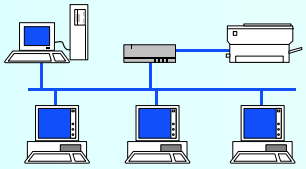


Distributed Systems
600.437
Replication

Department of Computer Science
The Johns Hopkins University

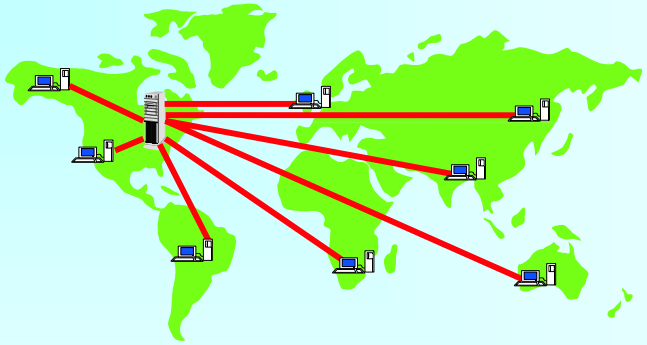
Replication

Lecture 8

Further readings:

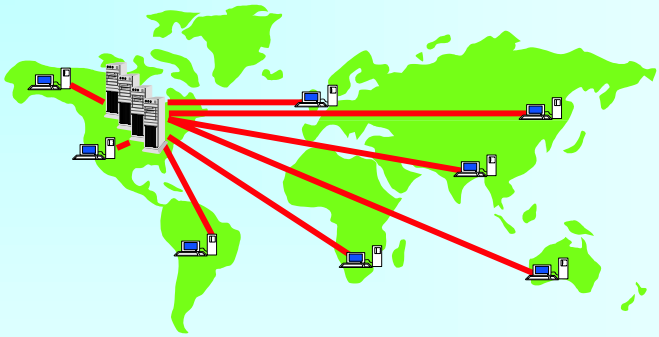
* Distributed Systems (Second edition) Sape Mullender, chapters 7 and 8. (Addison-Wesley) 1994.
* Concurrency control and recovery in Distributed Database Systems Bernstein, Hadzilacos and Goodman (Addison Wesley) 1987.
* Papers from ICDCS2002 and DISC98 on our www.cnds.jhu.edu/publications web page.

A Basic Client-Server Model



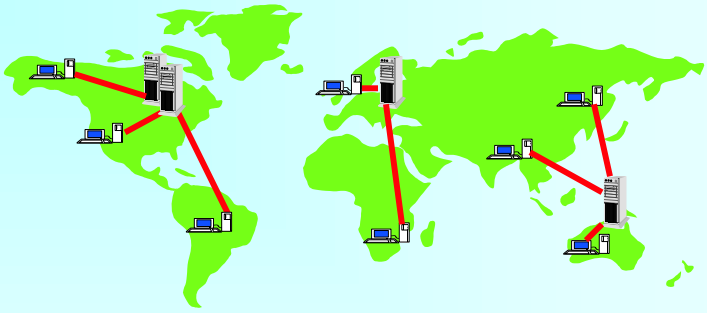
- Simple and easy to deploy.
- Single point of failure in the server.
- Single point of failure in the network.
- High latency for clients if server is busy.
- High latency for clients that are distant (network-wise).

Cluster replication



- Improves performance - the load is shared by several servers.
- Improves the availability of the server.
- Single point of failure in the network.
- High latency for clients that are distant (network-wise).

Wide Area Replication



- Replicating both processing ability and data.
- The Technical Challenge:
  - Maintain consistency among the replicated servers, supporting high volume of dynamic updates in a timely fashion.

Replication

- Benefits of replication:
  - **High Availability.**
  - **High Performance.**
- Costs of replication:
  - **Synchronization.**
- Requirements from a generic solution:
  - Strict consistency.
  - Sometimes too expensive so requirements are tailored to applications.

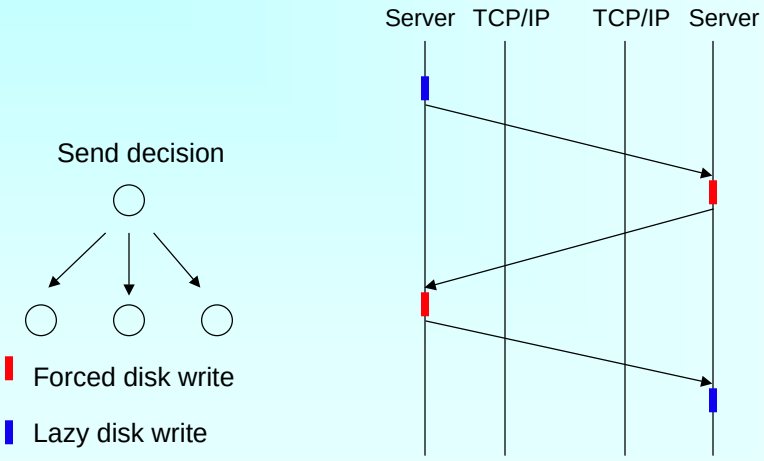
Replication Methods

- Two phase commit, three phase commit
- Primary and backups
- Weak consistency (weaker update semantics)
- Primary component.
  - What happens when there is no primary component?
- Replication using group communication.

Two Phase Commit

- Built for updating distributed databases.
- Can be used for the special case of replication.
- Consistent with generic update model.
- Relatively expensive.

Two Phase Commit

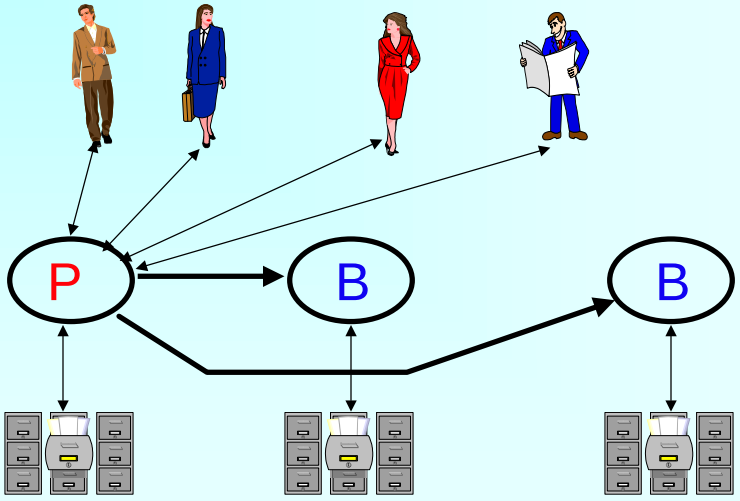


Primary and Backups

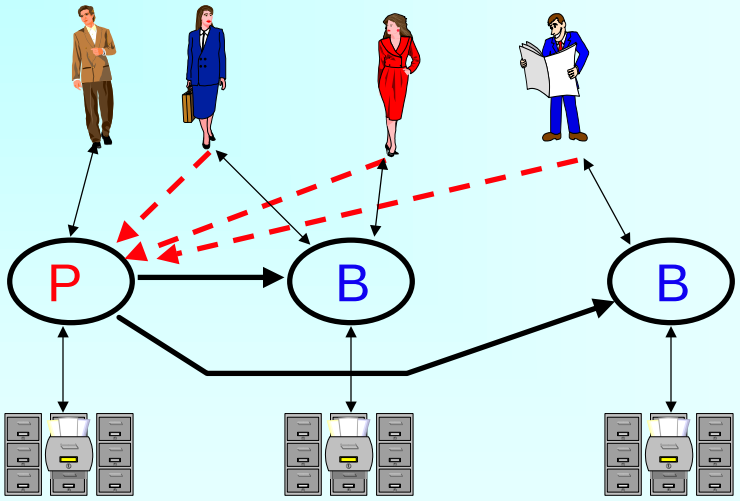
Possible options:

- Backups are maintained for availability only.
- Backups can improve performance for reads, updates are sent to the primary by the user.
  - **What is the query semantics? How can one copy serializability be achieved?**
- The user interacts with one copy, and if it is a backup, the updates are sent to the primary
  - **What is the query semantics with regards to our own updates?**

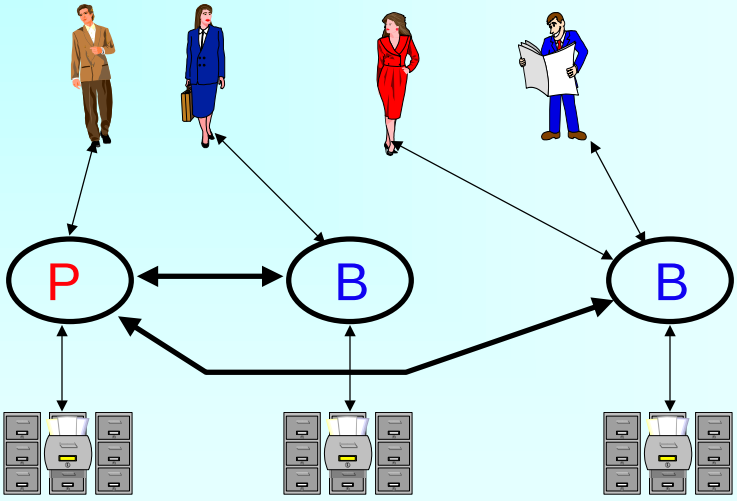
Primary and Backups (1)



Primary and Backups (2)



Primary and Backups (3)



Weak Consistency (weaker update semantics)

The Anti-Entropy method: Golding 92

- State kept by the replication servers can be weakly consistent. i.e. copies are allowed to diverge temporarily. They will eventually come to agreement.
- From time to time, a server picks another server and these two servers exchange updates and converge to the same state.
- Total ordering is obtained after getting one message from every server (directly).
- Lamport time stamps are used to order messages.

The Anti-Entropy method

Knowledge at Server A

A	1	3	5	12
B	2			
C	2	3	4	

Knowledge at Server B

A	1	3			
B	2	5	6	9	11
C	2				

Summary A

12
2
4

Summary B

3
11
2

Numbers refer to Lamport time stamps.

The Anti-Entropy Method (cont.)

Knowledge at Server A

A	1	3	5	12
B	2			
C	2	3	4	

Knowledge at Server B

A	1	3			
B	2	5	6	9	11
C	2				

Summary A

12
2
4

Summary B

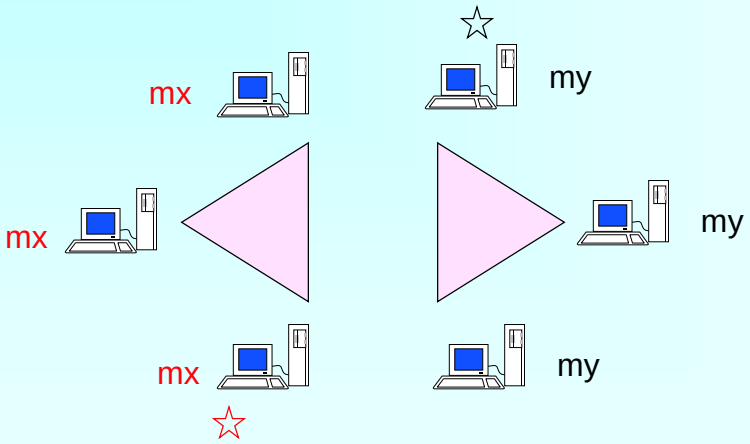
3
11
2

Summary After merge

12
11
4

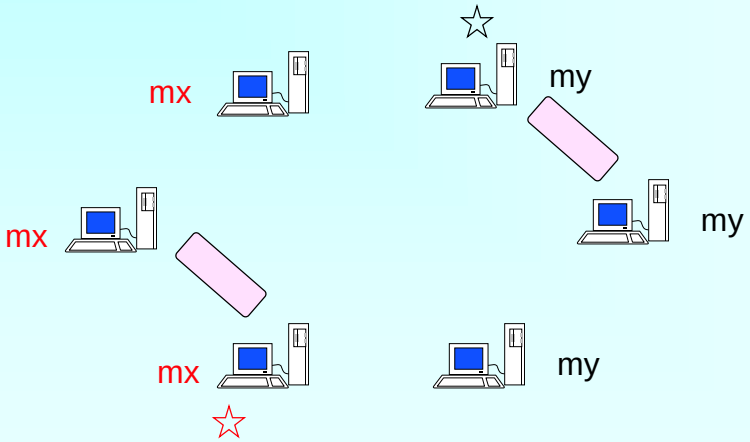
Eventual Path Propagation

Partitioned system



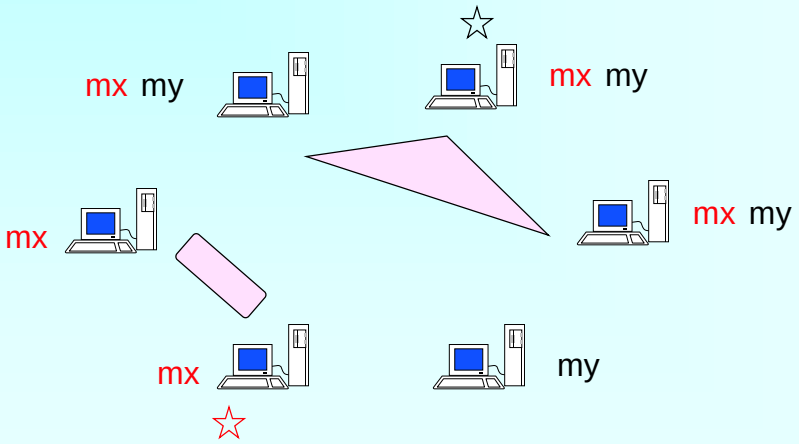
Eventual Path Propagation (cont.)

Further partitioning



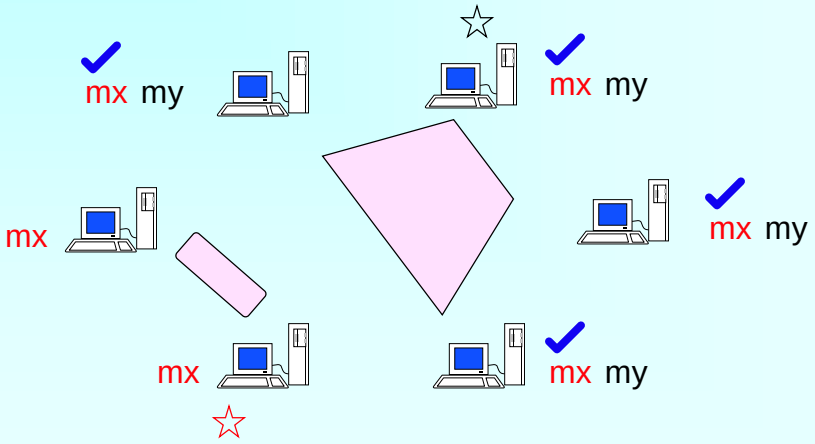
Eventual Path Propagation (cont.)

Merging

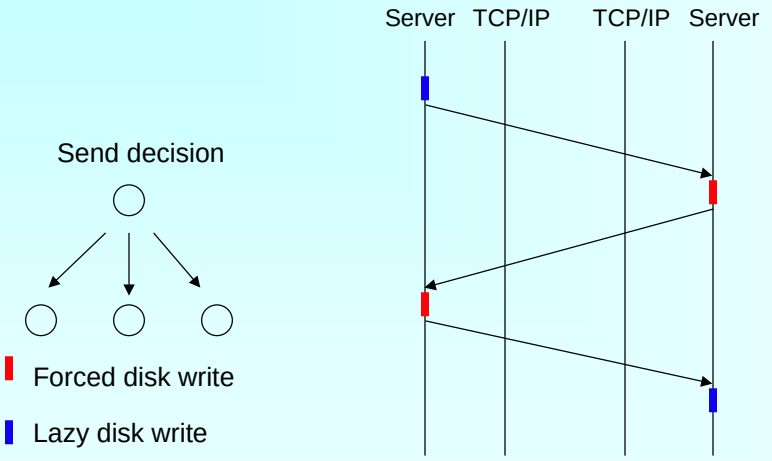


Eventual Path Propagation (cont.)

Further merging



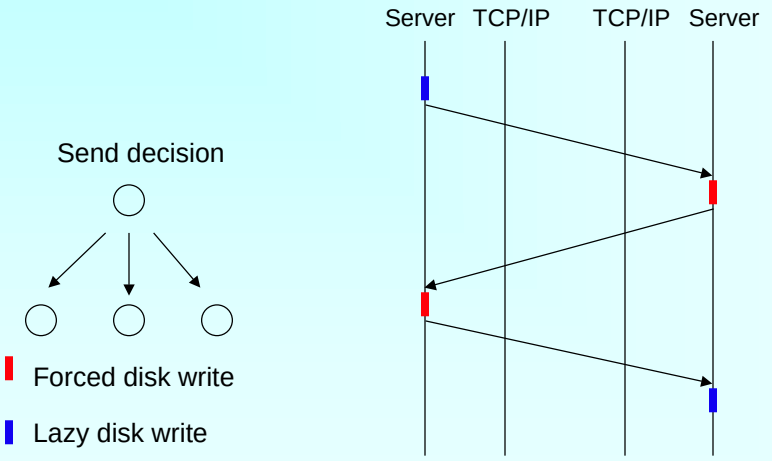
Going back to 2 Phase Commit



Primary Component

- A quorum can proceed with updates.
- When the network connectivity changes, if there is a quorum, the members can continue with updates.
- Dynamic methods will allow the next quorum to be formed based on the current quorum (for example – the next quorum is a majority of the current quorum).

Going back to 2 Phase Commit



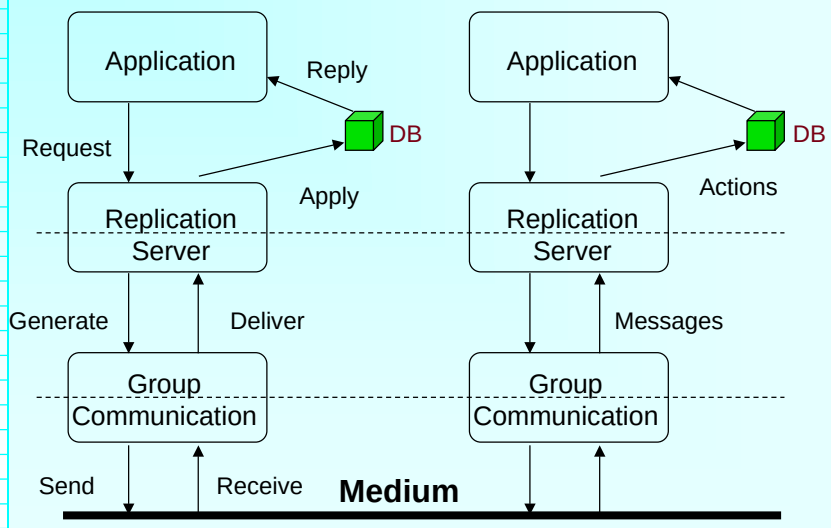
What can be improved?

- Reduce number of forced writes to disk
- Reduce number of messages
  - Aggregate acknowledgements
- Avoid end-to-end (application to application) acknowledgements
- Robustness

Group Communication "Tools"

- Efficient message delivery
  - Group multicast
- Message delivery/ordering guarantees
  - Reliable
  - FIFO/Causal
  - Total Order
- Partitionable Group Membership
- Strong semantics (what is actually needed?)

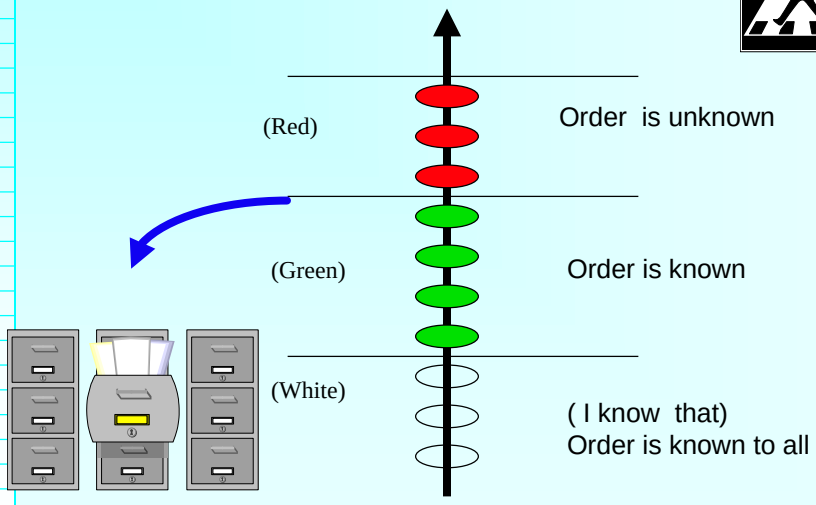
Replication using Group Communication



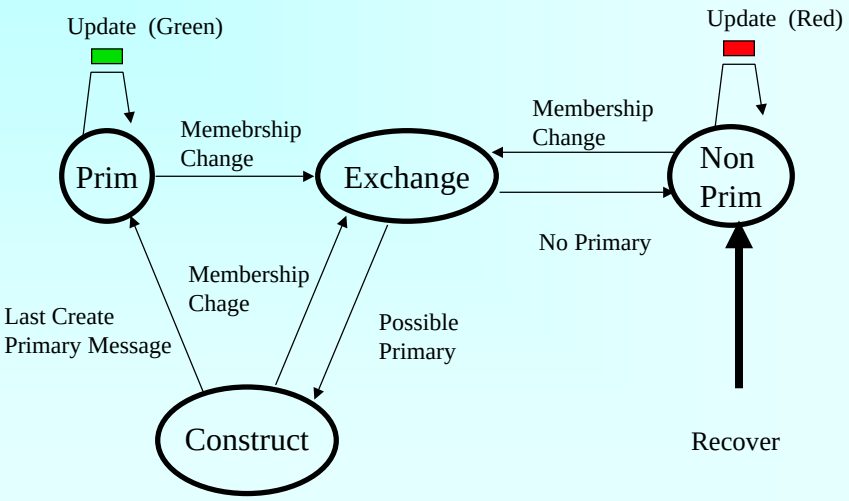
The Basic Idea

- Reduce database replication to **G**lobal **C**onsistent **P**ersistent **O**rder
  - Use group communication ordering to establish the Global Consistent Persistent Order on the updates.
  - deterministic + serialized = consistent
- Group Communication membership + quorum = **primary** partition.
  - Only replicas in the **primary** component can commit updates.
  - Updates ordered in a primary component are marked **green** and applied. Updates ordered in a non-primary component are marked red and will be delayed.

Action Ordering

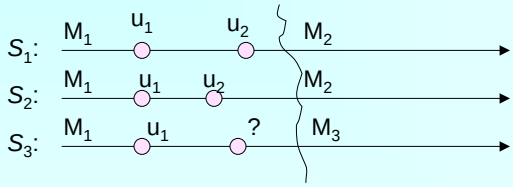


Conceptual State Diagram



Not so simple…

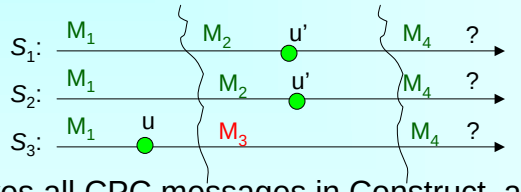
- **VS**: If s_1 and s_2 move directly from membership M_1 to M_2, then they deliver the same ordered set of messages in M_1.
  - What about s_3 that was part of M_1but is not part of M_2?



- Total (Agreed) Order **with no holes** is not guaranteed across partitions or server crashes/recoveries!

Delicate Points

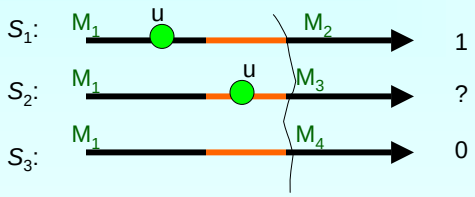
- s_3 receives update *u* in Prim and commits it right before a partition occurs, but s_1 and s_2 do not receive *u*. If s_1 and s_2 will form the next primary component, they will commit new updates, without knowledge of *u*!!



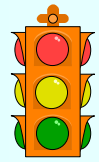
- s_1 receives all CPC messages in Construct, and moves to Prim, but one of the servers that were with s_1 in Construct does not receive the last CPC message. A new primary is created possibly without having the desired majority!!

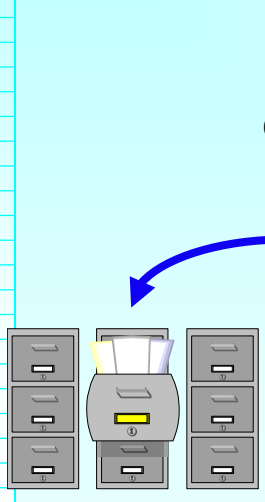
Extended Virtual Synchrony

- Transitional/Regular membership notification
- Safe message = Agreed plus every server in the current membership will deliver the message unless it crashes.
- Safe delivery breaks the two-way uncertainty into 3 possible scenarios, the extremes being mutually exclusive!



Action Ordering





(Red) Order is unknown

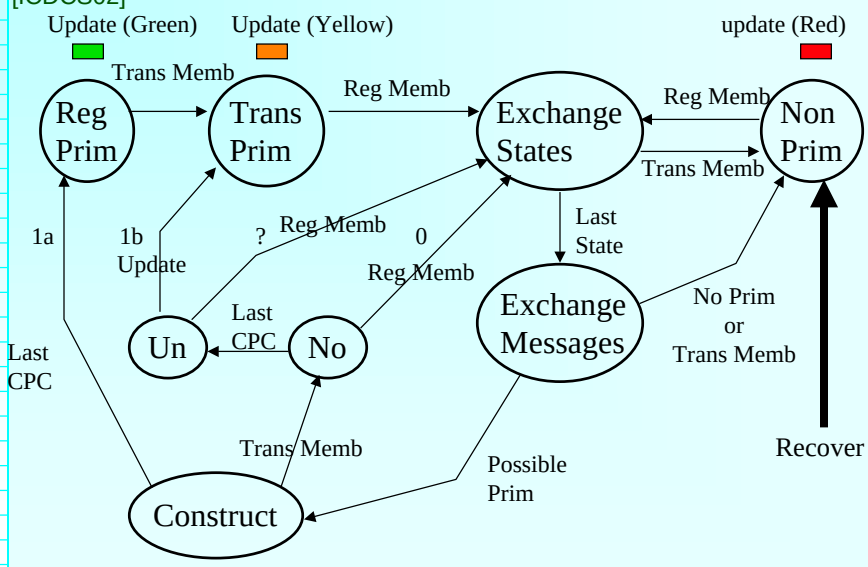
(Yellow) Transitional membership

(Green) Order is known

(White) (I know that)
Order is known to all

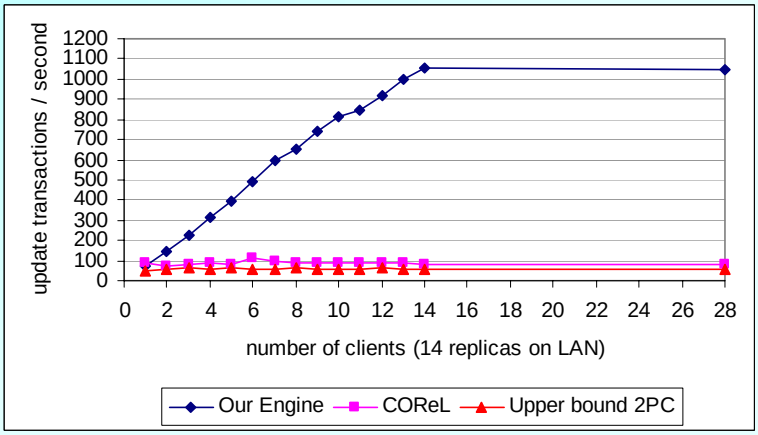
Replication Engine - State Diagram

[ICDCS02]



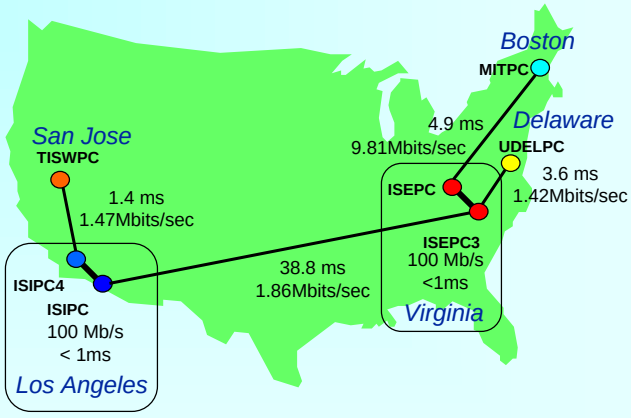
Throughput Comparison (LAN)

[ICDCS02]



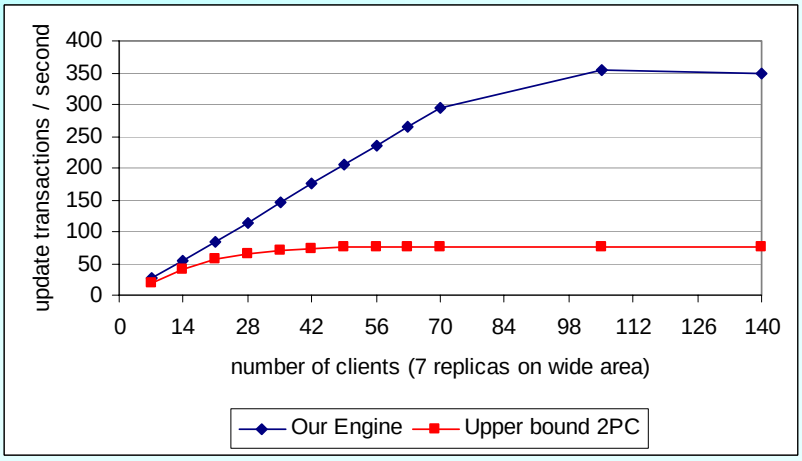
COReL – by Dolev and Keidar, using Agreed order + additional round.

The CAIRN Wide-Area Network



- A real experimental network (CAIRN).
- Was also modeled in the Emulab facility.

Throughput Comparison (WAN)



Replication Server Summary

- Knowledge propagation
  - Eventual Path Propagation.
- Amortizing end-to-end acknowledgments
  - Low level Ack derived from Safe Delivery of group communication.
  - End-to-end Ack upon membership changes.
- Primary component selection
  - Dynamic Linear Voting.