Google's Spanner

Pramod Bhatotia
http://homepages.inf.ed.ac.uk/pbhatoti/

Credits for the lecture material:
Spanner OSDI'12 paper and presentation

THE UNIVERSITY of EDINBURGH

What is Spanner?

Bird's-eye view of Spanner

- Globally-distributed scalable multi-version database
- Synchronous replication (using Paxos SMR)
- Externally-consistent (linearizable) distributed transactions
- It's just tip of iceberg (supports many more features…)

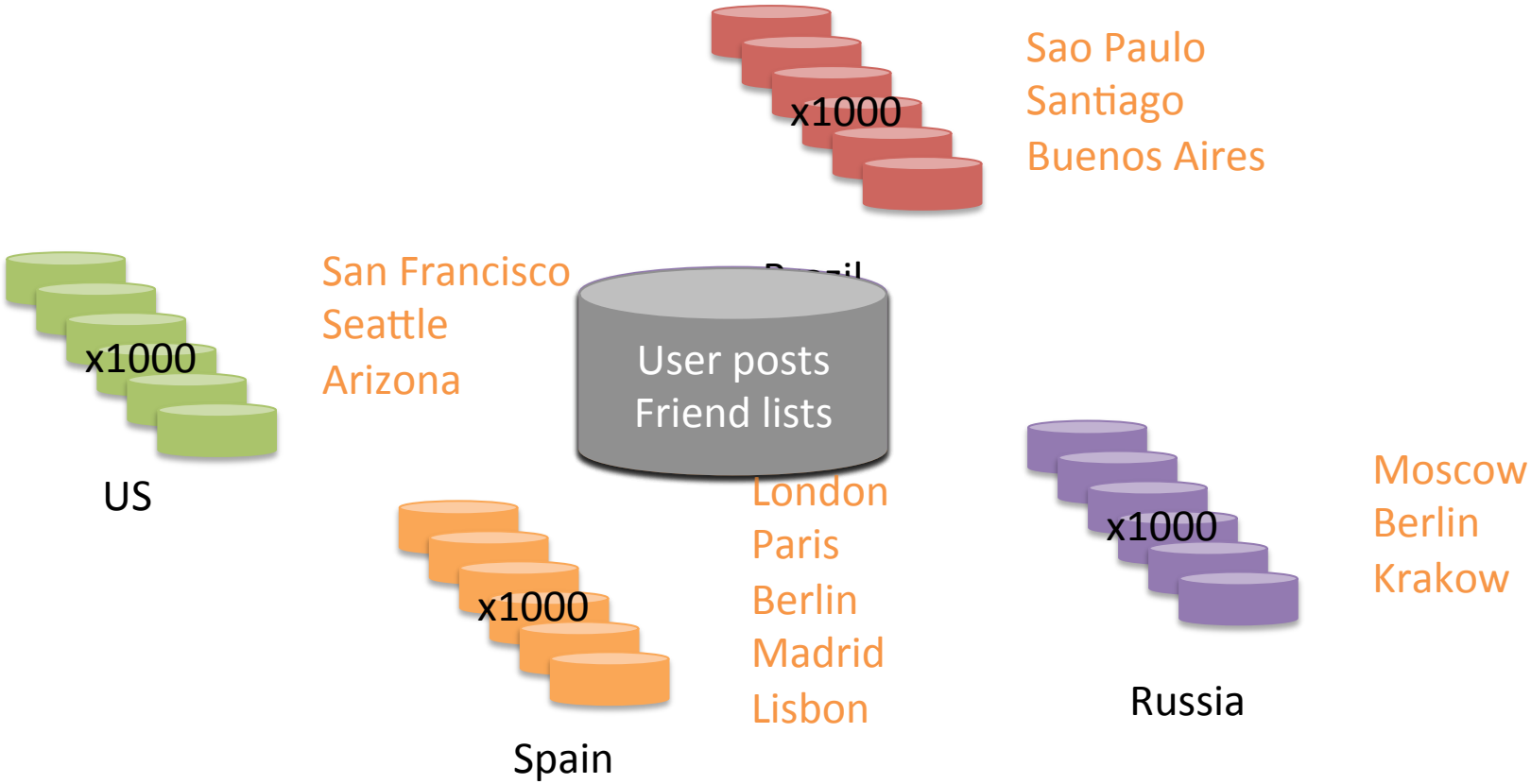
What is the problem being solved?

Building a global scale distributed database with transactional support to help application programmers

Why is it challenging?

Distributed transactions with strong semantics at global scale is difficult

Example: Social network data



So what?

Wasn’t BigTable* built to support such cases?

*BigTable [OSDI ’06]

Complex application requirements!

- An example operation for OSN application:
  - Remove untrustworthy person X as friend
  - Post P: “My government is repressive…”
- Consistency matters!
  - Generate a page of friends’ recent posts
  - Consistent view of friend list and their posts
- Require transactions
  - BigTable doesn’t support transactions!

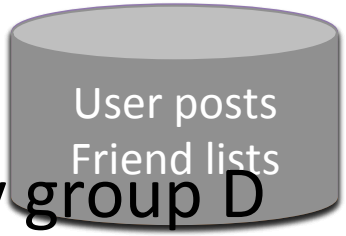
Hmm…
then what about MegaStore*?

*MegaStore[CIDR ’11]

MegaStore

Entity group B

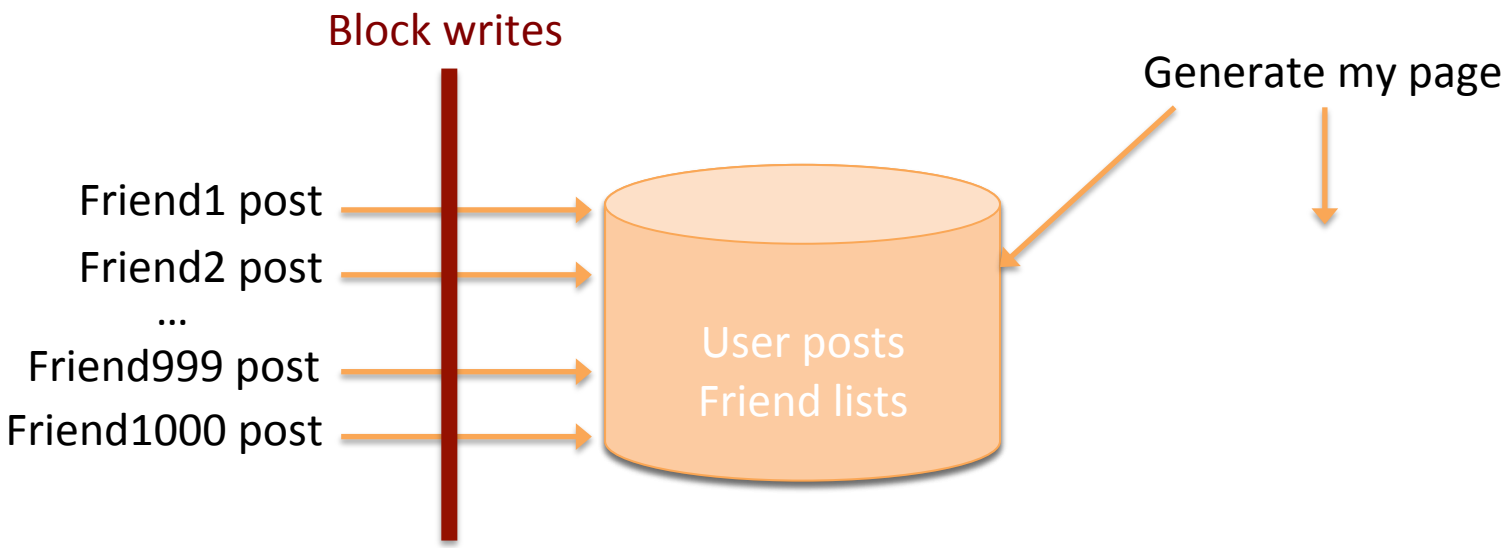
Entity group A



Entity group C

Entity group D

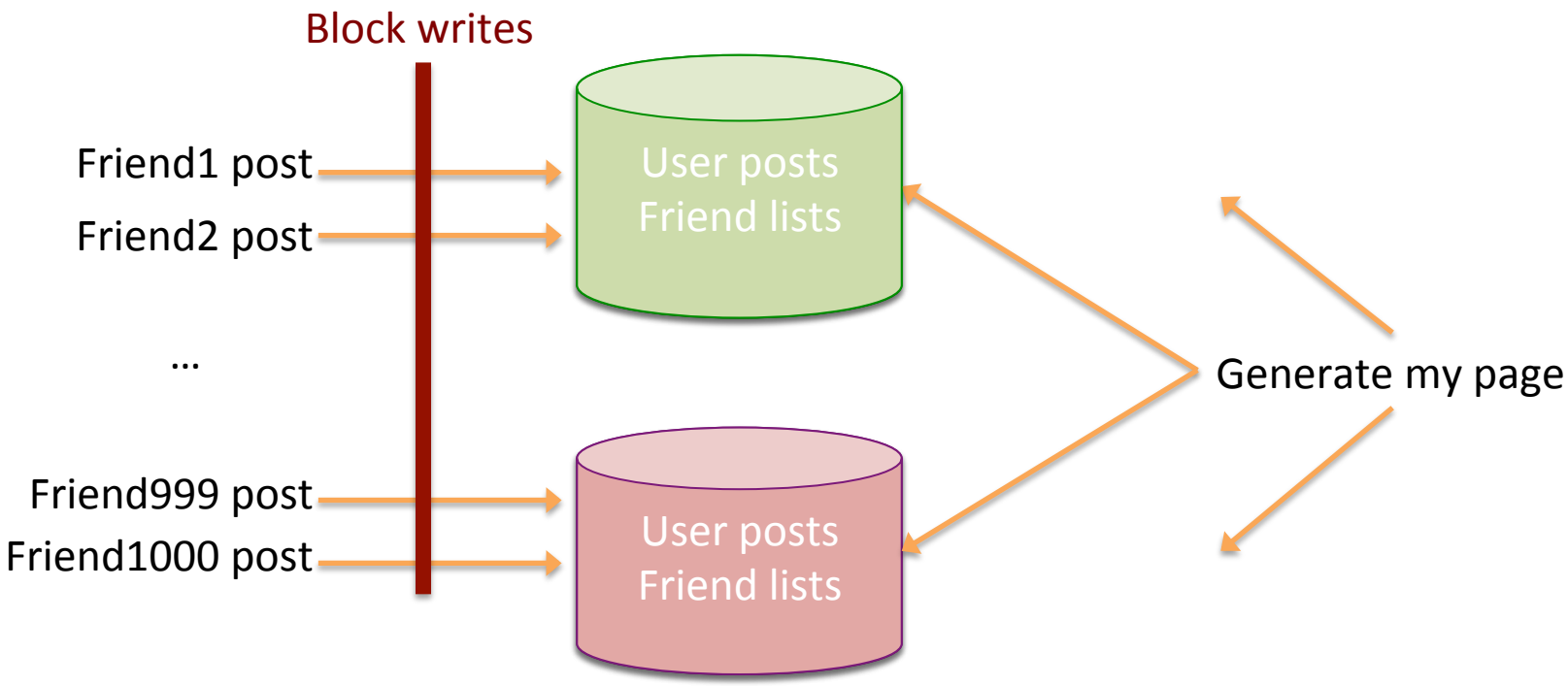
Transaction support in Megastore



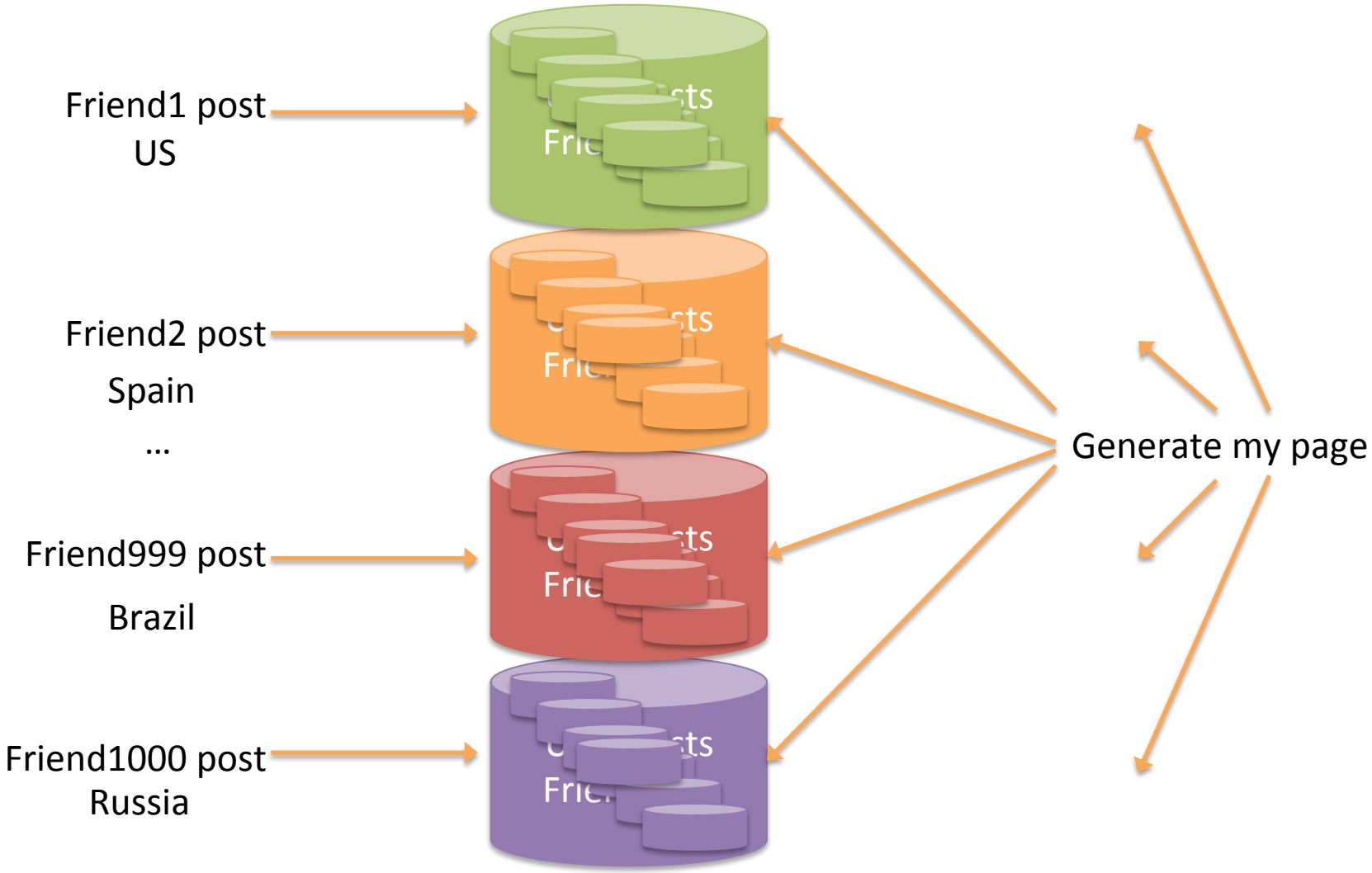
- Supports transactions within entity groups
- Between entity groups require 2PC (using async queue)

Low write throughput!

Multiple entity groups



Multiple data-centers



Let’s take a step back!

What property Spanner is trying to achieve?

Externally consistent transactions
(linearlizability)

Externally consistent transactions

- Transactions that write use strict 2PL
  - Each transaction *T* is assigned a timestamp *s*
  - Data written by *T* is timestamped with *s*

Time	<8	8	15
My friends	[X]	[]	
My posts			[P]
X's friends	[me]	[]	

If a transaction T1 commits before another transaction T2 starts, then T1's commit timestamp is smaller than T2's timestamp.

What are the key insights?

Key idea: Synchronize snapshots

Global wall-clock time

==

External Consistency:

Commit order respects global wall-time order

==

Timestamp order respects global wall-time order

given

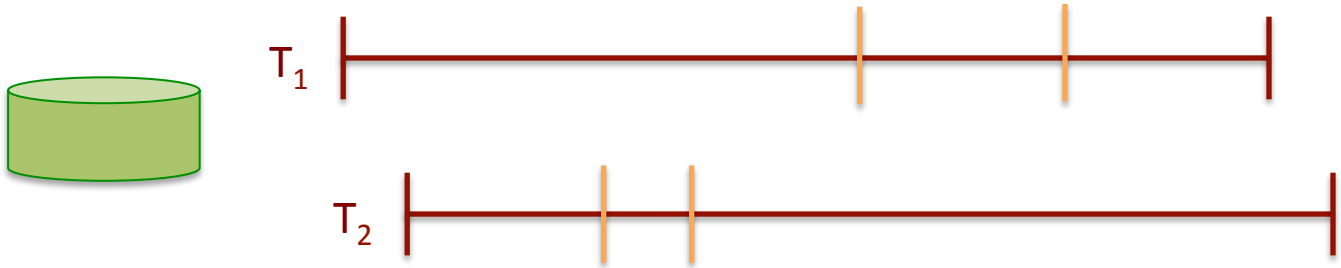
timestamp order == commit order

How would you do it for single machine?

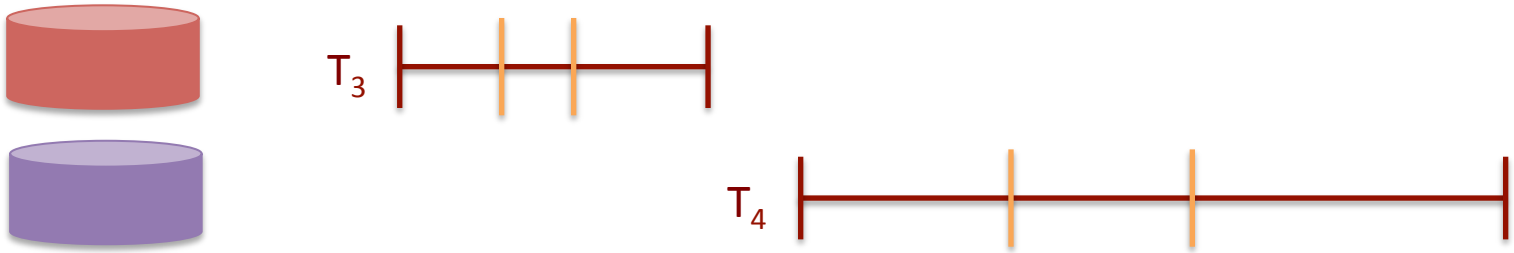
- Strict two-phase locking for write transactions
- Assign timestamp while locks are held

Timestamp Invariants

- Timestamp order == commit order

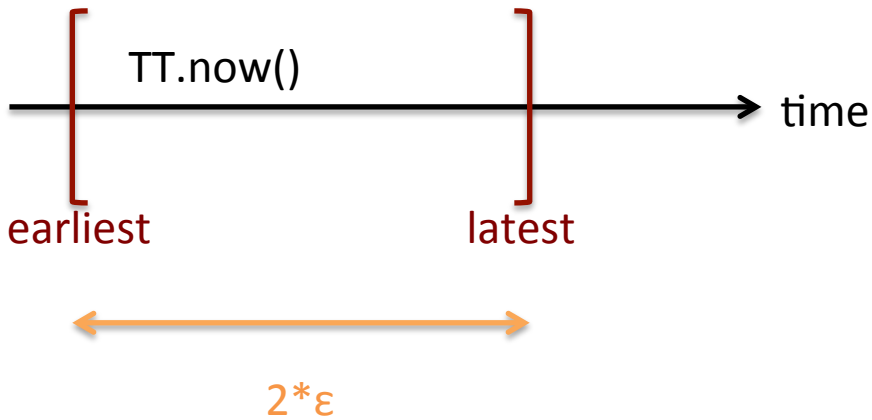


- Timestamp order respects global wall-time order



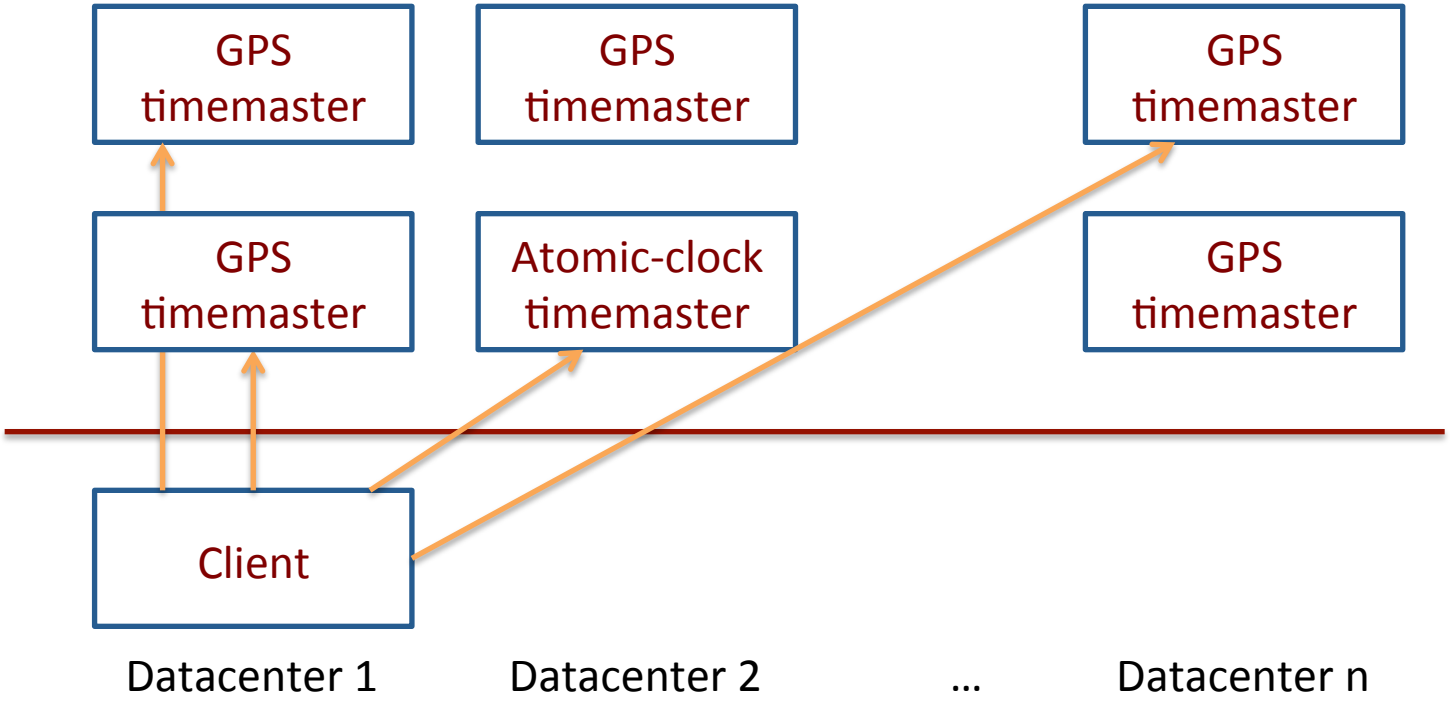
TrueTime

- A novel time API that exposes clock uncertainty



"Global wall-clock time"

TrueTime Architecture

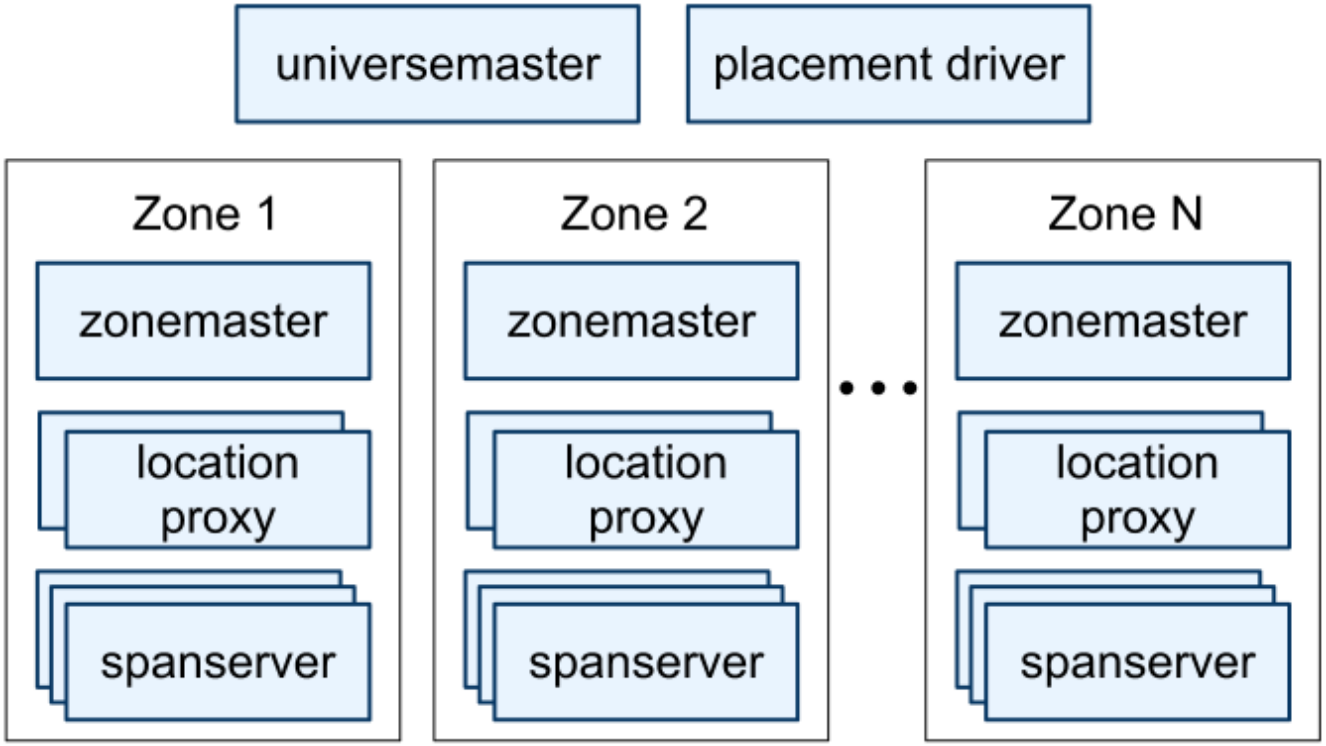


Compute reference [earliest, latest] = now ± ε

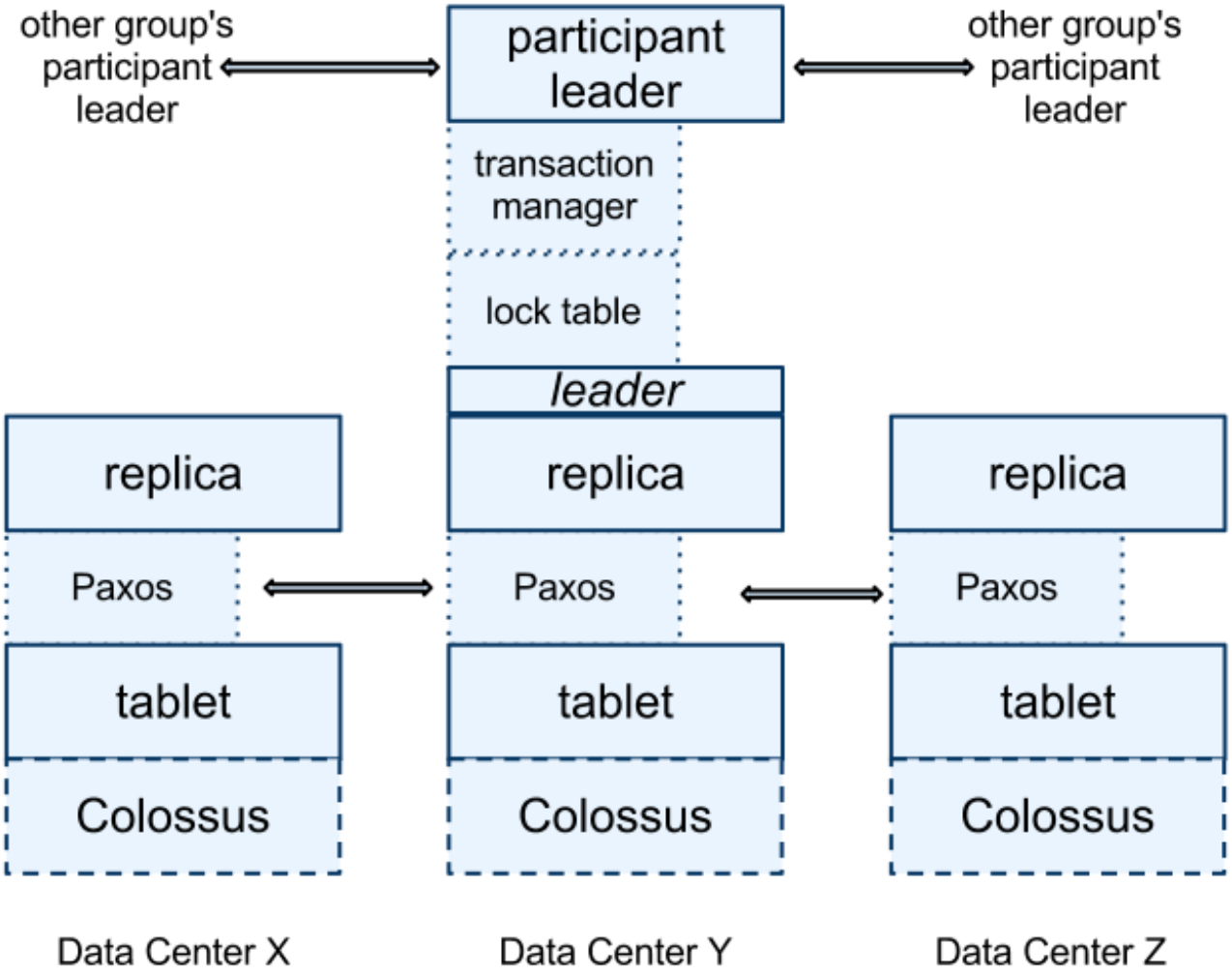
How does it works?

One line answer:
Running 2-PC over Paxos

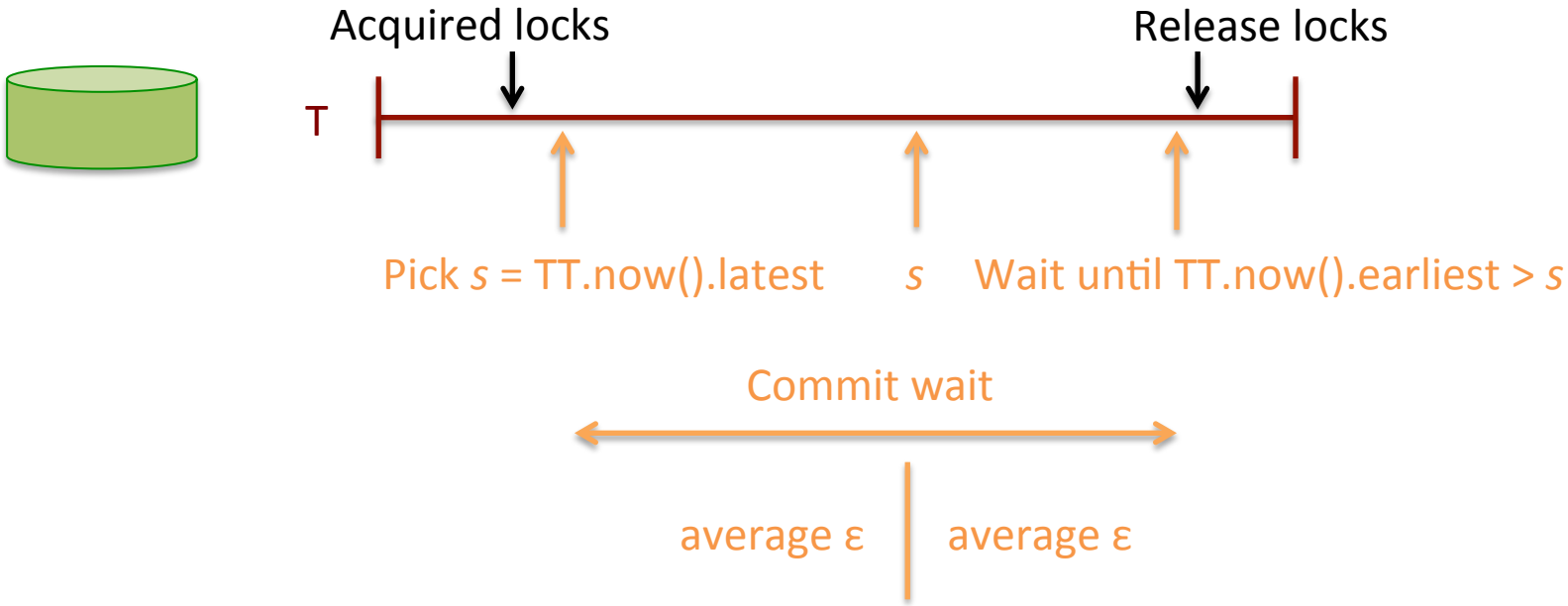
Spanner server organization



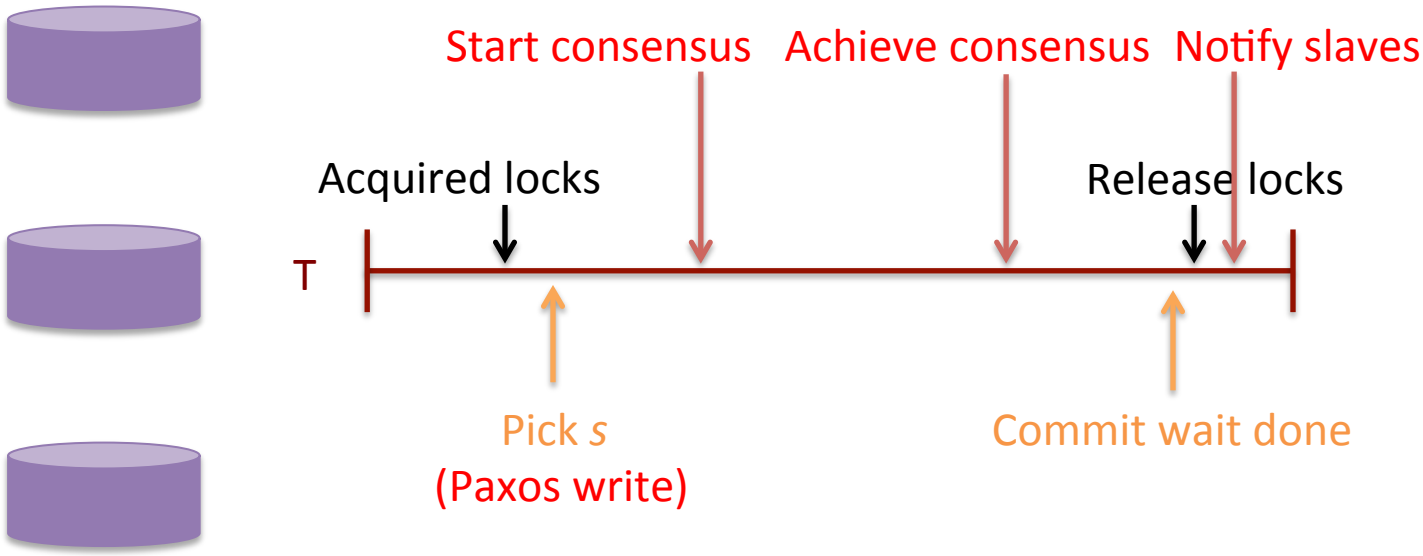
SpanServer software stack



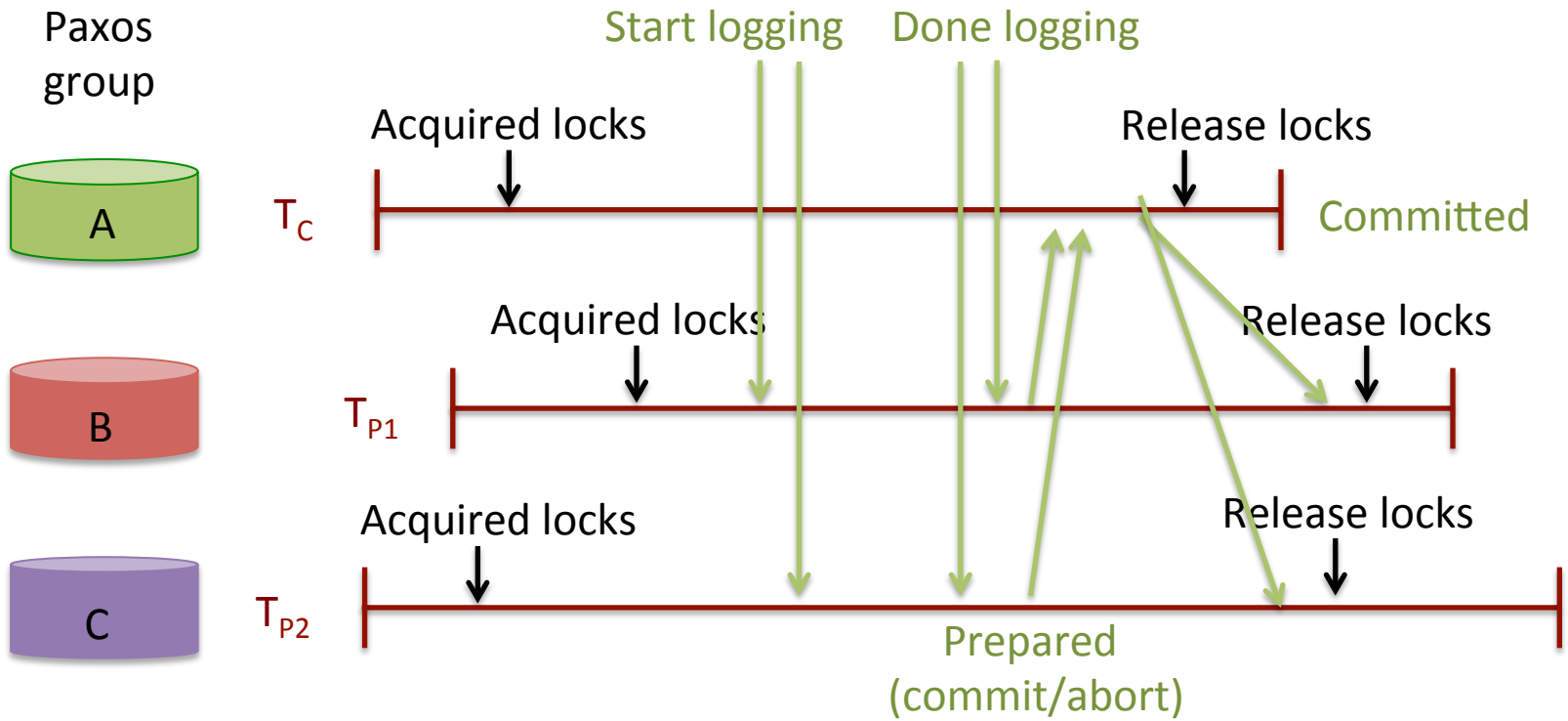
How to assign timestamp for a transaction using TrueTime



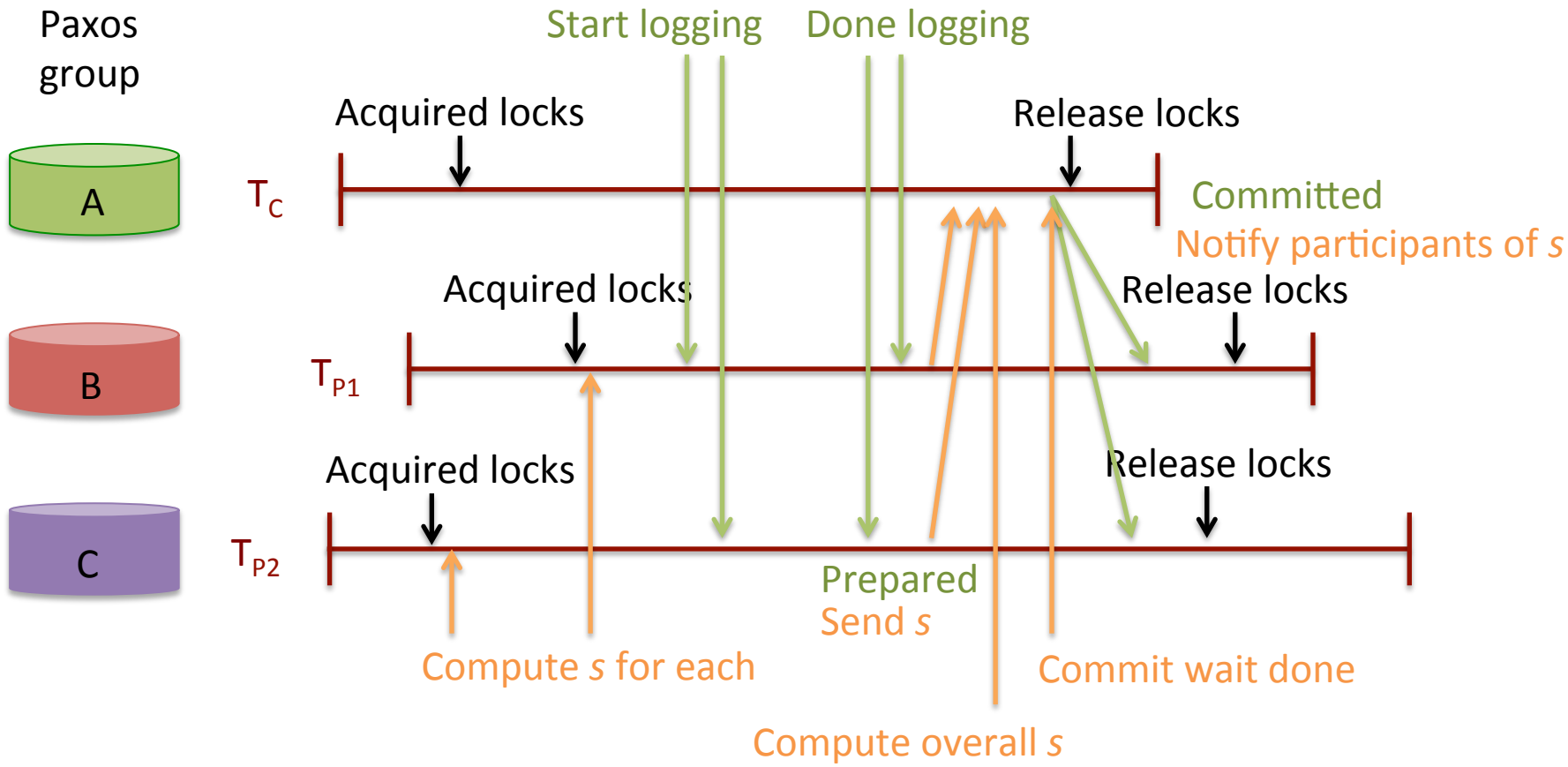
Commit Wait and Replication



Distributed transactions across Paxos groups

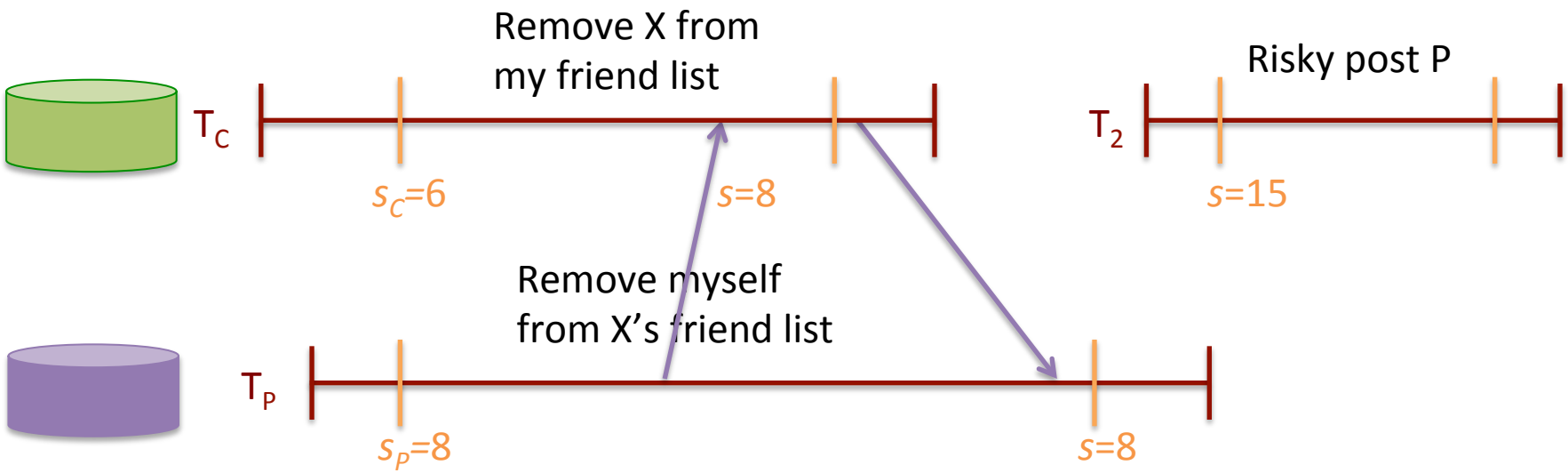


Commit Wait and 2-Phase Commit



(greater than TT.now().latest &&
Any other transaction TS assigned by co-ordinator)

Example



Time	<8	8	15
My friends	[X]	[]	
My posts			[P]
X's friends	[me]	[]	

Conclusions

- Reify clock uncertainty in time APIs
  - Known unknowns are better than unknown unknowns
  - Rethink algorithms to make use of uncertainty
- Stronger semantics are achievable
  - Greater scale != weaker semantics

Thanks!

http://homepages.inf.ed.ac.uk/pbhatoti/