Dynamic Scalable State Machine Replication

Long Hoang Le

Eduardo Bezerra

Fernando Pedone

May 13, 2016

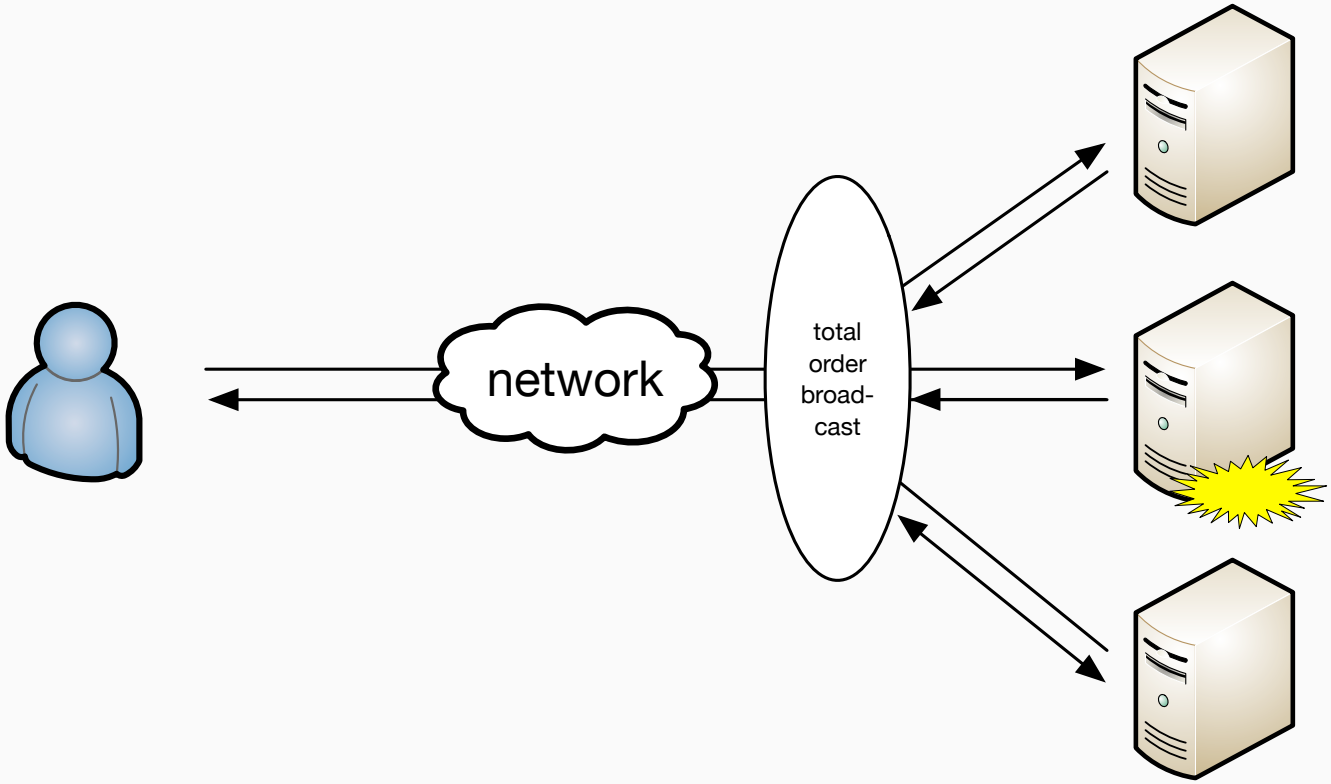
Universit della Svizzera italiana

Outlines

1. Motivation
2. Dynamic Scalable State Machine Replication
3. Implementation & Evaluation
4. Conclusion

Motivation

The need of availability



State Machine Replication (SMR)

Providing fault-tolerance, strong consistency

- All replicas start in the same initial state
- Apply same set of commands in the same order
- The executions of commands are deterministic
- Proceed through the same set of states

State Machine Replication

Production systems

- Keep critical managements service online
  - Google's Chubby, Apache Zookeeper
- Persistent storage in distributed databases
  - Google Spanner, Windows azure storage, CockroachDB, Apache H-Store

Scaling State Machine Replication

SMR lacks of scalability:

- Every replica executes all commands.
- Adding servers does not increase the maximum throughput

Different techniques have been developed to deal with these limitations

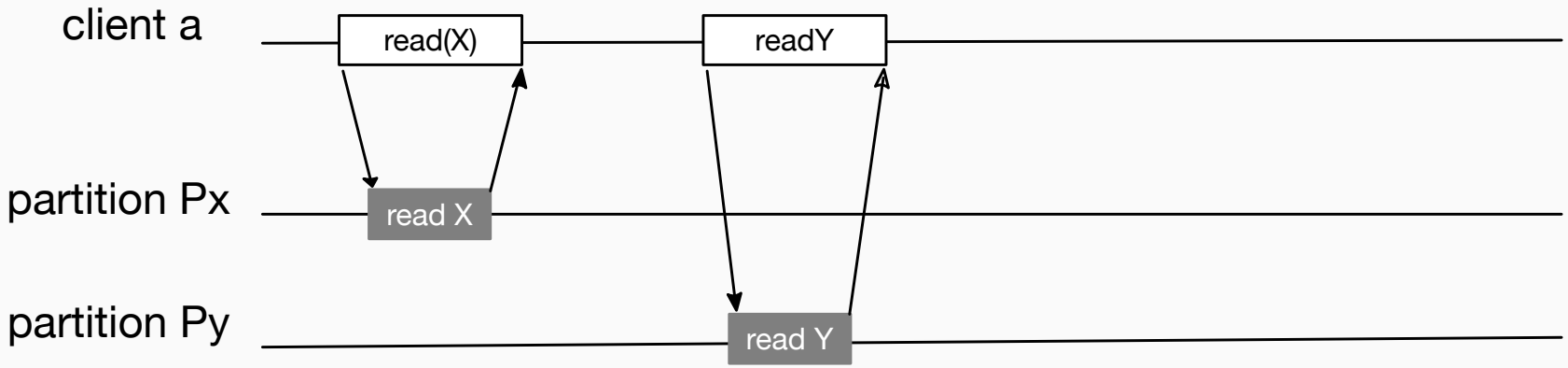
- Scaling up
- Scaling out

Scalable State Machine Replication (SSMR)

Partitions application's state and replicates each partition

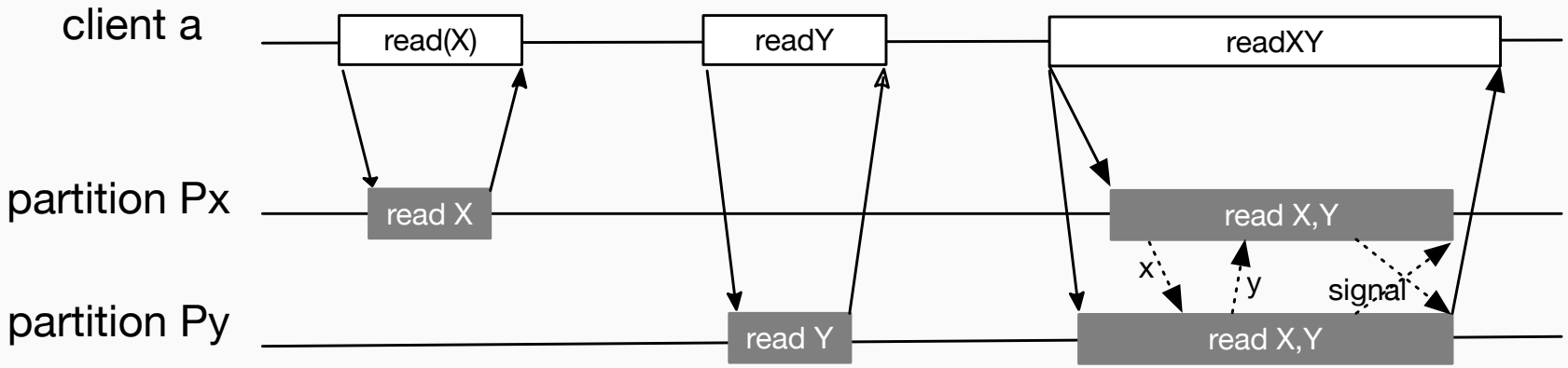
Scalable State Machine Replication (SSMR)

Partitions application's state and replicates each partition



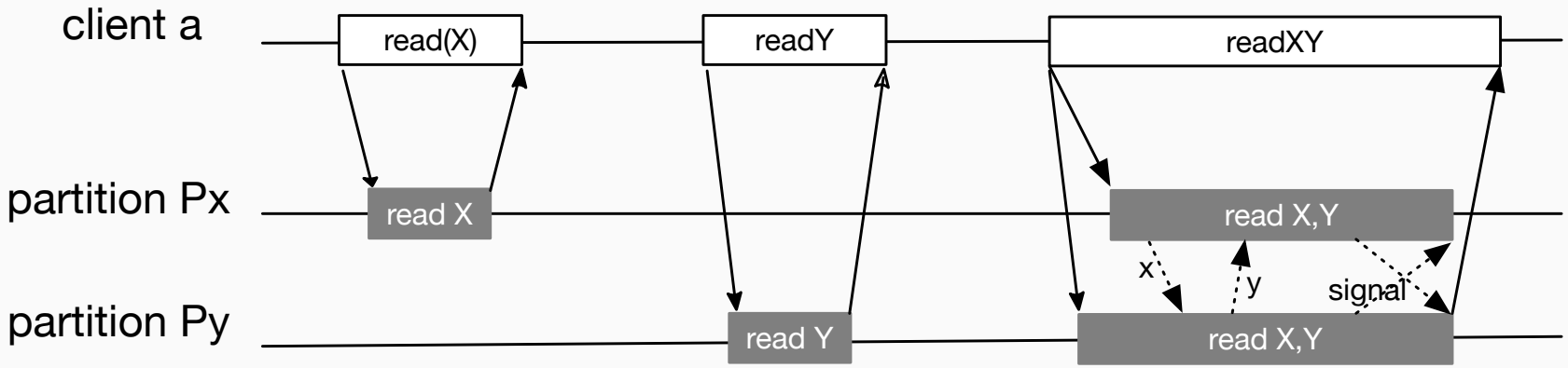
Scalable State Machine Replication (SSMR)

Partitions application's state and replicates each partition



Scalable State Machine Replication (SSMR)

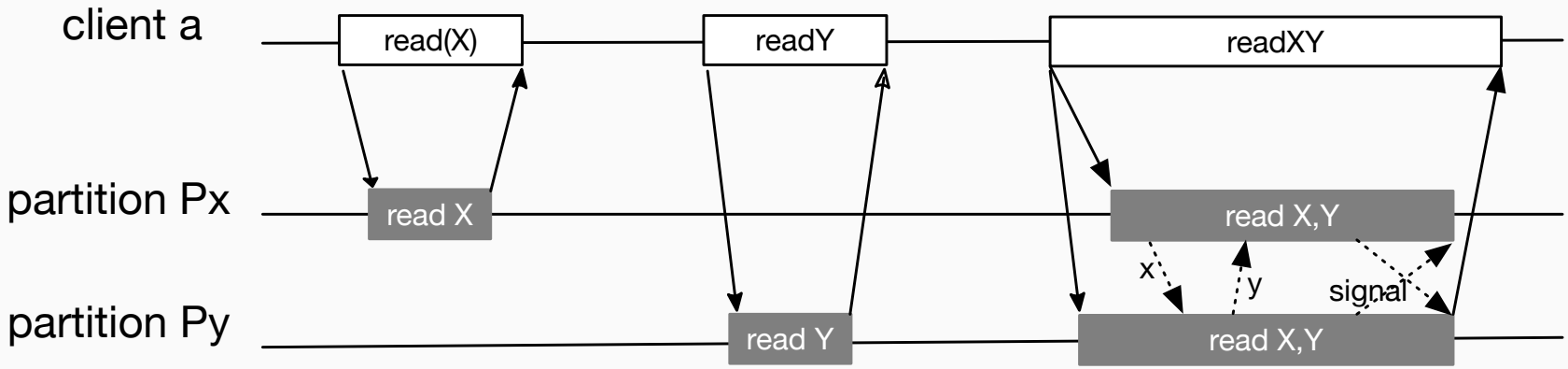
Partitions application's state and replicates each partition



Guarantees strong consistency (i.e., linearizability)

Scalable State Machine Replication (SSMR)

Partitions application's state and replicates each partition

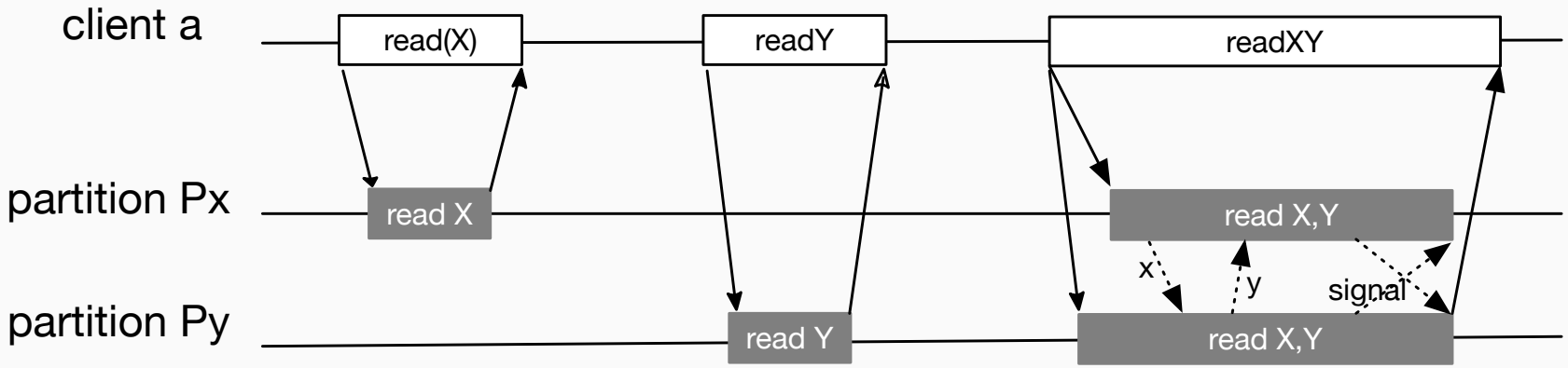


Guarantees strong consistency (i.e., linearizability)

Expensive cost of multi-partition commands

Scalable State Machine Replication (SSMR)

Partitions application's state and replicates each partition



Guarantees strong consistency (i.e., linearizability)

Expensive cost of multi-partition commands

Assumes a static workload partitioning

Dynamic Scalable State Machine Replication

System Model

Crash-stop failure model

- No Byzantine behavior

Communication by message passing

- One-to-one communication use reliable multicast
- One-to-many communication relies on atomic multicast
- Messages can be lost, reordered, but not corrupted

System is partially synchronous

- No delay bound

Consistency level: linearizability

General idea

Dynamically changing the state partitioning by exploiting locality

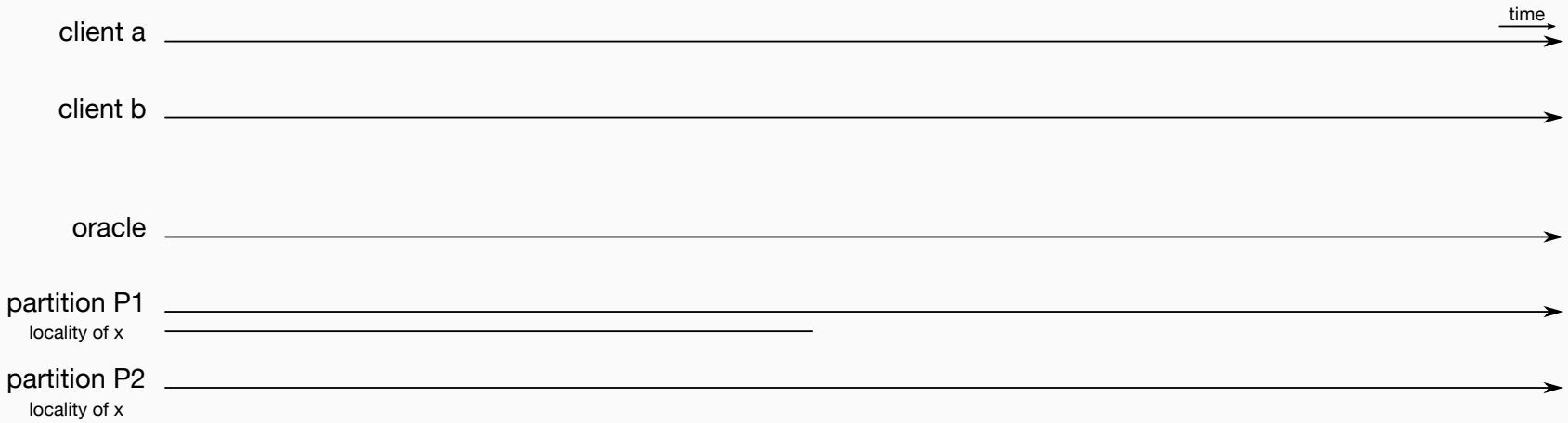
- Involved variables are moved to a single partition
- Command is executed against this partition
- Supported commands: *consult*, *create*, *access*, *move*, *delete*

Variables that are usually accessed together are moved to the same partition

Variable mapping managed by an Oracle partition and client cache

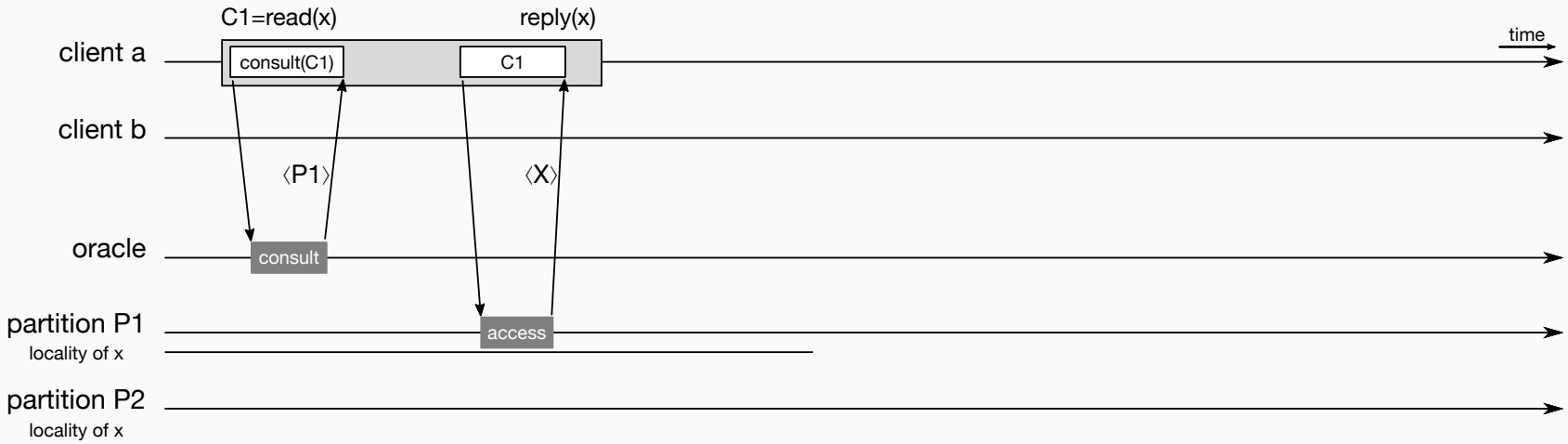
DS-SMR falls back to S-SMR execution after few unsuccessful attempts.

General idea

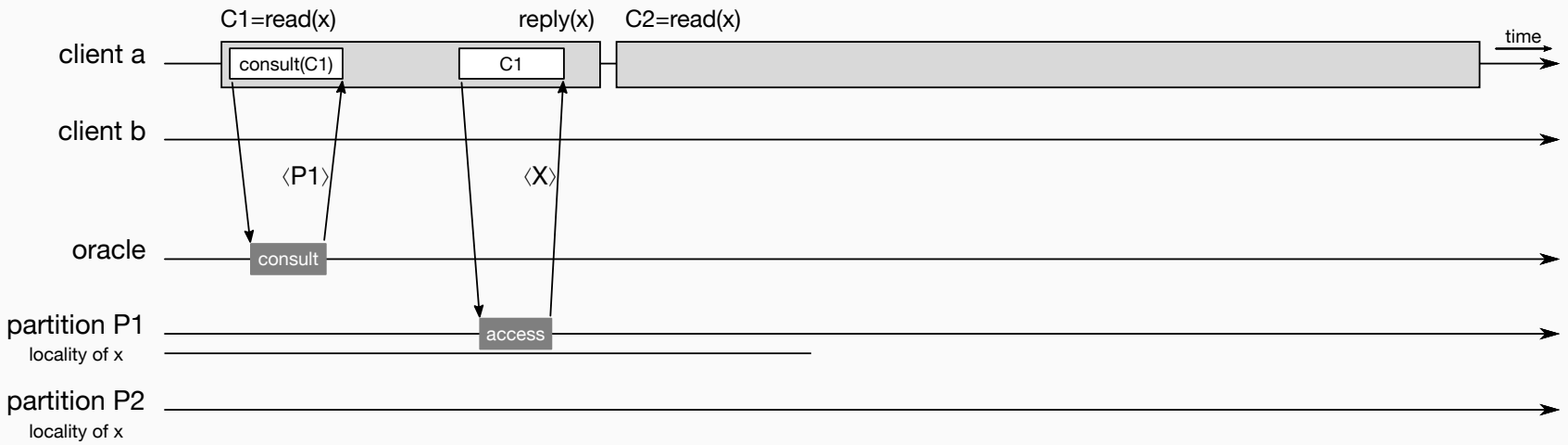


General idea

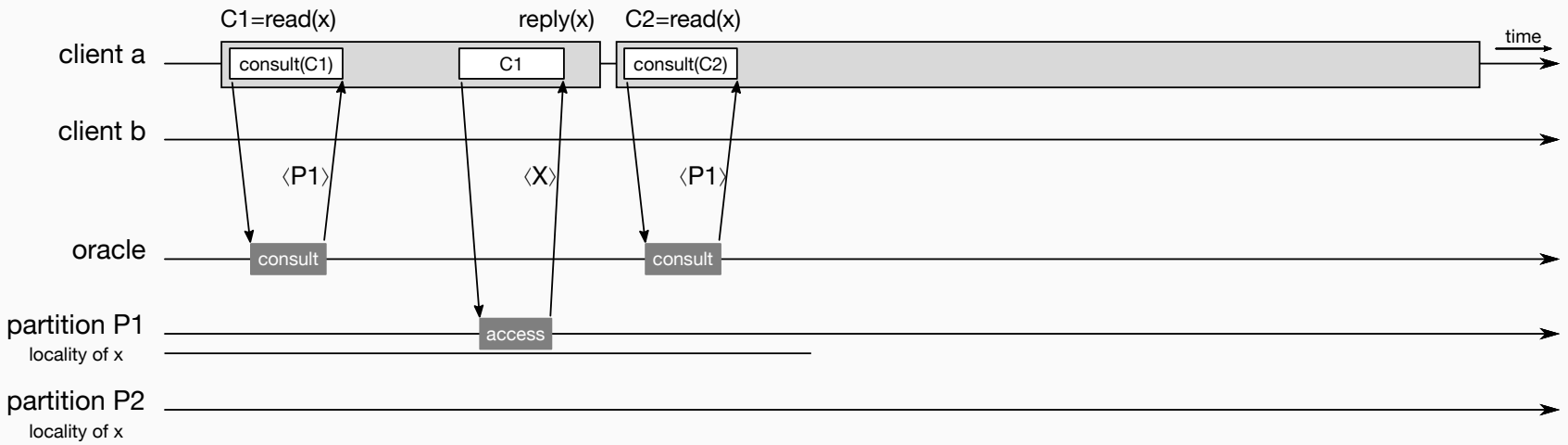
General idea



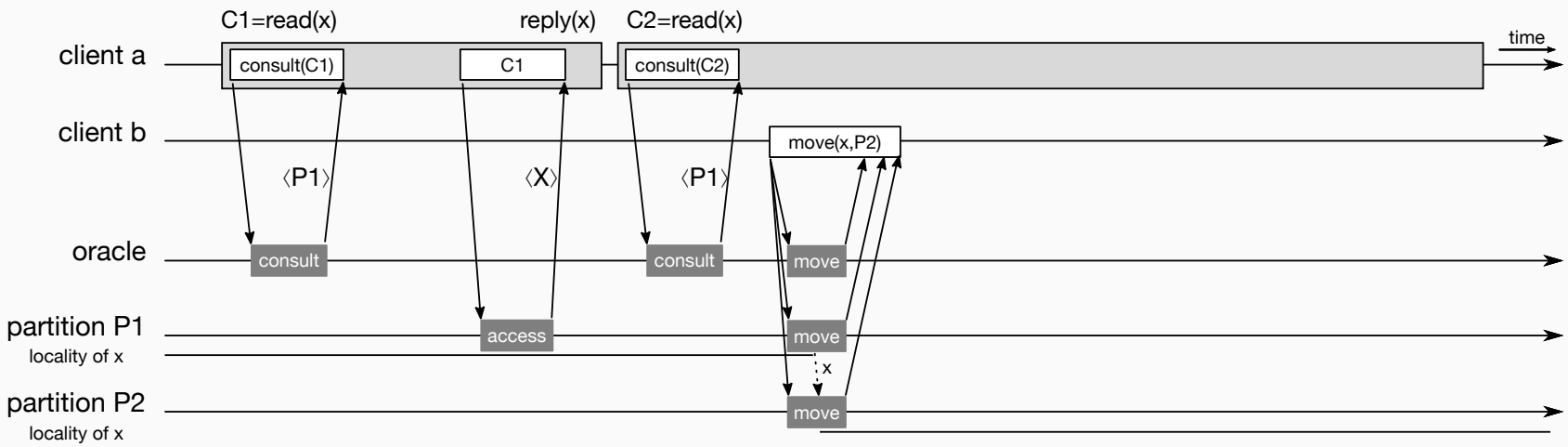
General idea



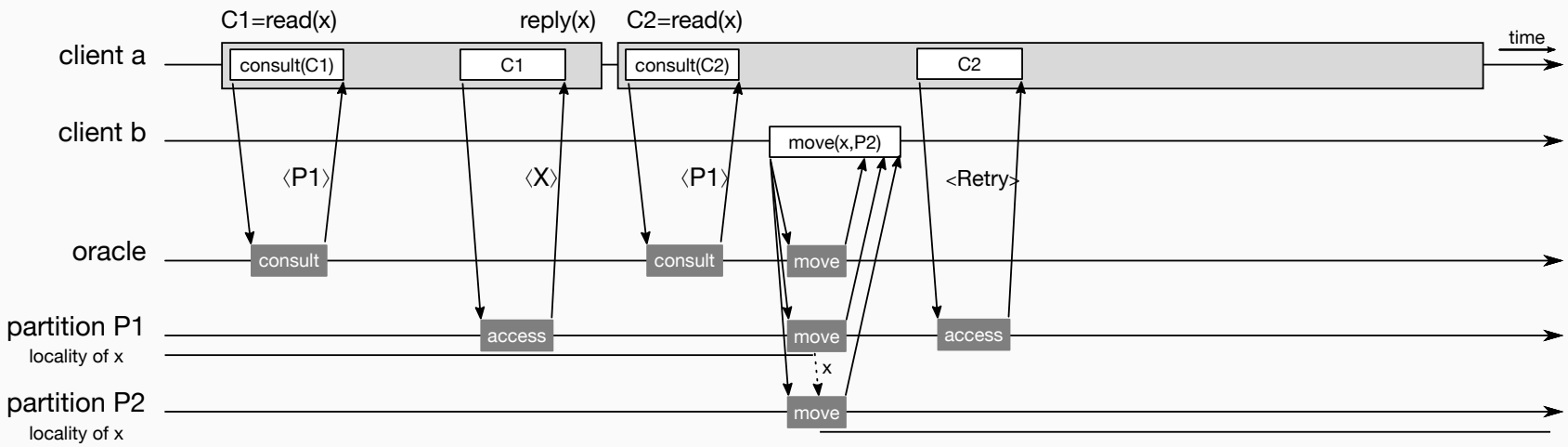
General idea



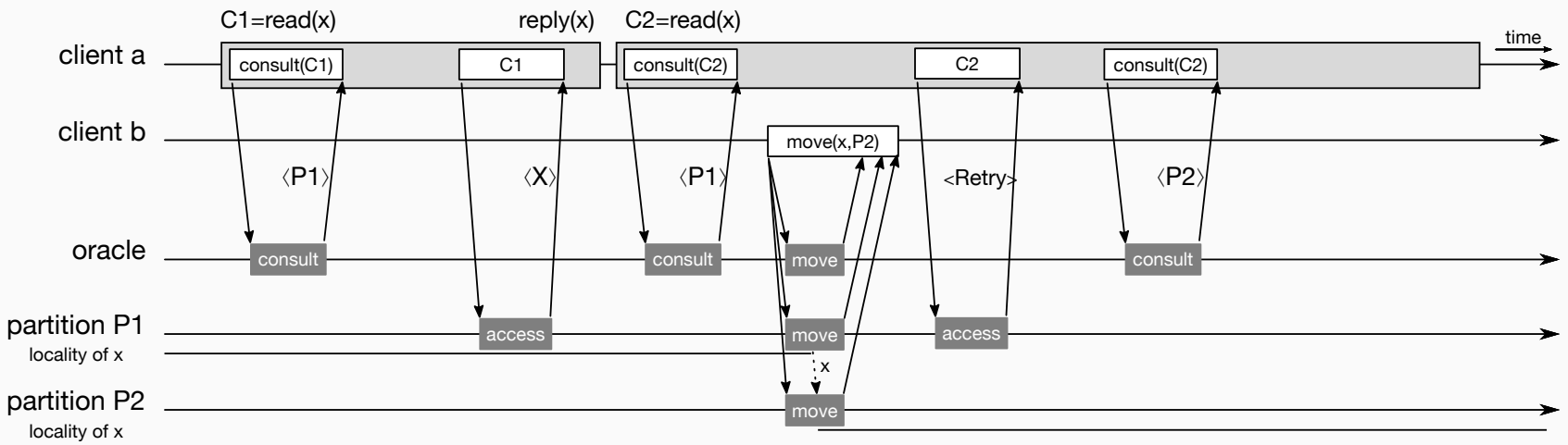
General idea



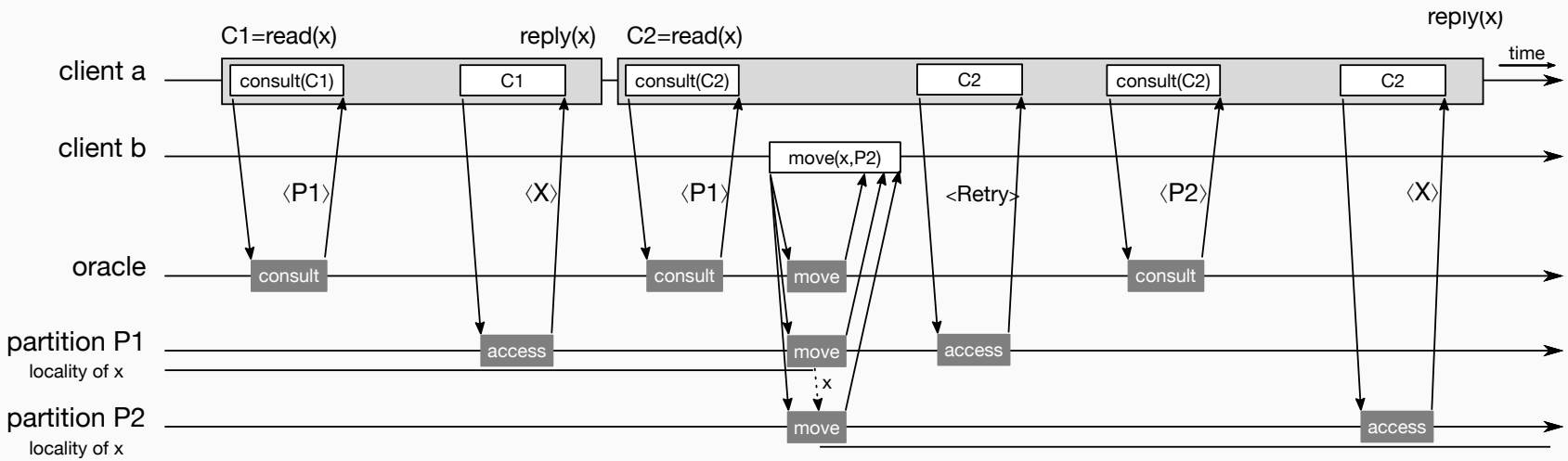
General idea



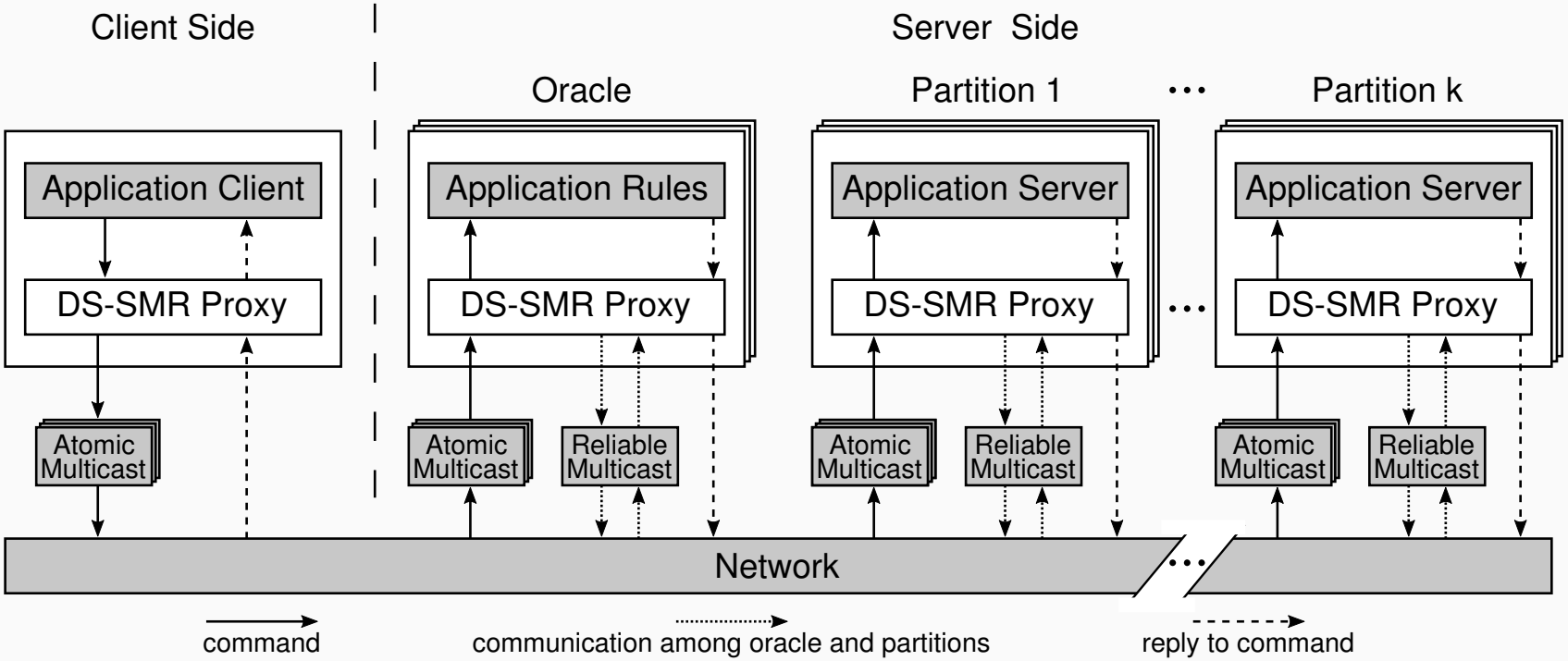
General idea



General idea

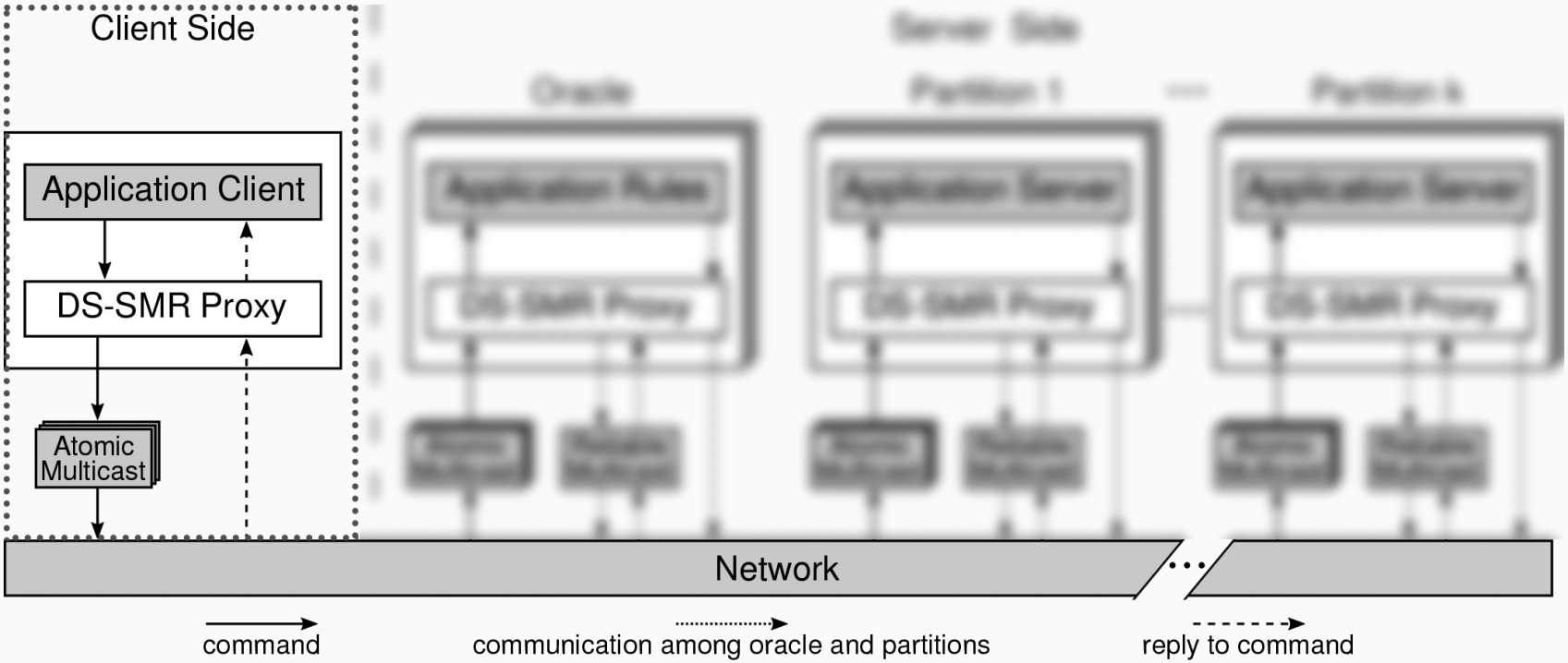


Architecture



- Clients atomically multicast commands to oracle and partitions
- Oracle and partitions exchange messages through reliable multicast

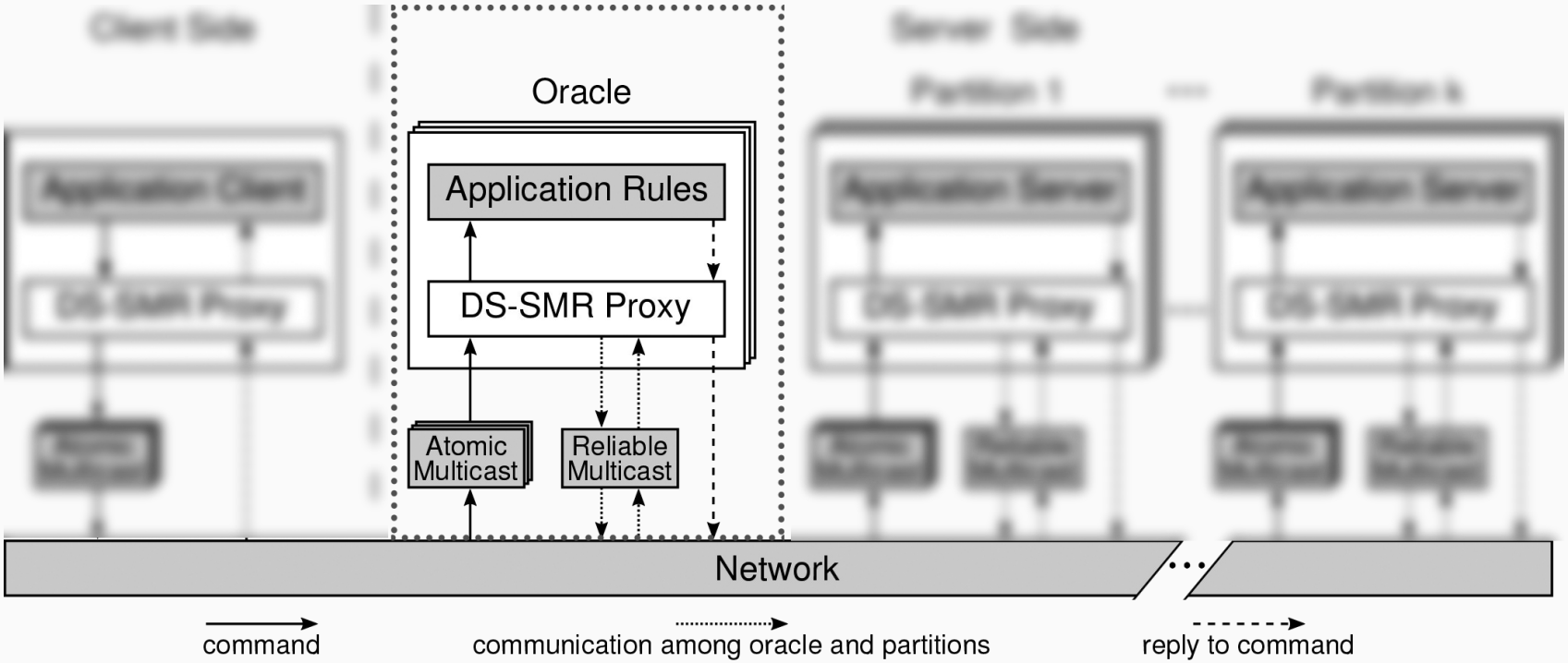
Architecture



Client

- Application Client
- DS-SMR Client Proxy

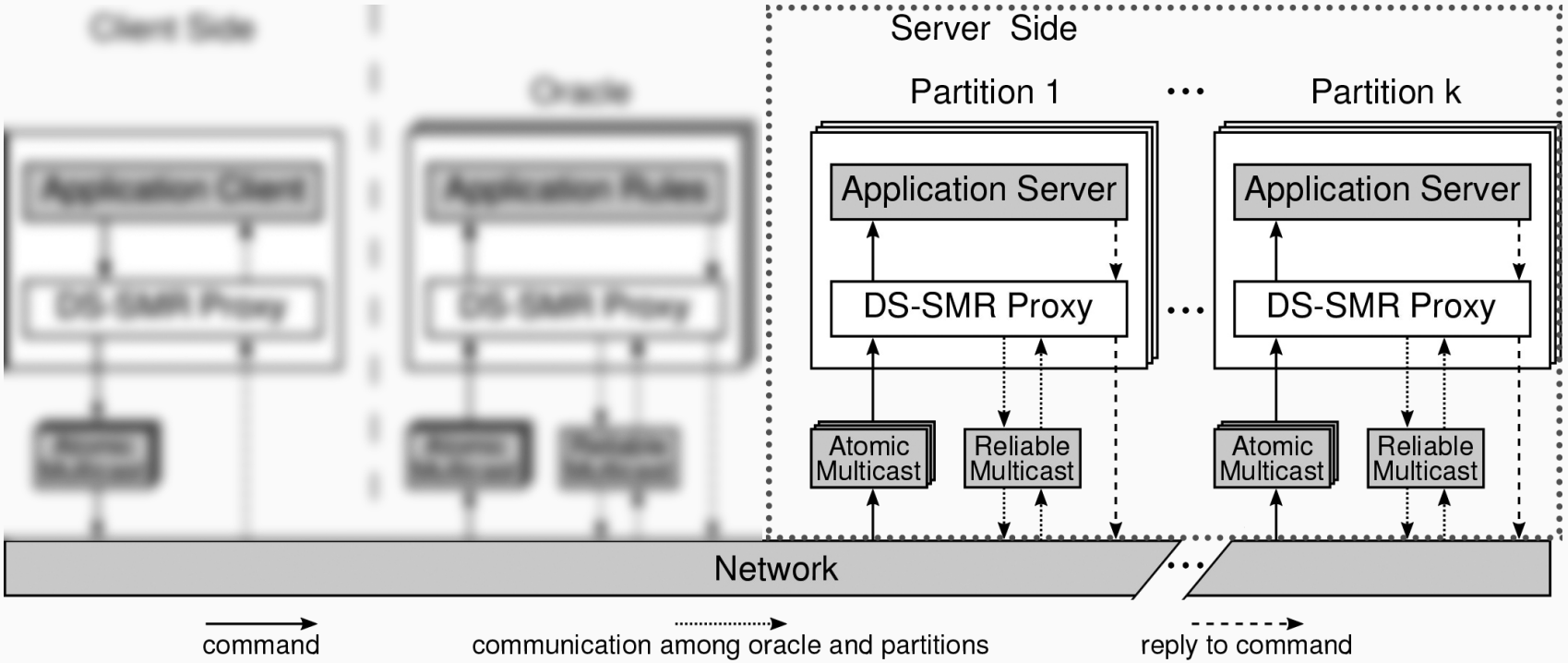
Architecture



Oracle

- Application Oracle
- DS-SMR Oracle Proxy

Architecture



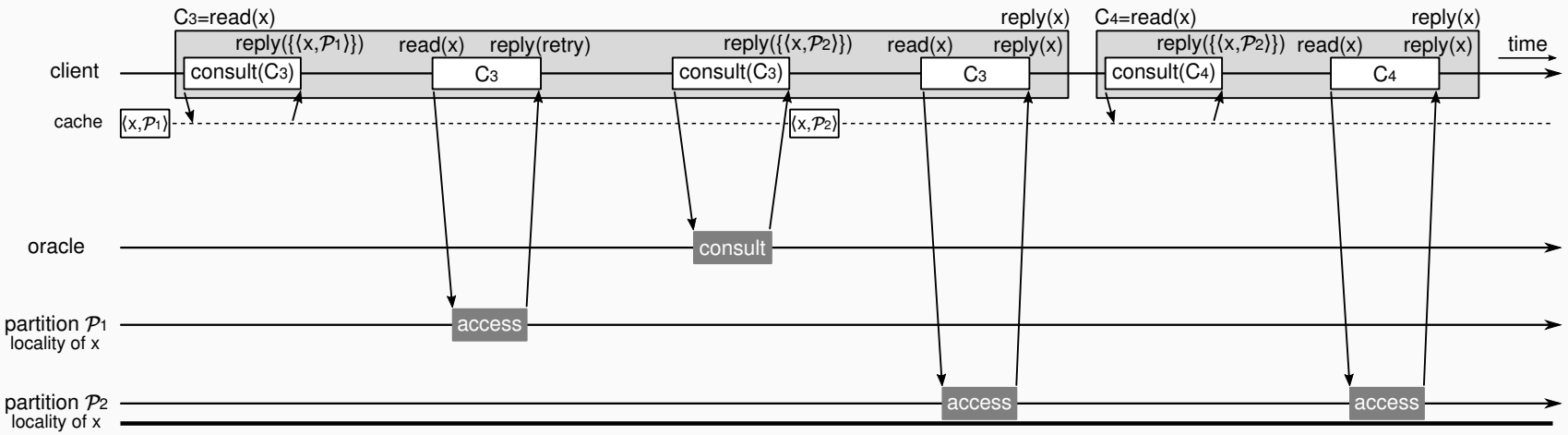
Server

- Application Server
- DS-SMR Server Proxy

Performance optimizations

- Caching
  - Each client proxy has a local cache
  - Client consults local cache to determine variable's location
  - When retrying command, clients update cache
- Client's cache could accurately resolve most query

Performance optimizations



Implementation & Evaluation

Eyrie

Eyrie simplifies implementing services based on DS-SMR

Provides proxy layers

Allows application designers to override default behaviors

- The PRObject class
- The Client class
- The StateMachine class
- The OracleStateMachine class

Chirper

Social network application similar to Twitter

Supports commands:

- post
- getTimeline
- follow, unfollow

State partitioning is based on users' interest

Environment setup and configuration parameters

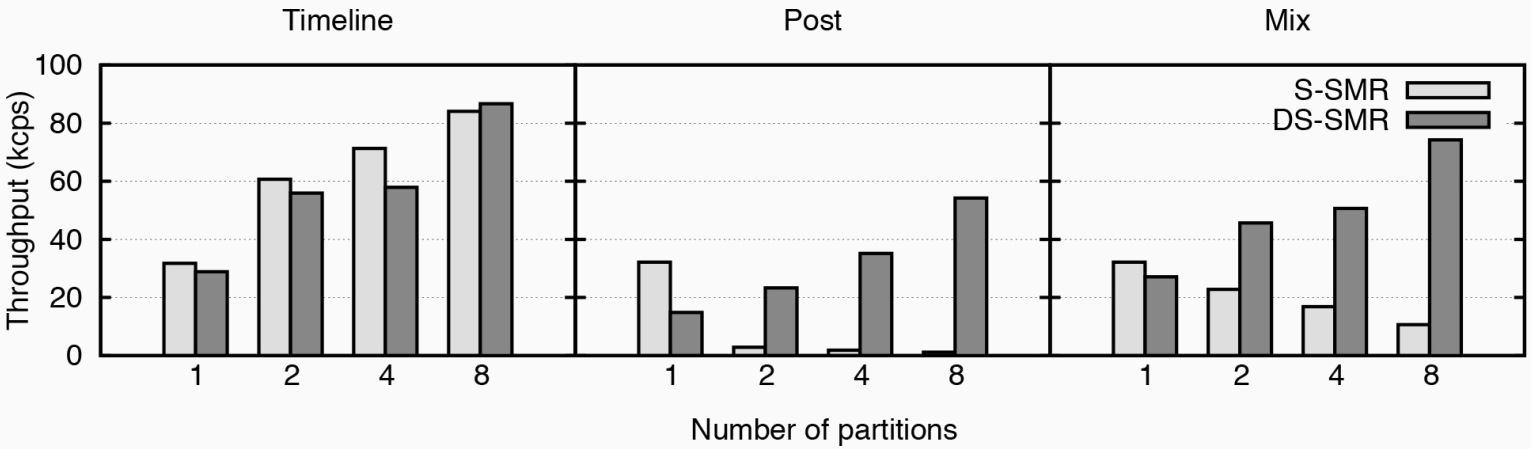
Running *Chirper* under different loads and partitionings

Number of partitions: 2, 4, and 8

Workloads:

- Timeline
- Post
- Follow/Unfollow
- Mix (Timeline: 85%, Post: 7.5%, Follow: 3.75%, Unfollow: 3.75%)

Results


Timeline
Post
Mix
Throughput (kcps)
100
80
60
40
20
0
S-SMR
DS-SMR
1
2
4
8
Number of partitions

Conclusion

Summary

- S-SMR does not adapt to high rate of global command
- DS-SMR introduces dynamic repartitioning to S-SMR
- Results show that D-SSMR outperforms S-SMR when there is access locality
- Eyrie makes developing services based on DS-SMR much simpler

Q&A