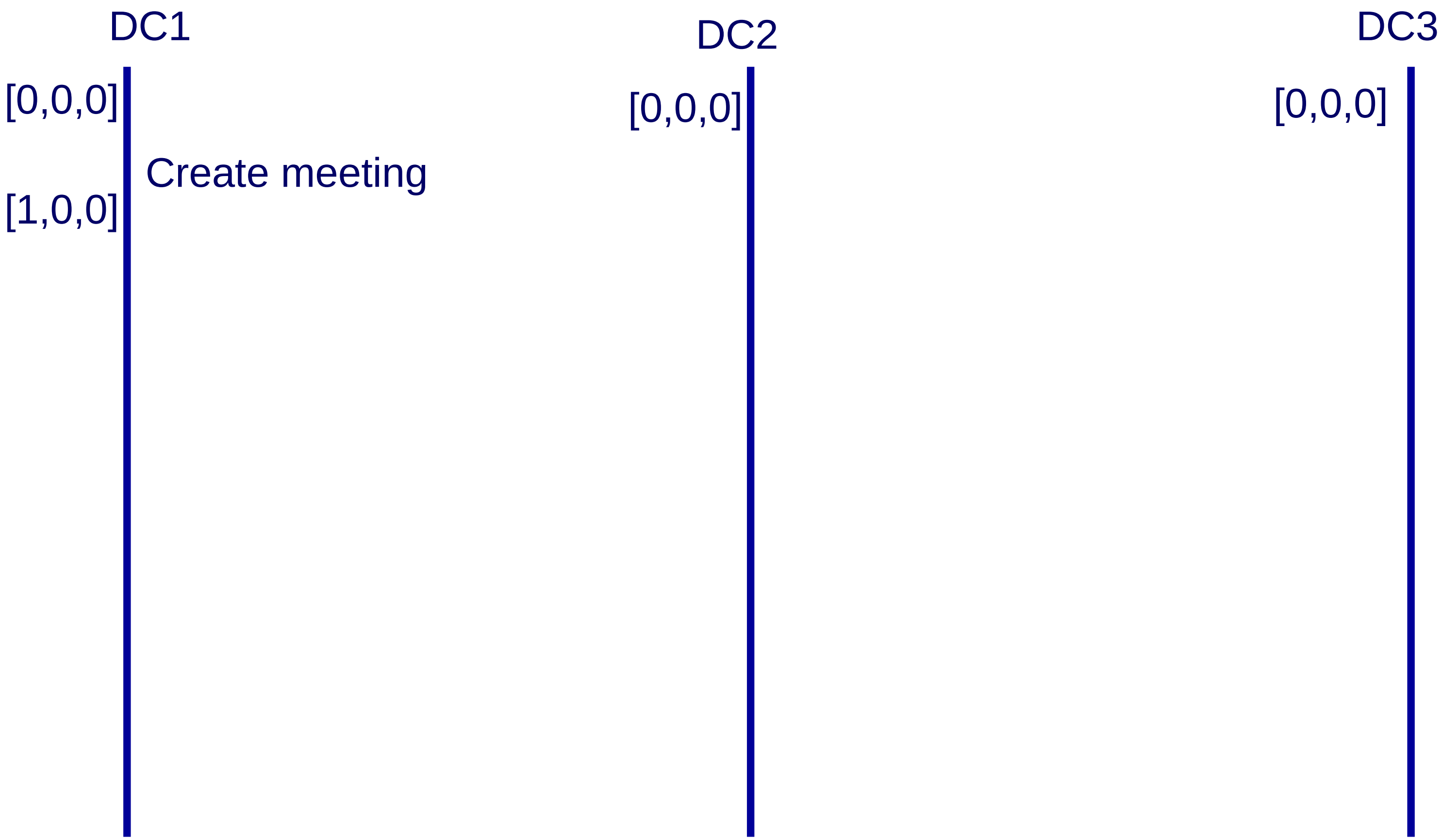
Detect Causality violation using Vectorclocks



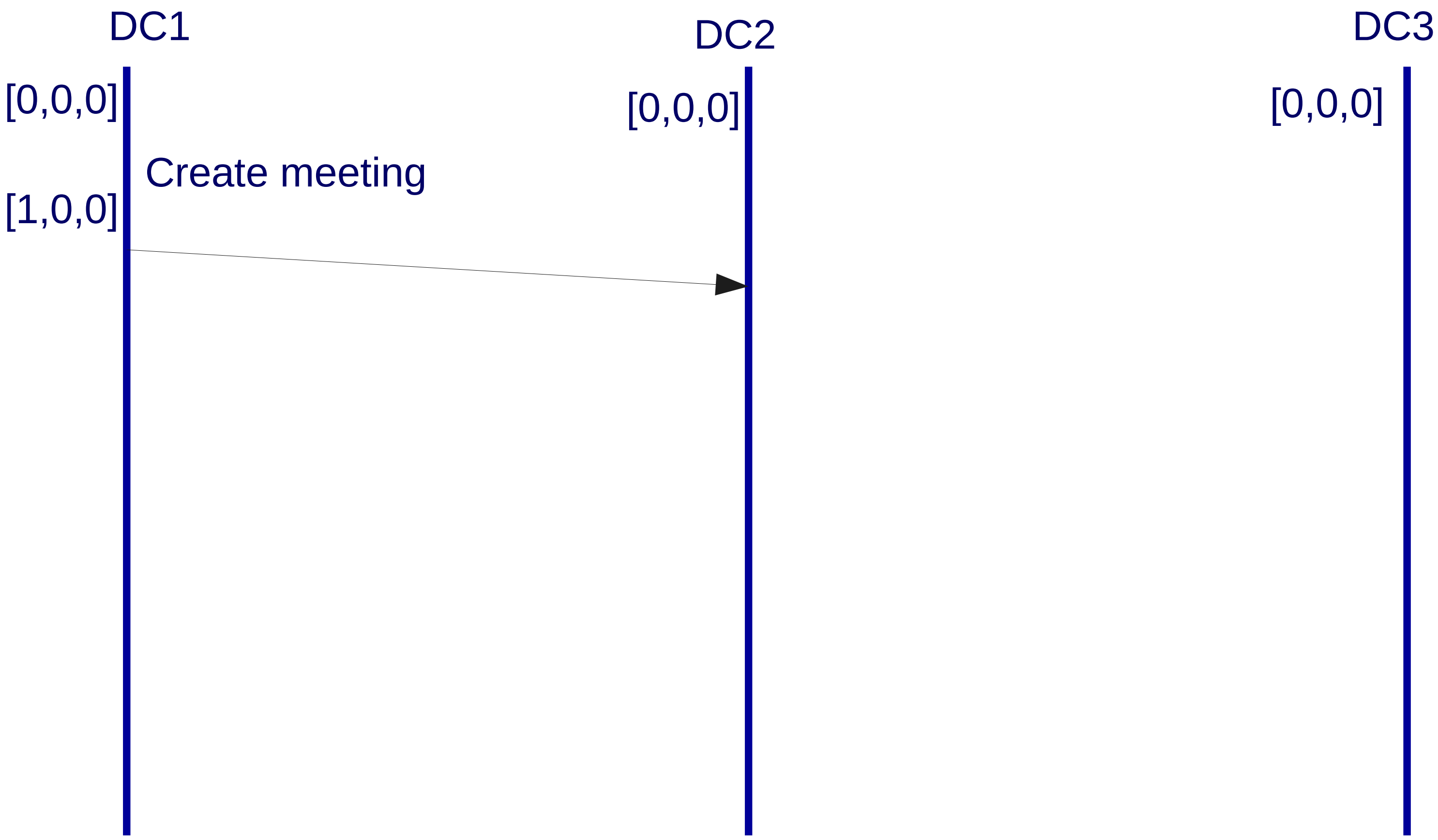
Detect Causality violation using Vectorclocks



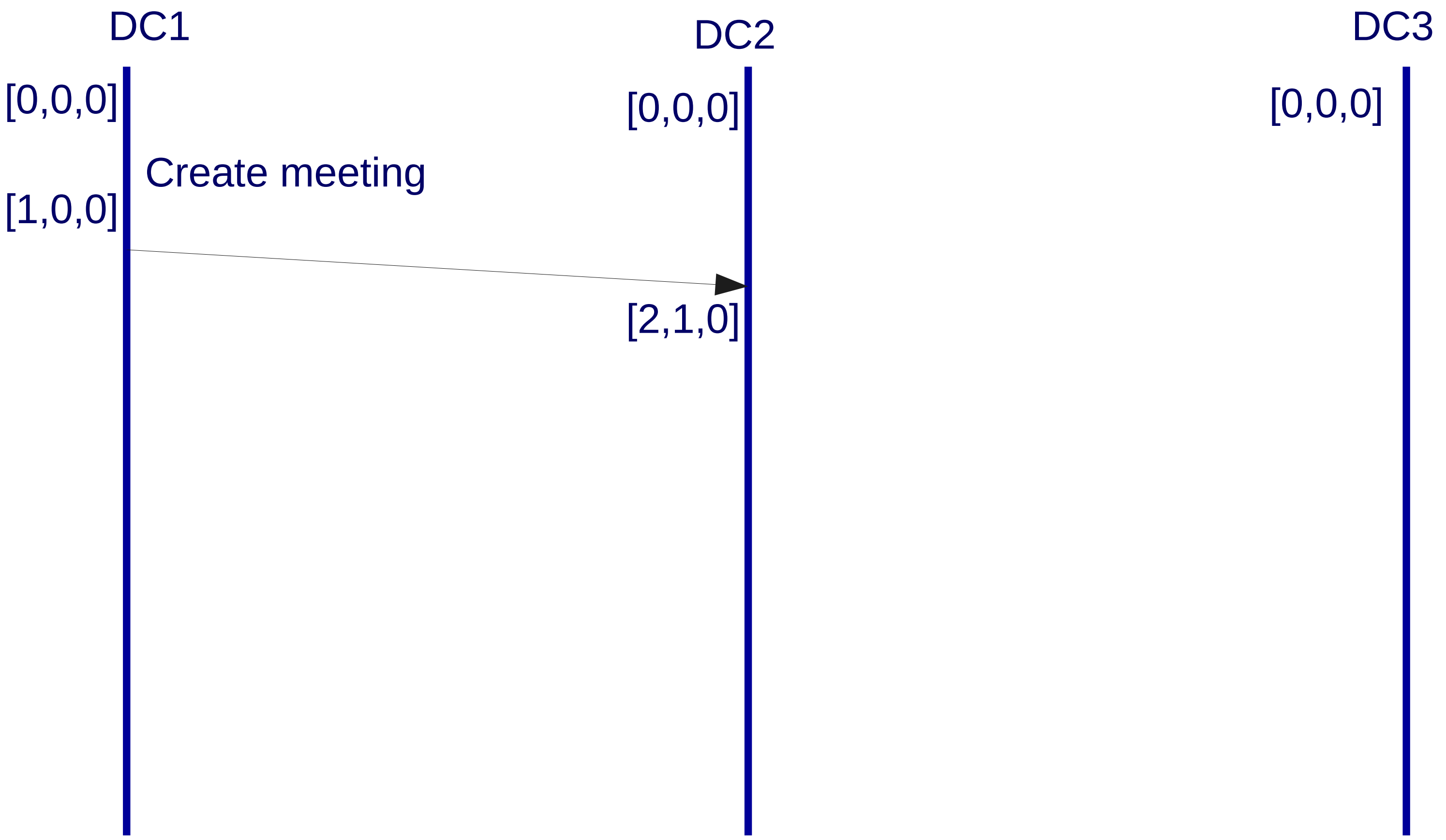
Detect Causality violation using Vectorclocks



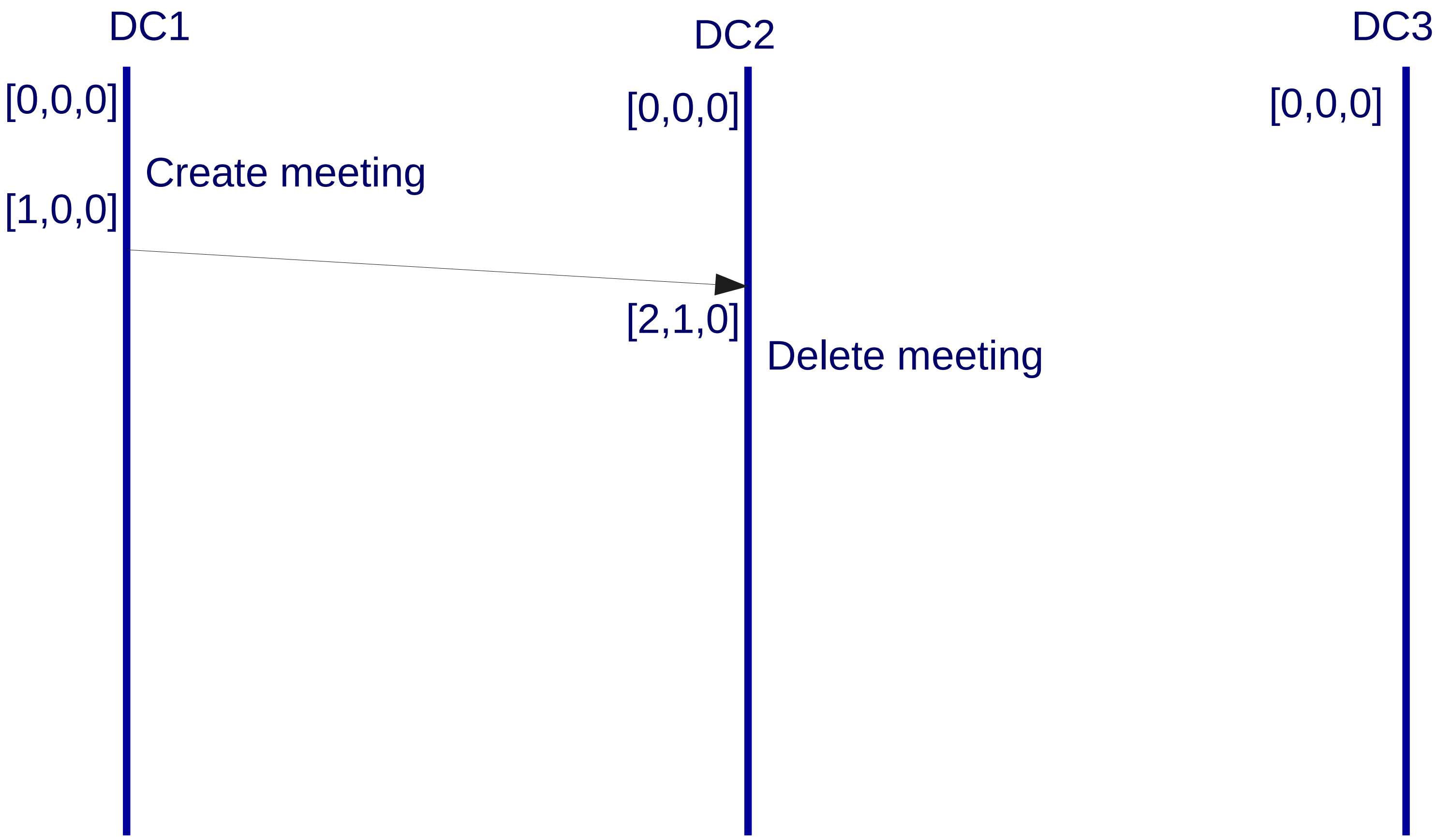
Detect Causality violation using Vectorclocks



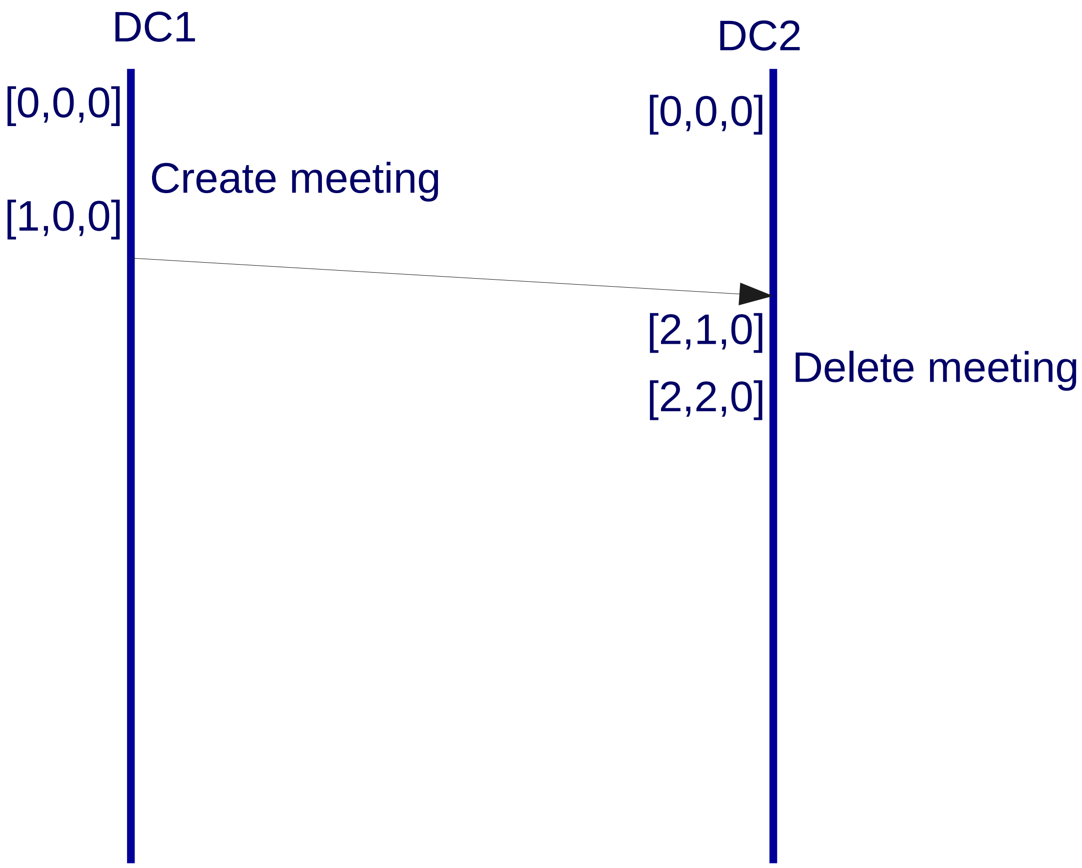
Detect Causality violation using Vectorclocks



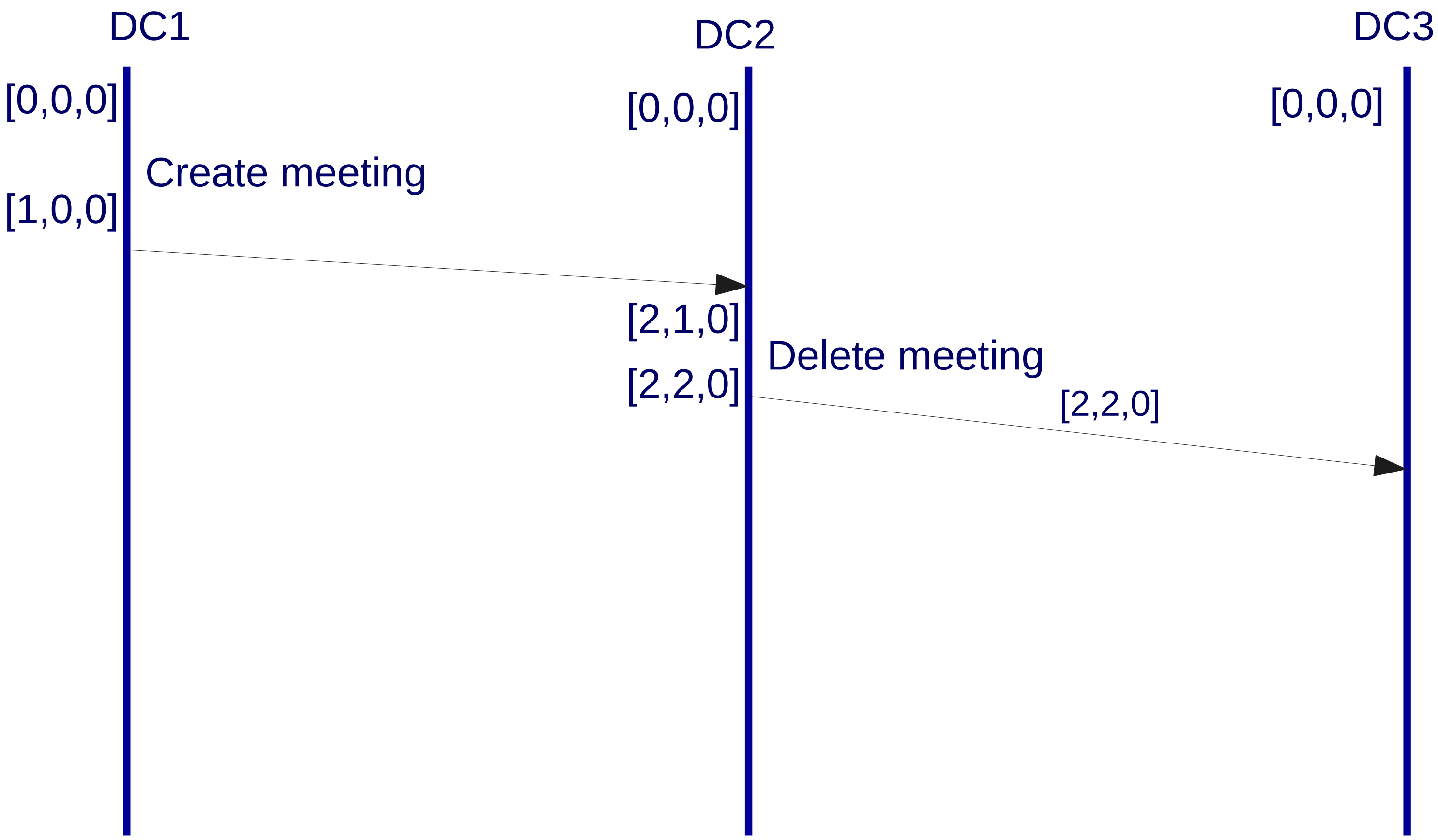
Detect Causality violation using Vectorclocks



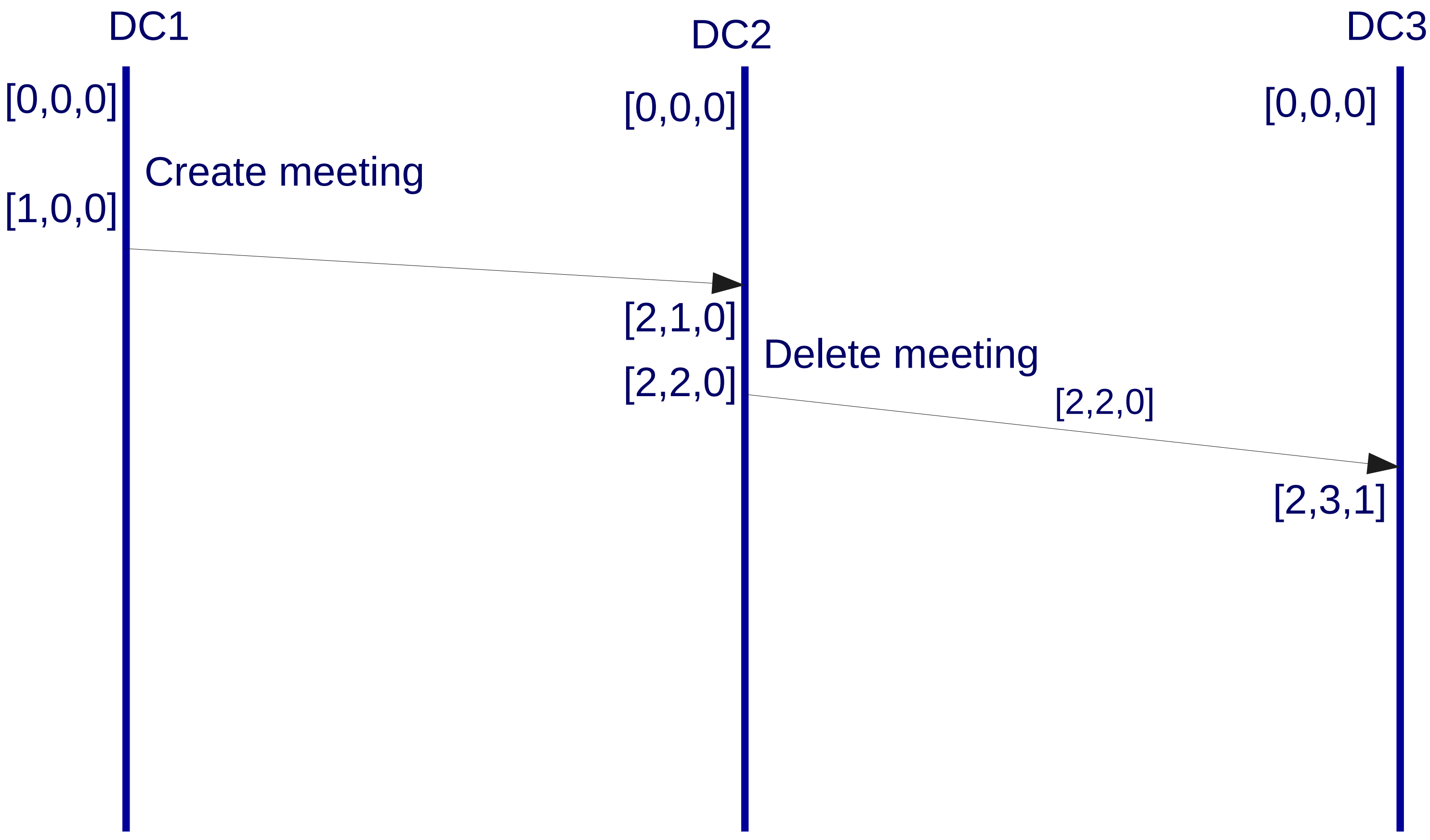
Detect Causality violation using Vectorclocks



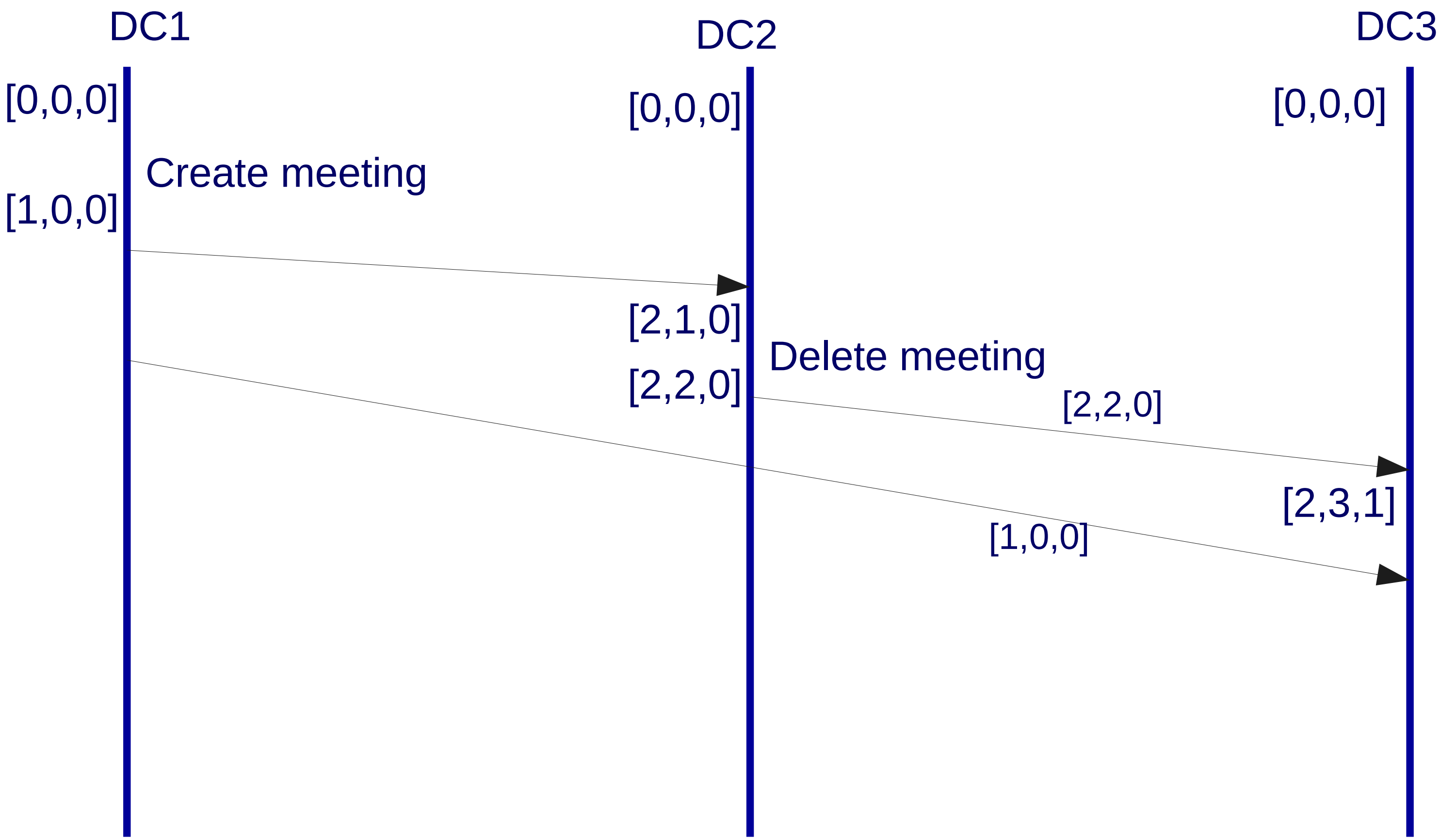
Detect Causality violation using Vectorclocks



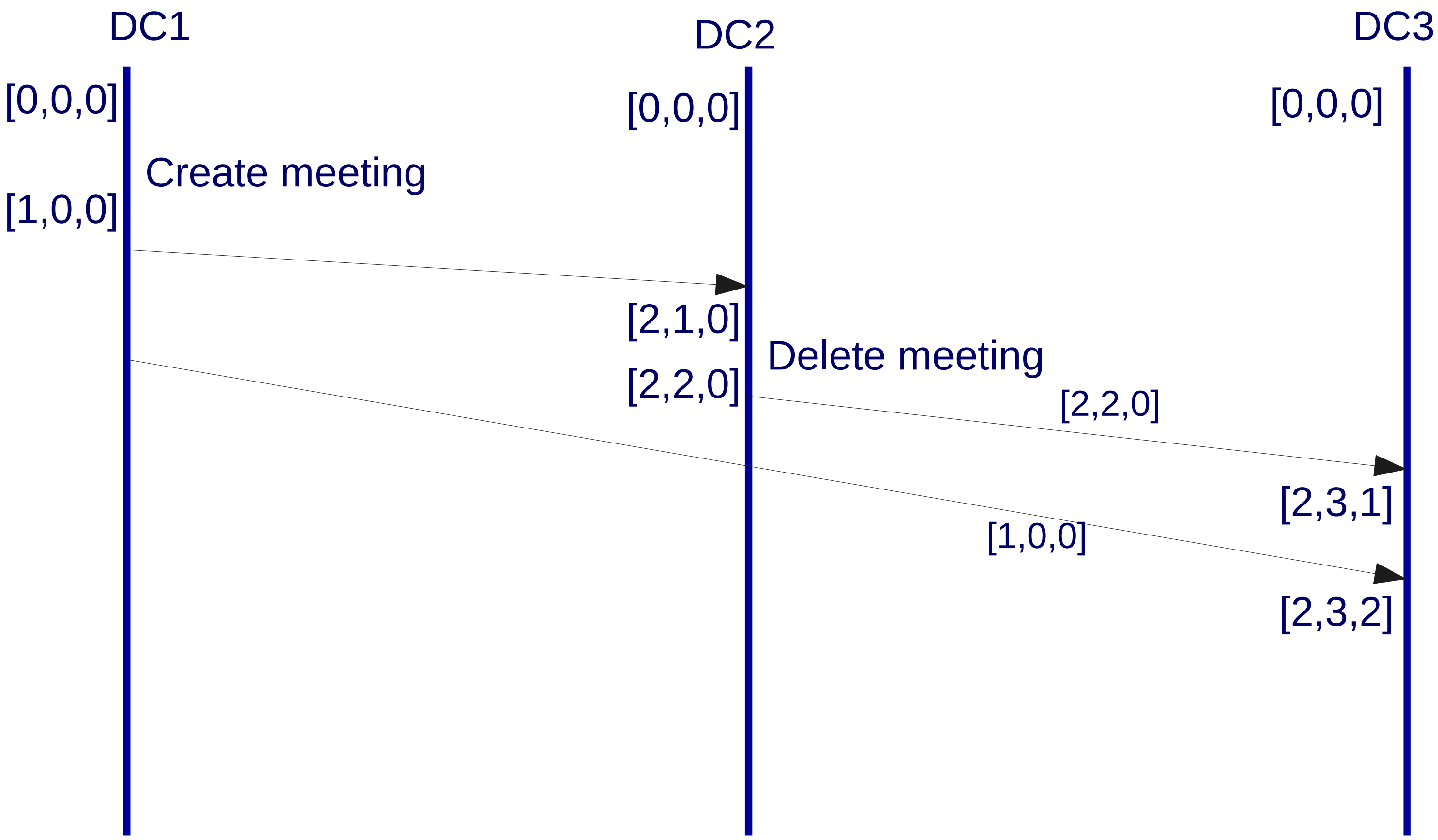
Detect Causality violation using Vectorclocks



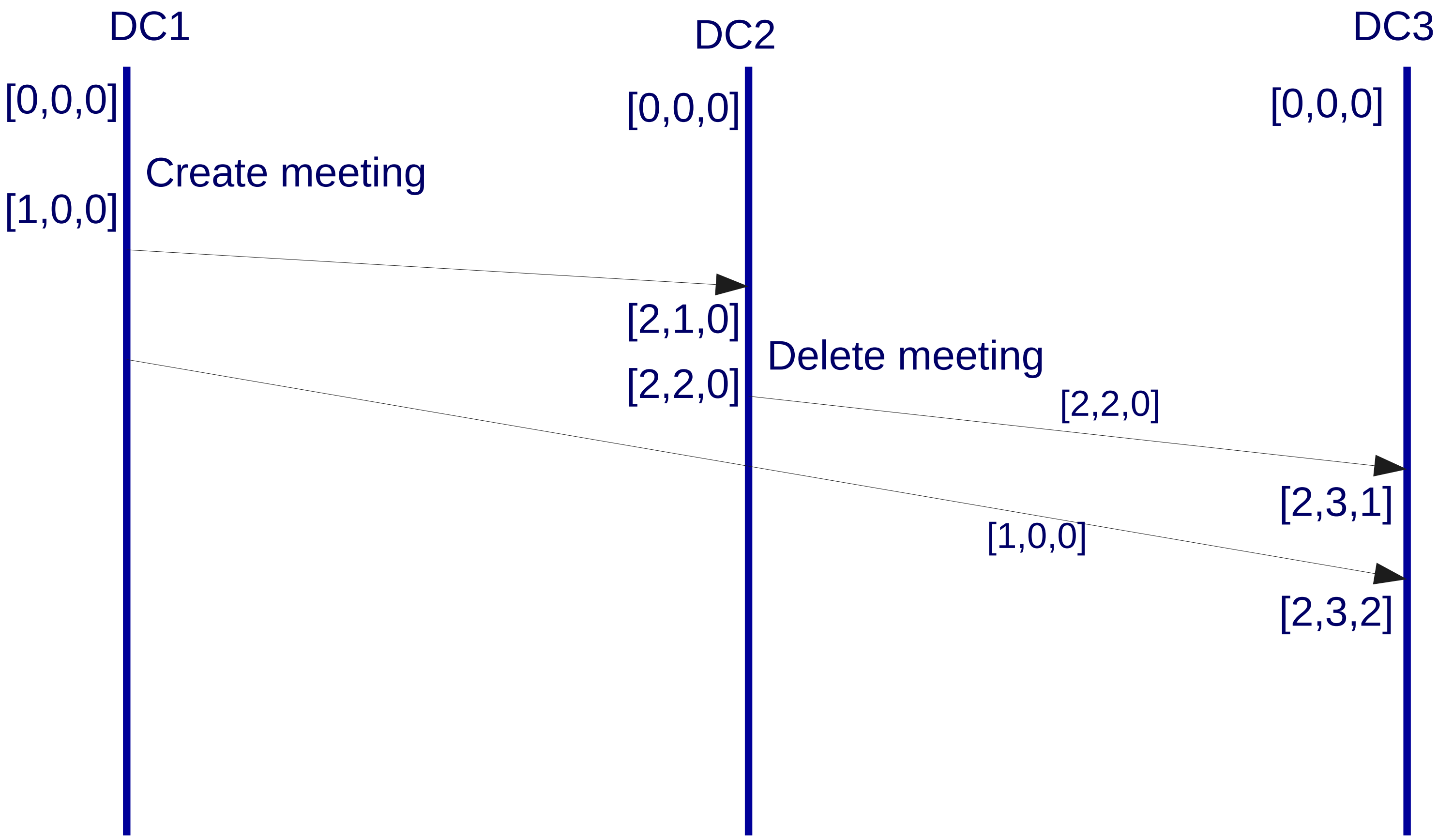
Detect Causality violation using Vectorclocks



Detect Causality violation using Vectorclocks



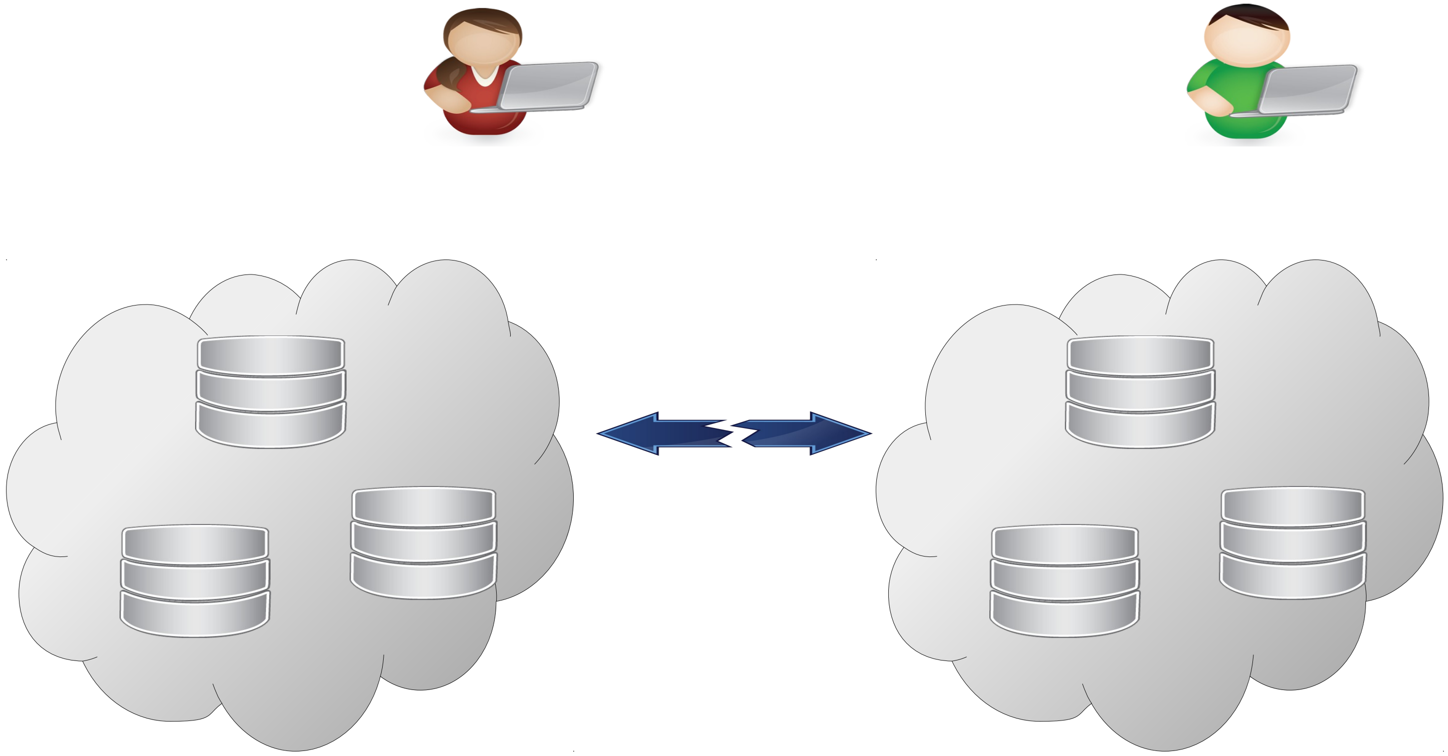
Detect Causality violation using Vectorclocks



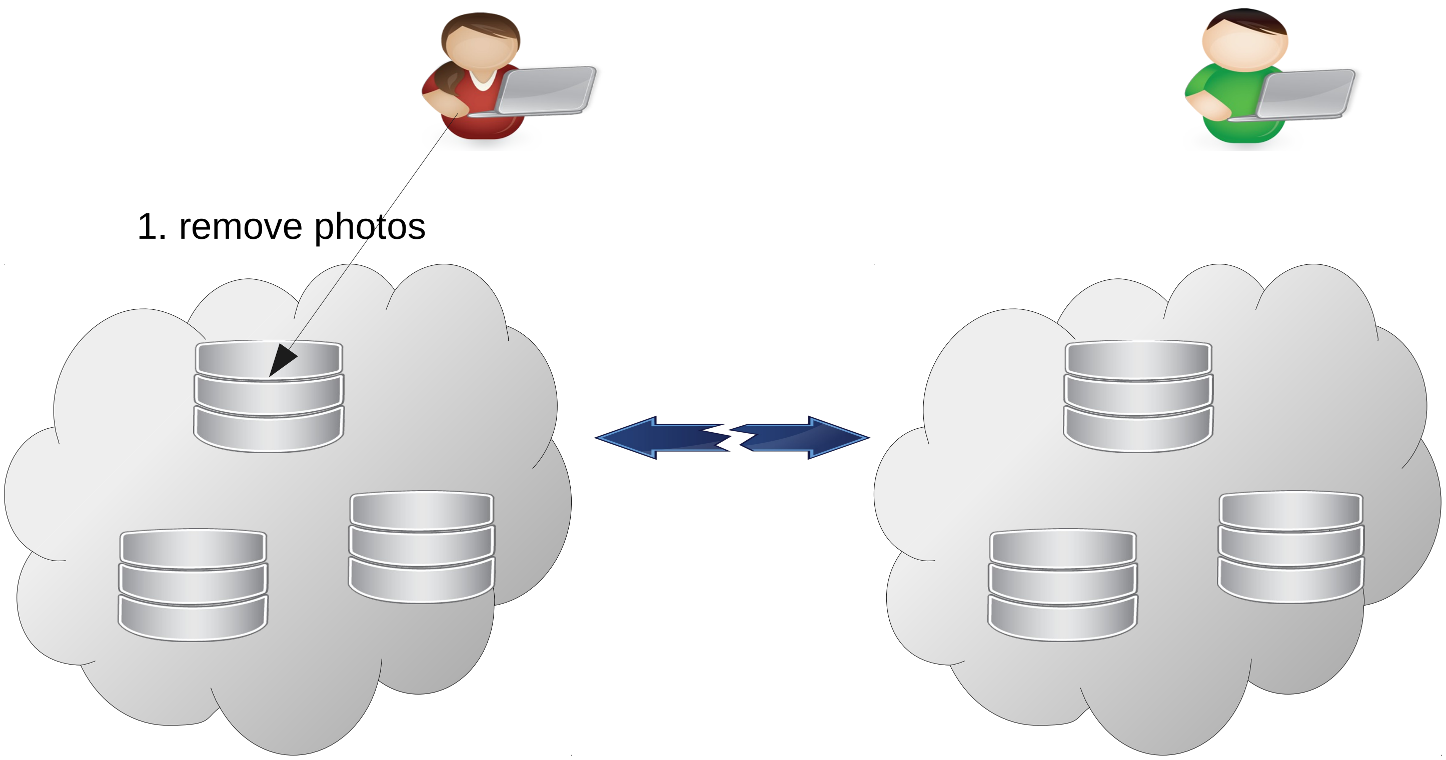
Version Vectors

- Similar to vector clocks
- Partial order among replicas of an object
- Several mechanisms to keep size of version vector small
  - Bounded Version Vectors
  - Dotted Version Vectors
- Causality across objects cannot be tracked

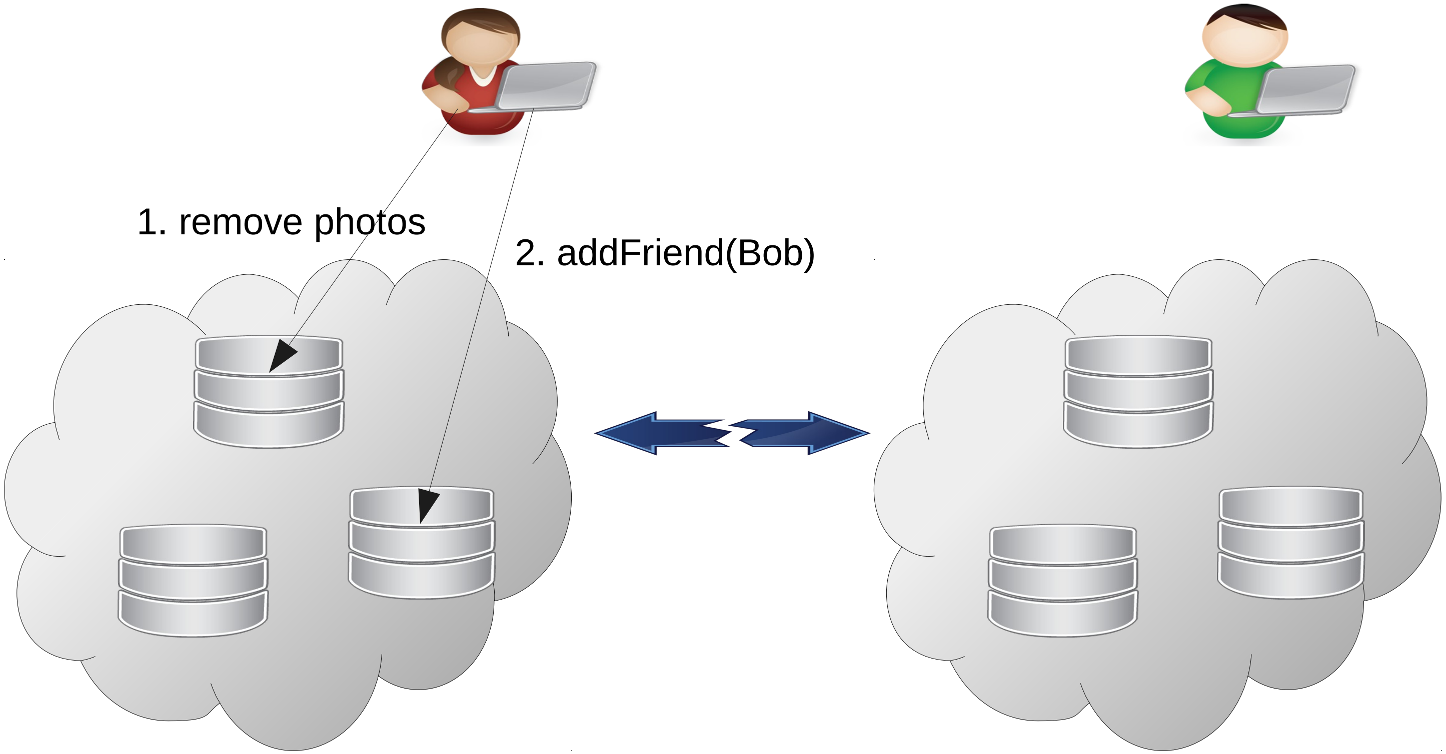
Partitioned and Geo-Replicated Distributed System



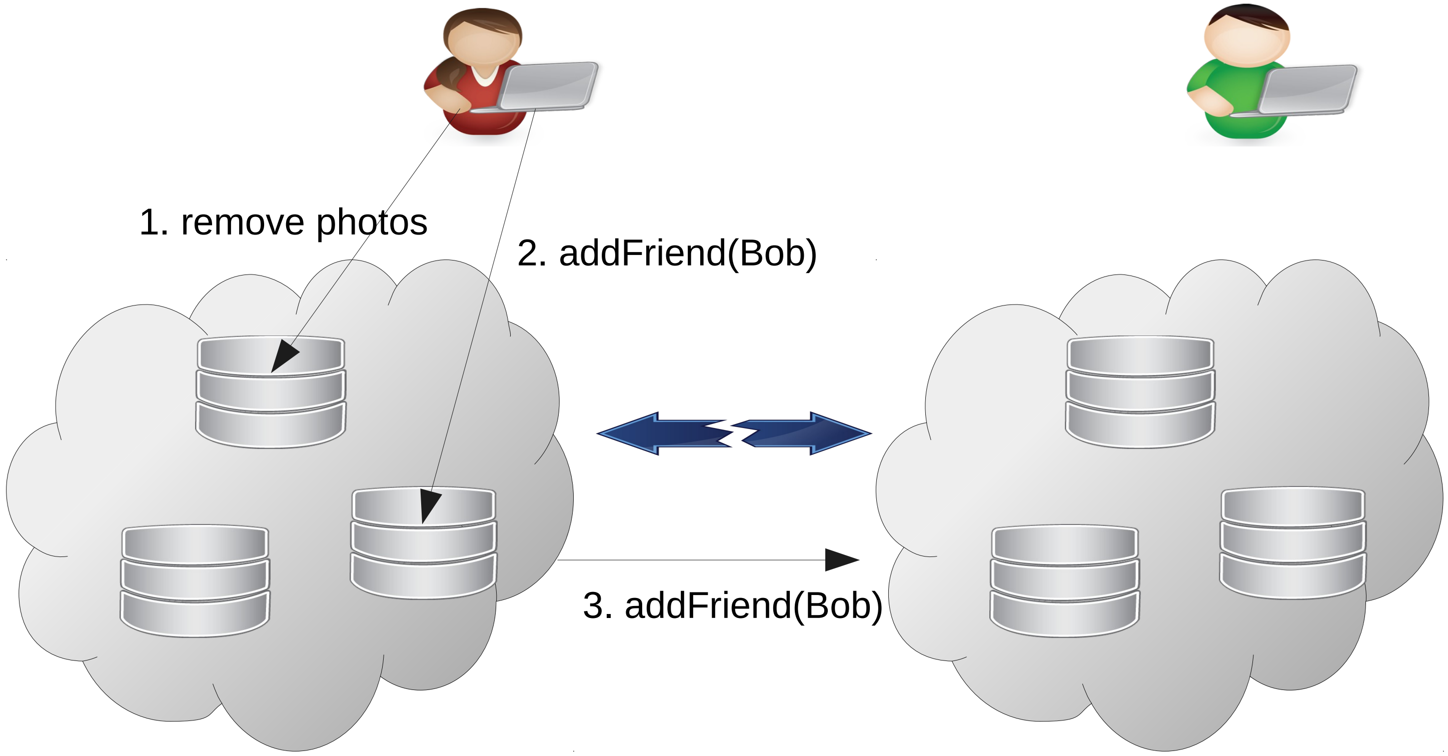
Partitioned and Geo-Replicated Distributed System



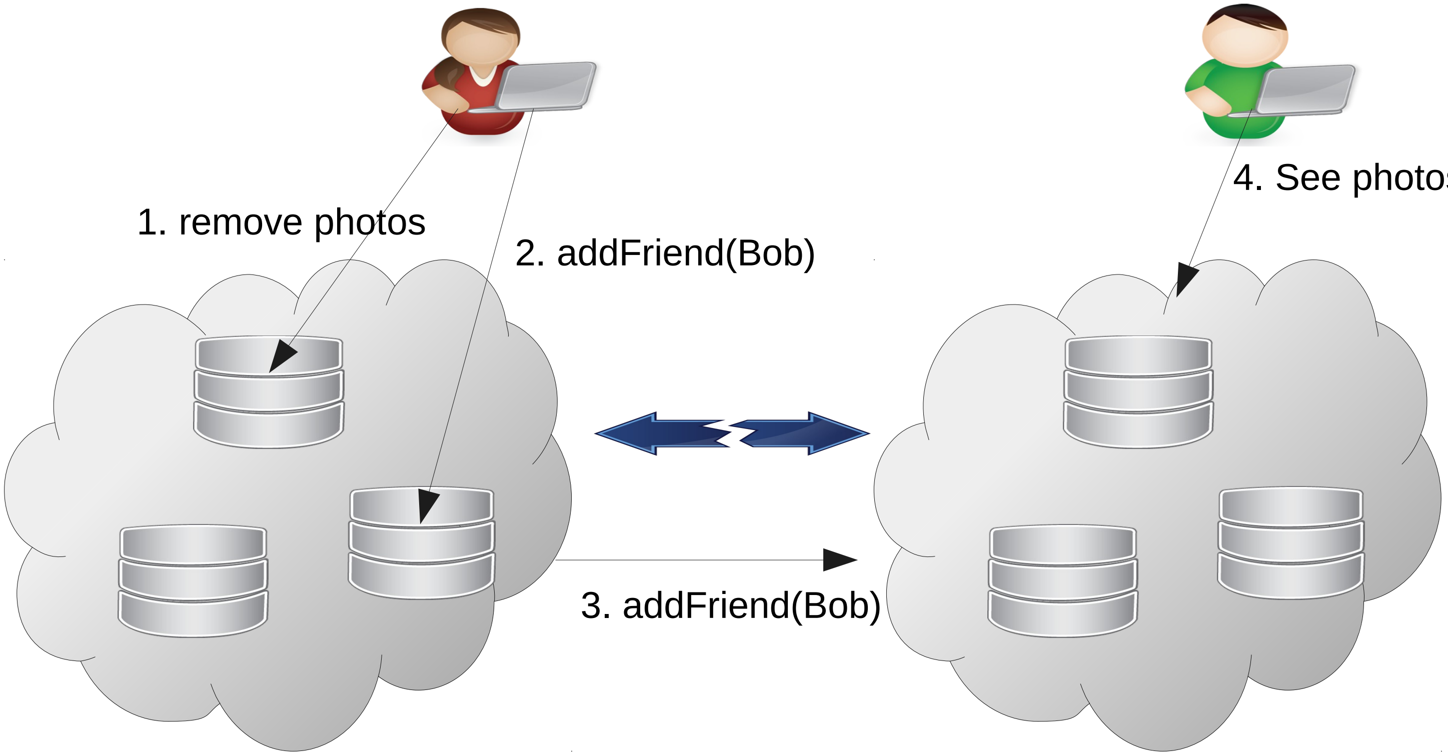
Partitioned and Geo-Replicated Distributed System



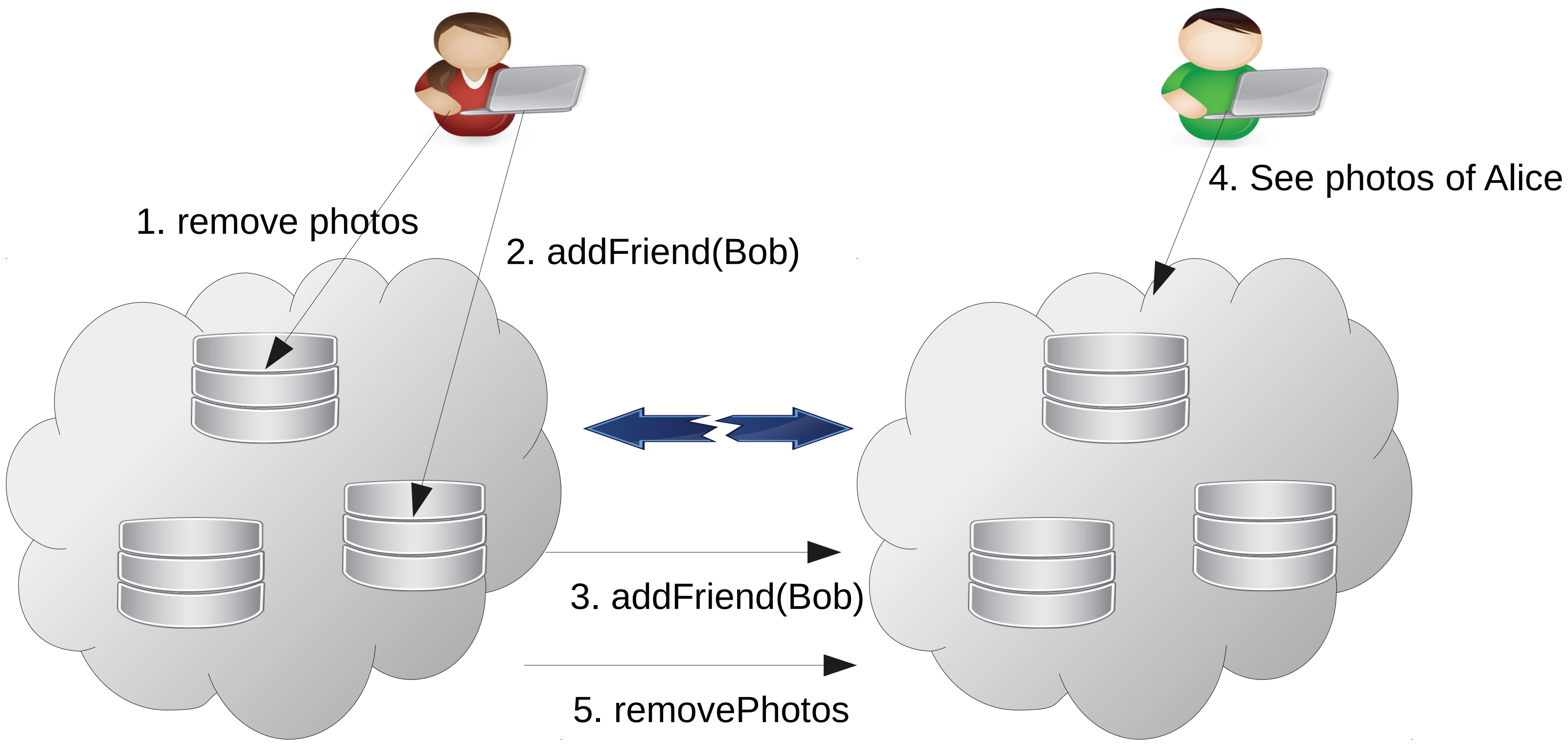
Partitioned and Geo-Replicated Distributed System



Partitioned and Geo-Replicated Distributed System



Partitioned and Geo-Replicated Distributed System



Orbe: Causal Consistency with Dependency Matrix

Clock	R1	R2	R3	R4
P1	0	2	0	0
P2	1	0	3	0
P3	0	0	0	1

Orbe: Causal Consistency with Dependency Matrix

- Dependency matrices to track causality
- Client updates its DM when ever it reads a new version

Clock	R1	R2	R3	R4
P1	0	2	0	0
P2	1	0	3	0
P3	0	0	0	1

- Client has seen first 2 updates at replica 2 of partition 1

Orbe: Causal Consistency with Dependency Matrix

- Each Partition has its own version vector - VV

VV	R1	R2	R3	R4
P1/R1	1	2	1	0

- P1 at DC1 has
  - 1 local update
  - 2 updates from R2
  - 1 update from R3

Orbe: Causal Consistency with Dependency Matrix

- Client send put(k,v,DM) to partition P1 at DC1
- P1 at DC1
  - Increment its own VV[R1]
  - Ts = VV[R1]
  - New entry U<k, v, 2, DM, R1>
  - Replicate U to P1 at DC2 and DC3
- On receiving U< k, v, ts, DM, replicaid> at Pn
  - Check VV >= DM[n]
  - Check if causality is satisfied at other partitions
  - Update VV[replicaid] = ts

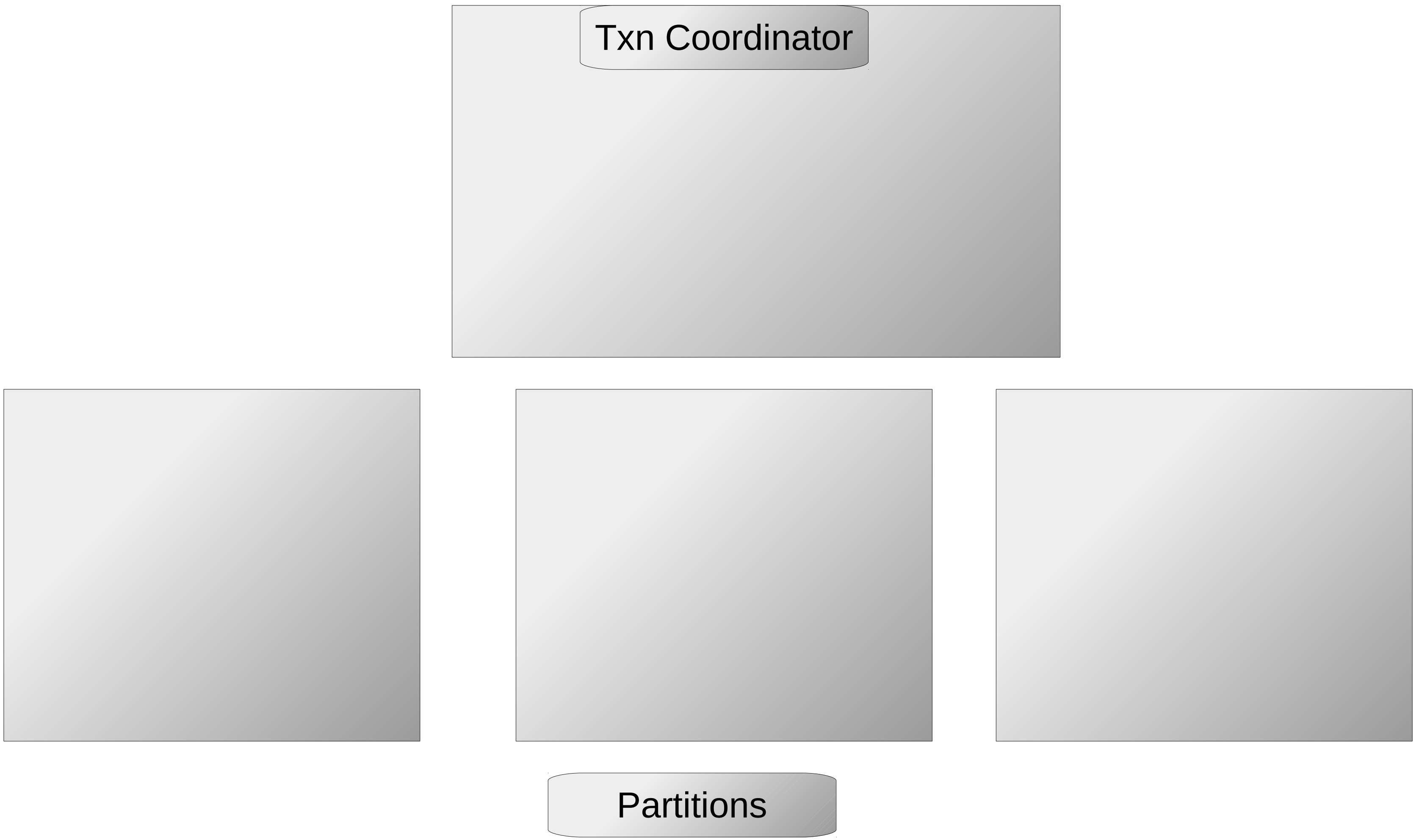
Total order in a partitioned system

- Snapshot isolation
  - Reads a consistent snapshot
- Consistent Snapshot
  - Includes all updates committed before snapshot time
- Transactions commit in total order
- Snapshot identified by its commit time
- Update A is causally before B if A.commit-time < B.commit-time

Clock SI – Snapshot Isolation using physical clocks

- Loosely synchronized clocks
- No centralized time-stamp generator
- Distributed protocol
- Snapshot-time
  - Time when transaction begins
  - Reads return values committed on or before this time
- Commit-time decided by transaction coordinator and partitions involved in transaction

ClockSI – Commit Protocol



ClockSI – Commit Protocol

Txn Coordinator

- T.snapshottime = Localclock = 8
- Send prepare to partitions
- Commit-time = max(11,9,10)
- Commit to partitions

Partitions

ClockSI – Commit Protocol

Txn Coordinator

- T.snapshottime = Localclock = 8
- Send prepare to partitions
- Commit-time = max(11,9,10)
- Commit to partitions

Partitions

ClockSI – Commit Protocol

Txn Coordinator

- T.snapshottime = Localclock = 8
- Send prepare to partitions
- Commit-time = max(11,9,10)
- Commit to partitions

- Receive Prepare
- Localclock = 11
- Reply 11
- Commit-time = 11

- Receive Prepare
- Localclock = 9
- Reply 9
- Commit-time = 11

- Receive Prepare
- Localclock = 10
- Reply 10
- Commit-time = 11

Partitions

ClockSI – Commit Protocol

Txn Coordinator

- T.snapshottime = Localclock = 8
- Send prepare to partitions
- Commit-time = max(11,9,10)
- Commit to partitions

- Receive Prepare
- Localclock = 11
- Reply 11
- Commit-time = 11

- Receive Prepare
- Localclock = 9
- Reply 9
- Commit-time = 11

- Receive Prepare
- Localclock = 10
- Reply 10
- Commit-time = 11

Partitions

ClockSI – Commit Protocol

Txn Coordinator

- T.snapshottime = Localclock = 8
- Send prepare to partitions
- Commit-time = max(11,9,10)
- Commit to partitions

Partitions

- Receive Prepare
- Localclock = 11
- Reply 11
- Commit-time = 11

- Receive Prepare
- Localclock = 9
- Reply 9
- Commit-time = 11

- Receive Prepare
- Localclock = 10
- Reply 10
- Commit-time = 11

ClockSI – Commit Protocol

Txn Coordinator

- T.snapshottime = Localclock = 8
- Send prepare to partitions
- Commit-time = max(11,9,10)
- Commit to partitions

Partitions

- Receive Prepare
- Localclock = 11
- Reply 11
- Commit-time = 11

- Receive Prepare
- Localclock = 9
- Reply 9
- Commit-time = 11

- Receive Prepare
- Localclock = 10
- Reply 10
- Commit-time = 11

ClockSI – Commit Protocol

Txn Coordinator

- T.snapshottime = Localclock = 8
- Send prepare to partitions
- Commit-time = max(11,9,10)
- Commit to partitions

- Receive Prepare
- Localclock = 11
- Reply 11
- Commit-time = 11

- Receive Prepare
- Localclock = 9
- Reply 9
- Commit-time = 11

- Receive Prepare
- Localclock = 10
- Reply 10
- Commit-time = 11

Partitions

Clock SI – Read protocol

Read(Transaction T, dataitem Obj)

- Wait if T.snapshotime > localclock
- If any pending Transaction T' with possible commit-time < T'.snapshottime
  - wait until T' is committed
- Return latest snapshot before snapshot-time

Extended ClockSI: Partitioned and Replicated System

- Vectorclock per partitio

	R1	R2	R3	R4
P1/R1	10	9	13	8

  - P1 at DC1 has seen all updates from DC2 before time 9
- Snapshot-time is Vectorclock of coordinator at the time when transaction begins
- Updates in a transaction depends on Snapshot which it reads from
- Snapshot-time encodes causal dependency

Extended ClockSI: Replication

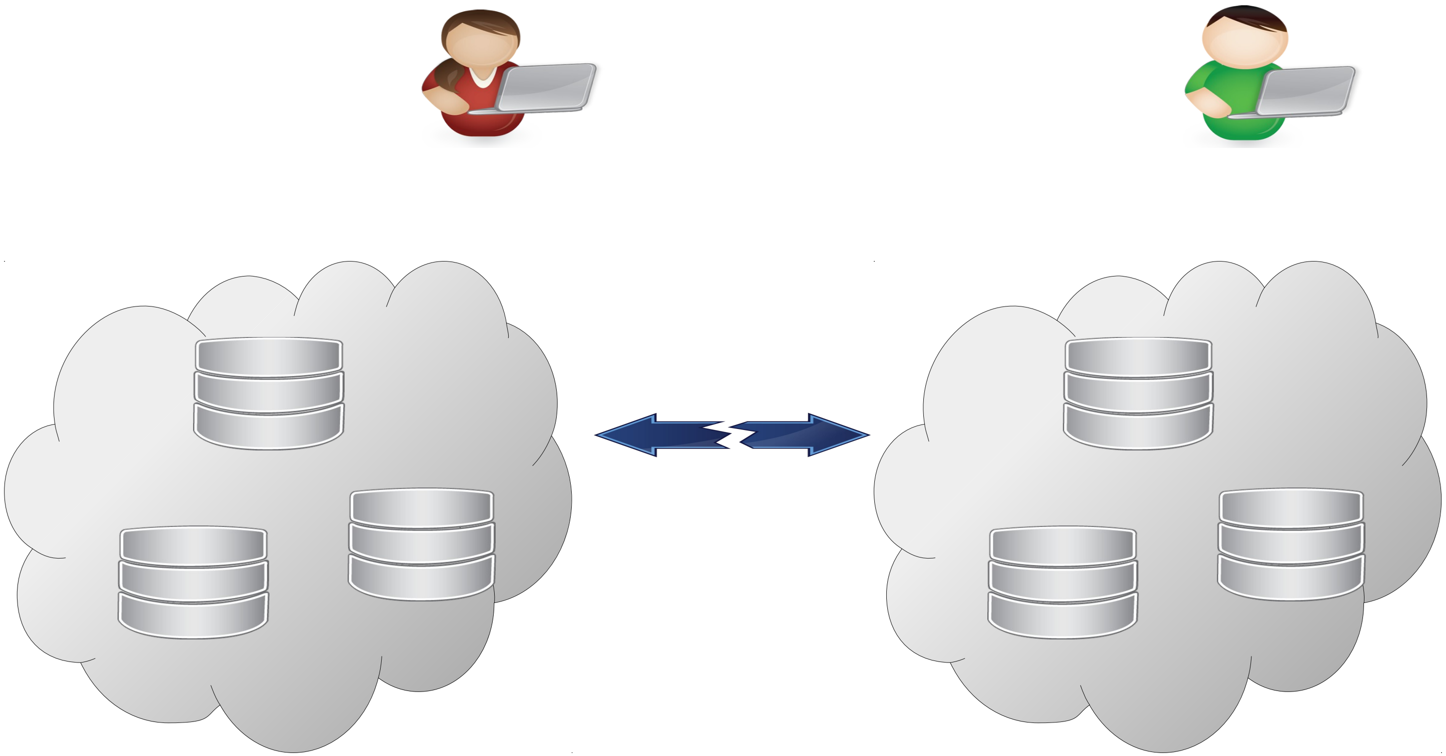
- P1 at DC1 sends updates to P1 at DC2 in *Commit-time order*
- Send snapshot-time and commit-time with every update
- On receiving an update U<DC, Commit-time, Snapshot-time> from a partition
  - Apply U if local vectorclock > Snapshot-time
  - Set vectorclock[DC] = Commit-time

Extended ClockSI: Read

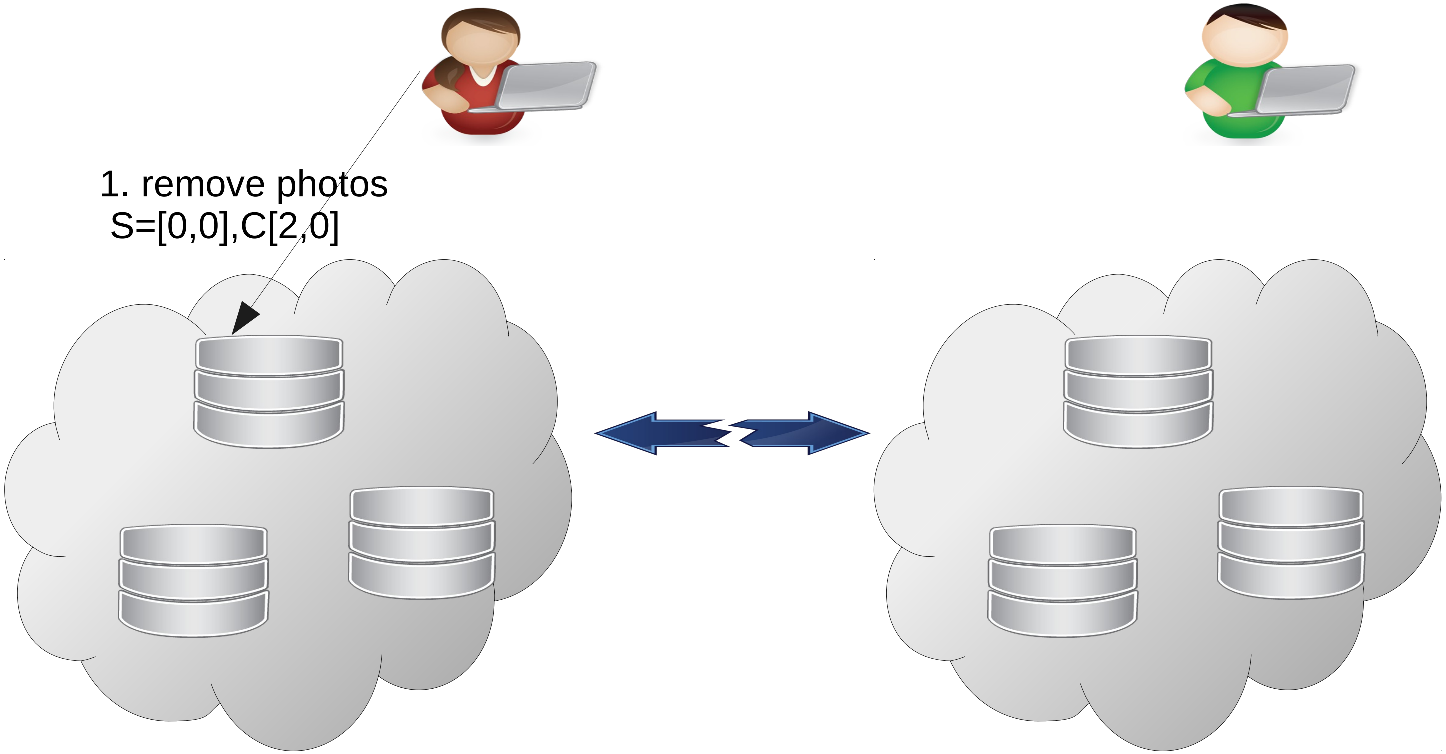
- Upon receiving a read request in a partition
  - Wait until local vectorclock >= snapshot-time
  - Return latest value before snapshot-time

- Causality metadata = O(N)
- No communication between partitions

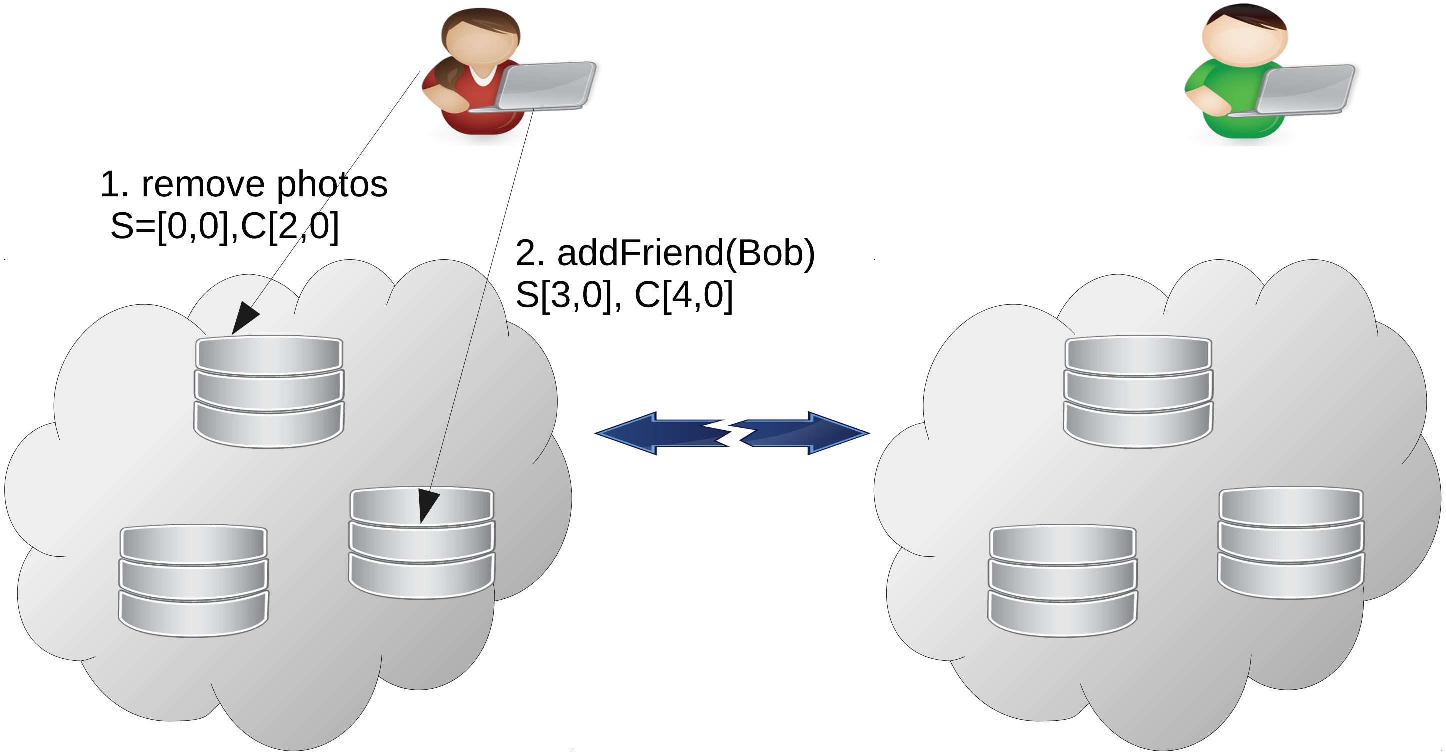
Social Network Application



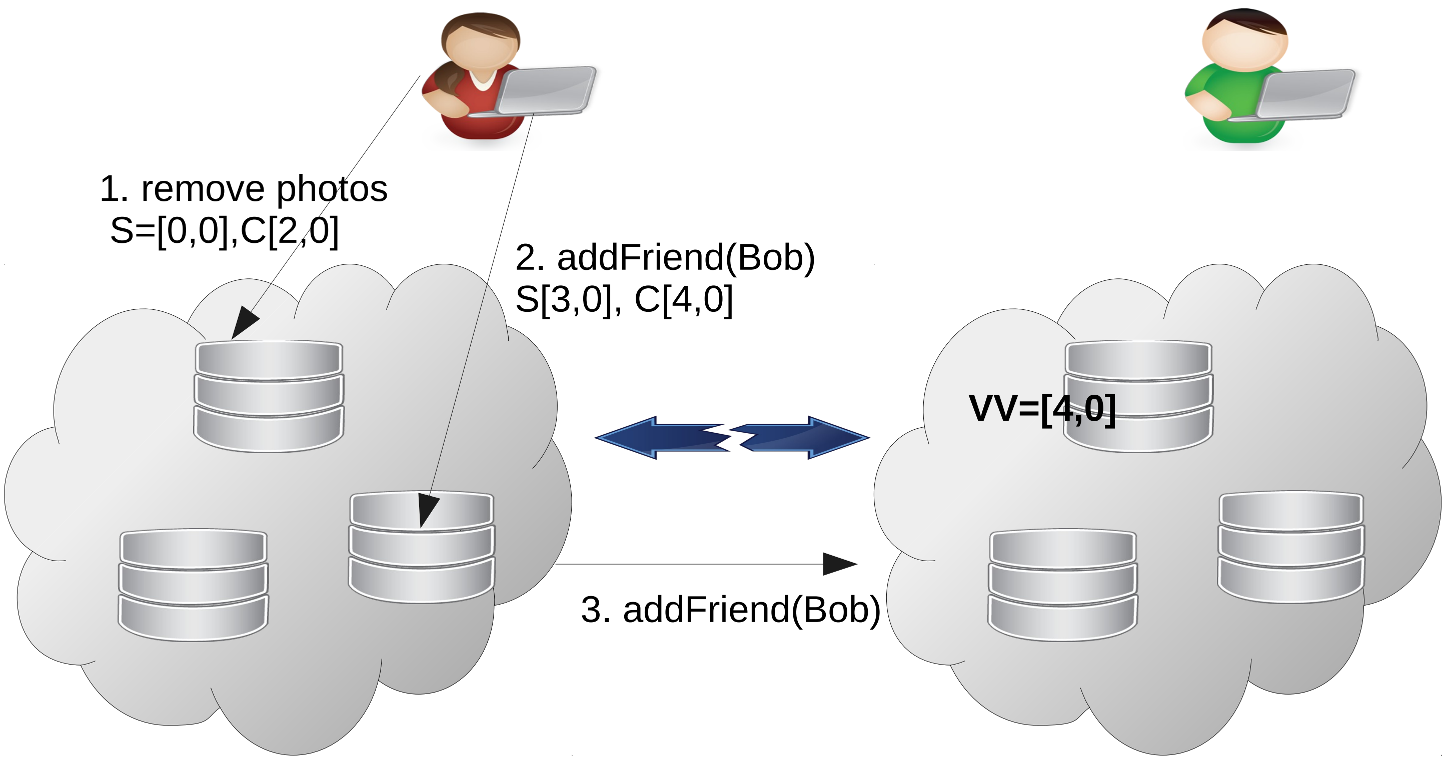
Social Network Application



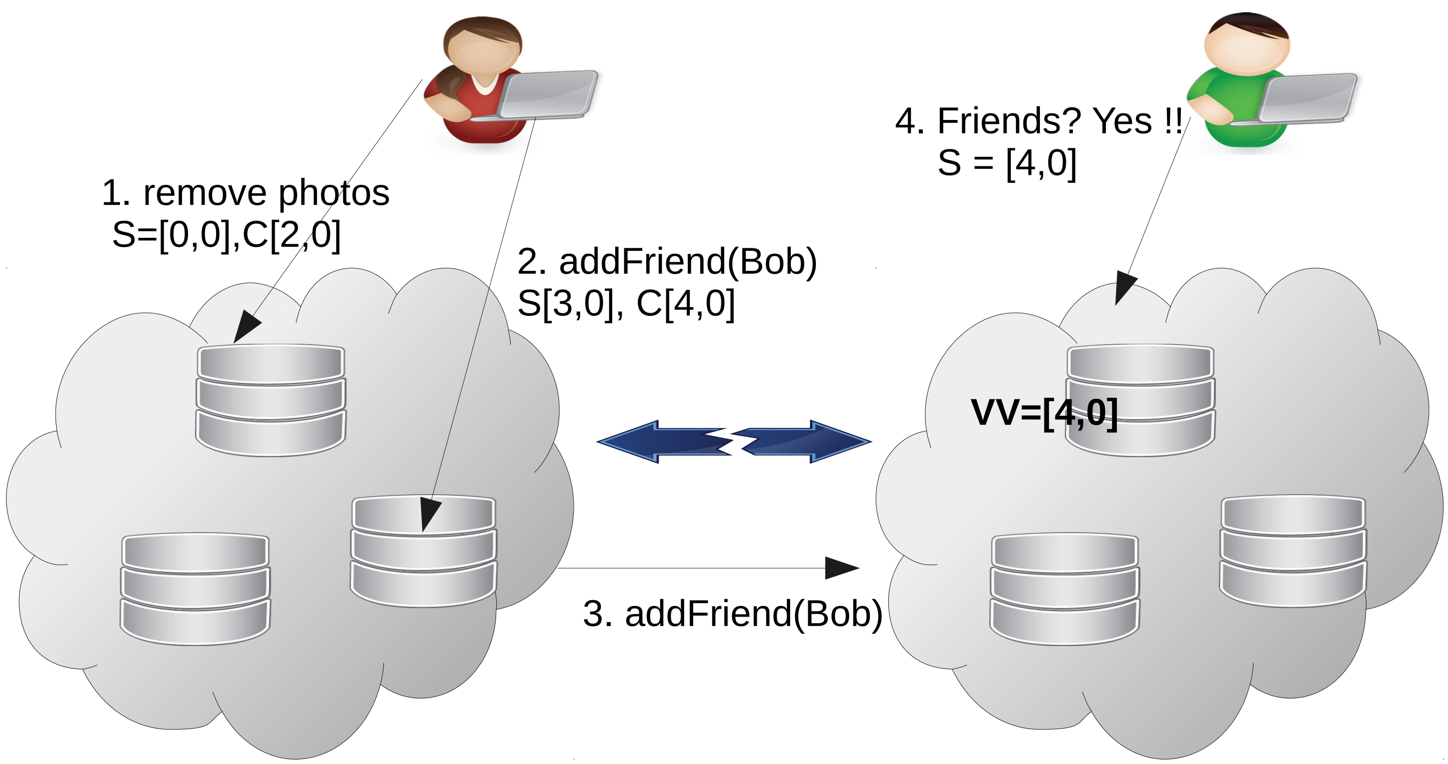
Social Network Application



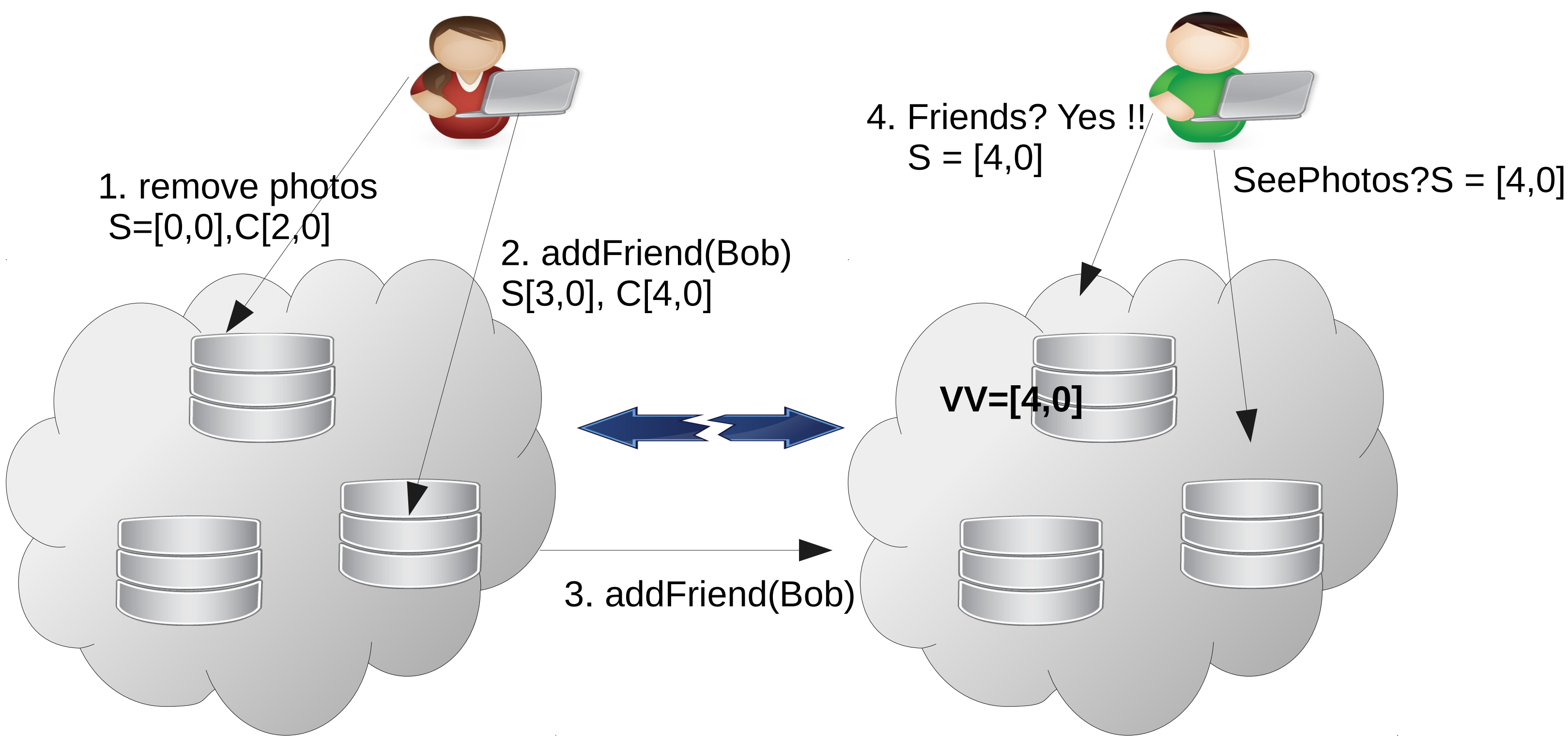
Social Network Application



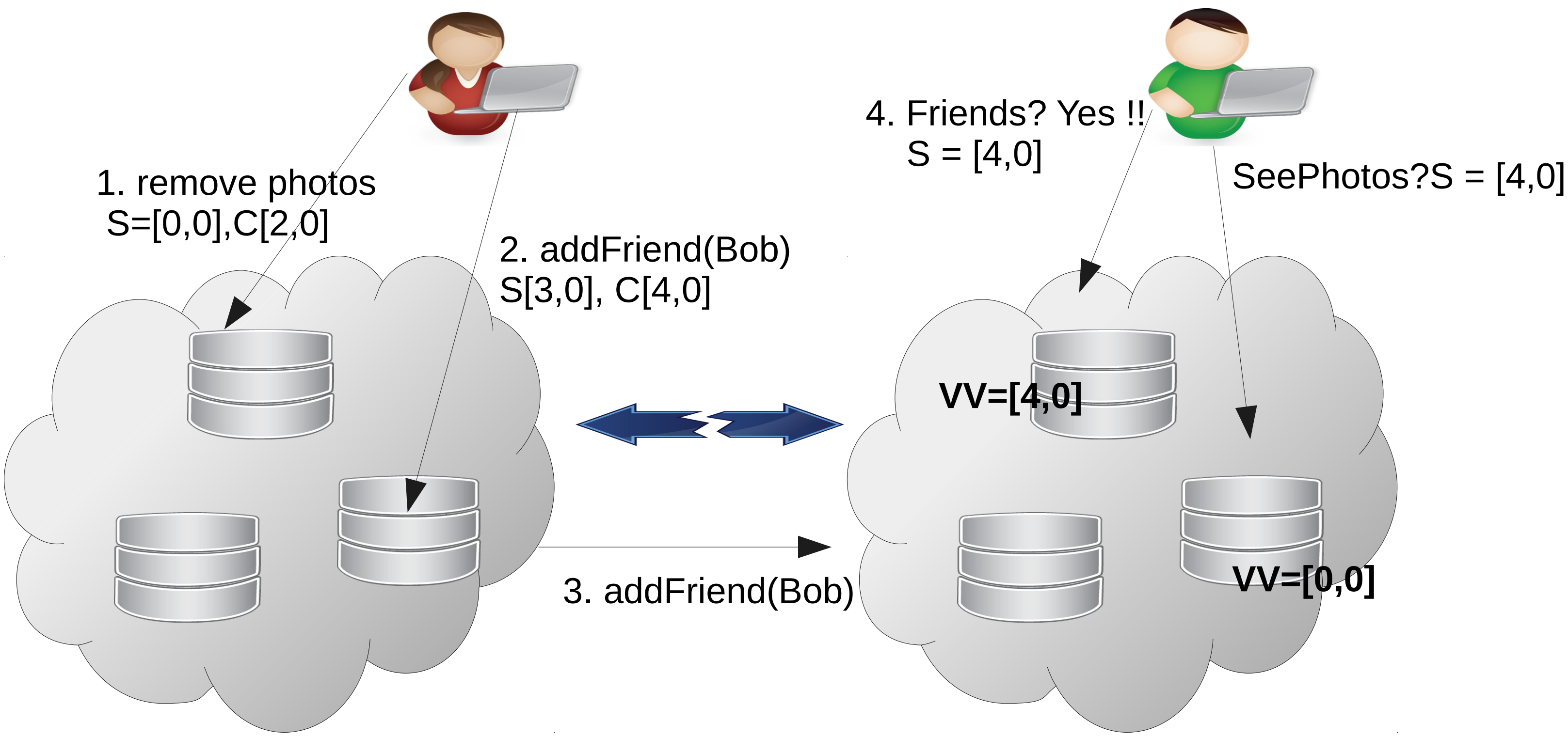
Social Network Application



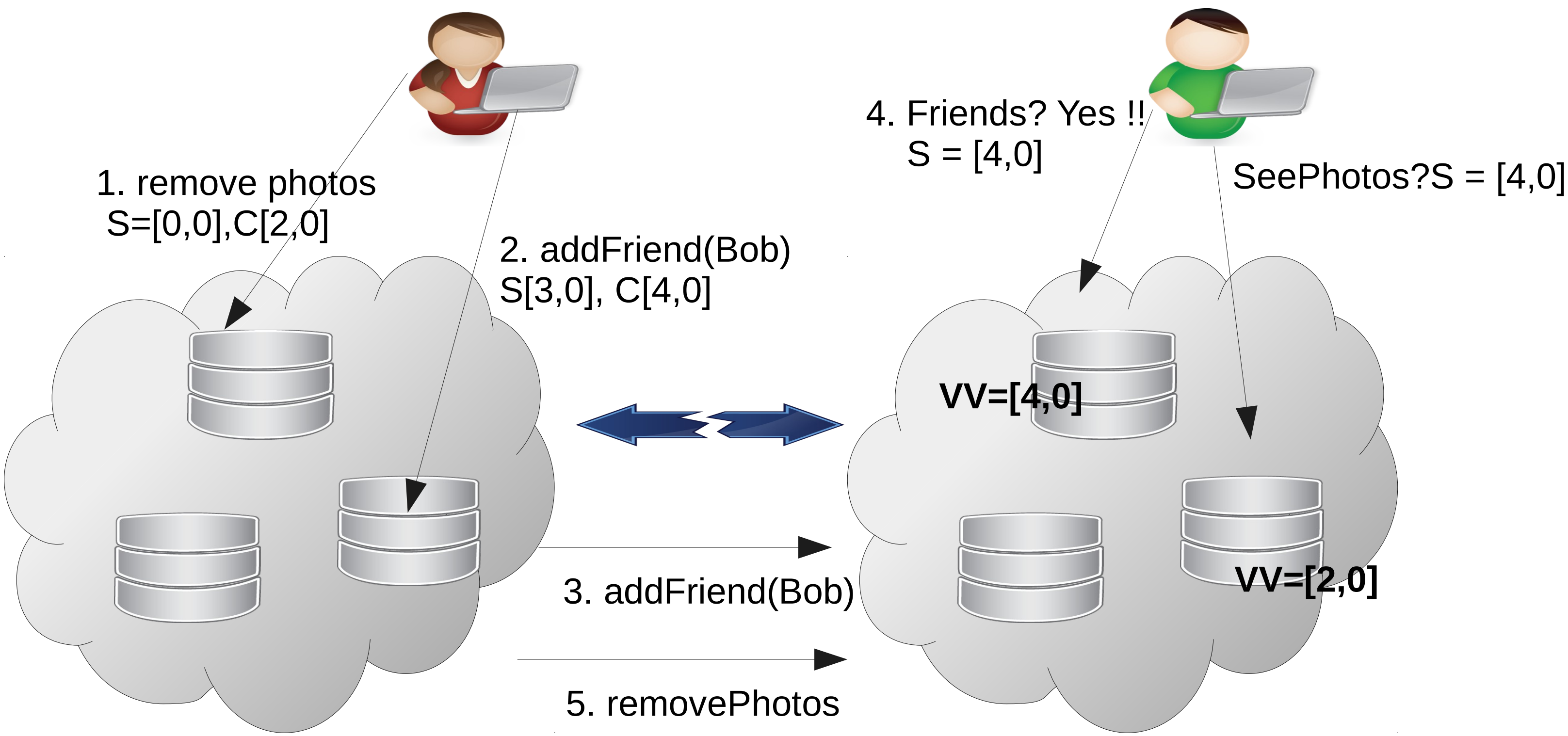
Social Network Application



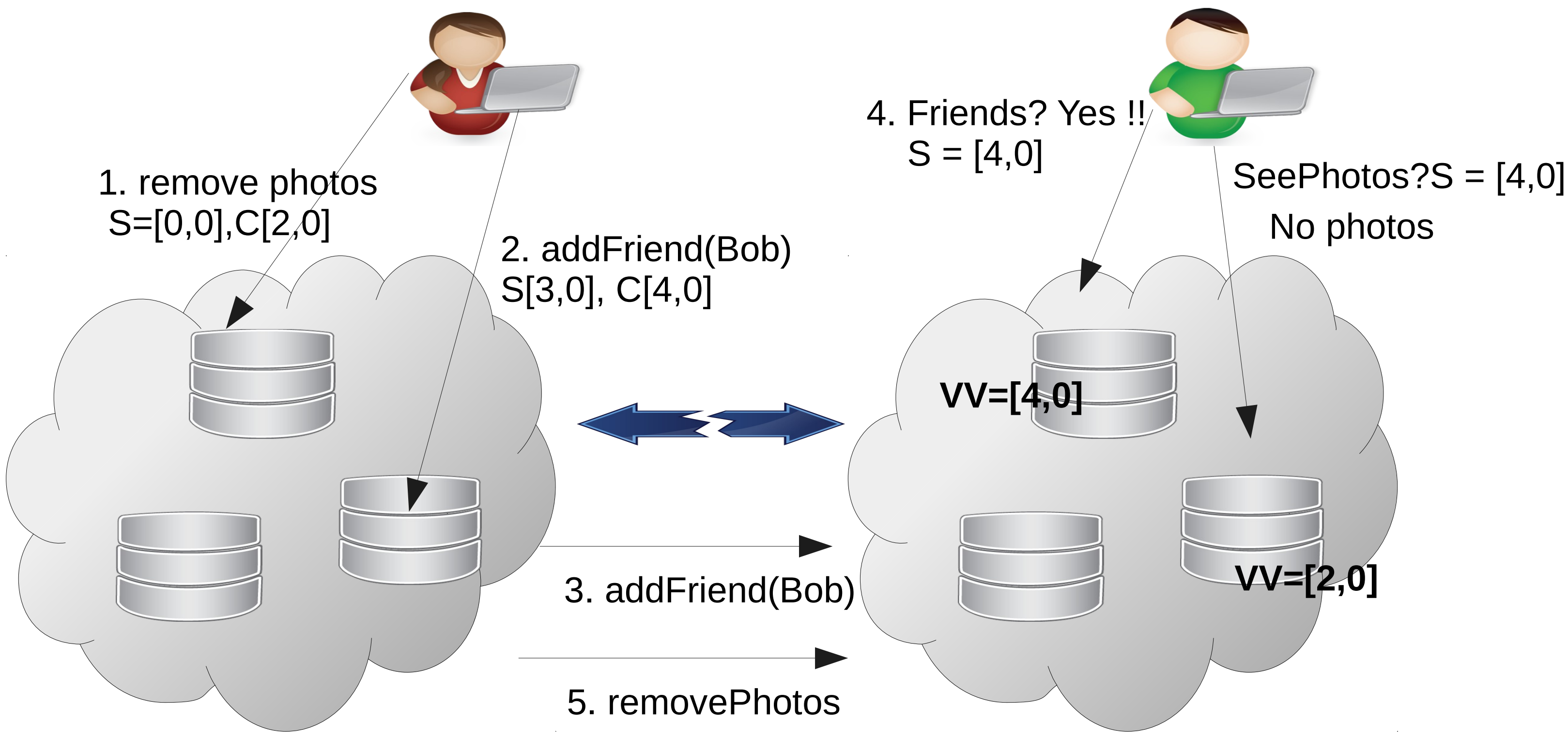
Social Network Application



Social Network Application



Social Network Application



Conclusion

- Total ordering using Lamport's timestamp
- Causality tracking using Vectorclocks
- Explicit causality tracking
  - Orbe using dependency matrix
  - ClockSI using physical clock and dependency vector

Reference

1. Leslie Lamport, 1978, "*Time,Clocks and the Ordering of Events in a Distributed System*", Communications of the ACM, Vol. 21
2. Colin J. Fidge, 1988, "*Timestamps in Message-Passing Systems That Preserve the Partial Ordering*". In K. Raymond (Ed.). Proc. of the 11th Australian Computer Science Conference (ACSC'88). pp. 56–66. Retrieved 2009-02-13.
3. D. Stott Parker et.al, "*Detection of mutual inconsistency in distributed systems*" Transactions on Software Engineering, 9(3):240–246, 1983.
4. B. Charron-Bost, "*Concerning the size of logical clocks in distributed systems*", Information Processing Letter, Vol. 39, 1991
5. Jiaqing Du et.al, "*Orbe: scalable causal consistency using dependency matrices and physical clocks*" SOCC'13 Proceedings of 4th annual Symposium on Cloud Computing, 2013
6. Jiaqing Du et.al, "*ClockSI: Snapshot Isolation for Partitioned Data Stored Using Loosely Synchronized Clocks*", SRDS'13 Proceedings of the 2013 IEEE International Symposium of Reliable Distributed Systems