CRDT Sets: Paper to Product

(Or Everything You Always Wanted to Know About ORSets* (*But Were Afraid to Ask))

What?

- Why Riak?
- What is Riak?
- What's a CRDT, anyway?
- A replicated set

This project is funded by the European Union,
7th Research Framework Programme, ICT call 10,
grant agreement n°609551.

Why Riak?

Scale Up

$$$Big Iron
(still fails)

Scale Out

Commodity Servers
CDNs, App servers
Expertise

Fault Tolerance

Low Latency

Low Latency

Amazon found every 100ms of latency cost them 1% in sales.

Low Latency

Google found an extra 0.5 seconds in search page generation time dropped traffic by 20%.

Trade Off

CAP

http://aphyr.com/posts/288-the-network-is-reliable

C A

C A

C

A

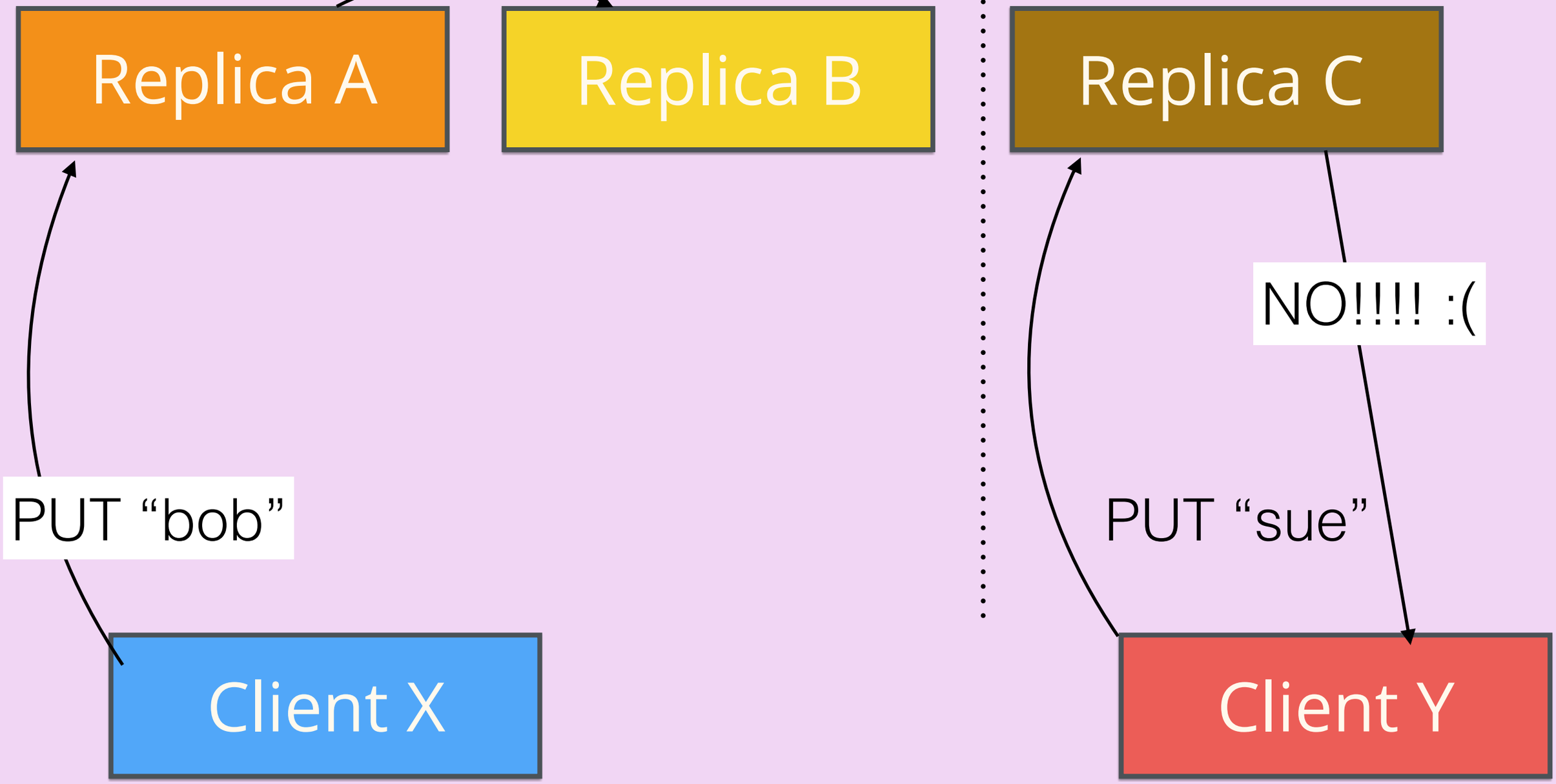
CP

AP

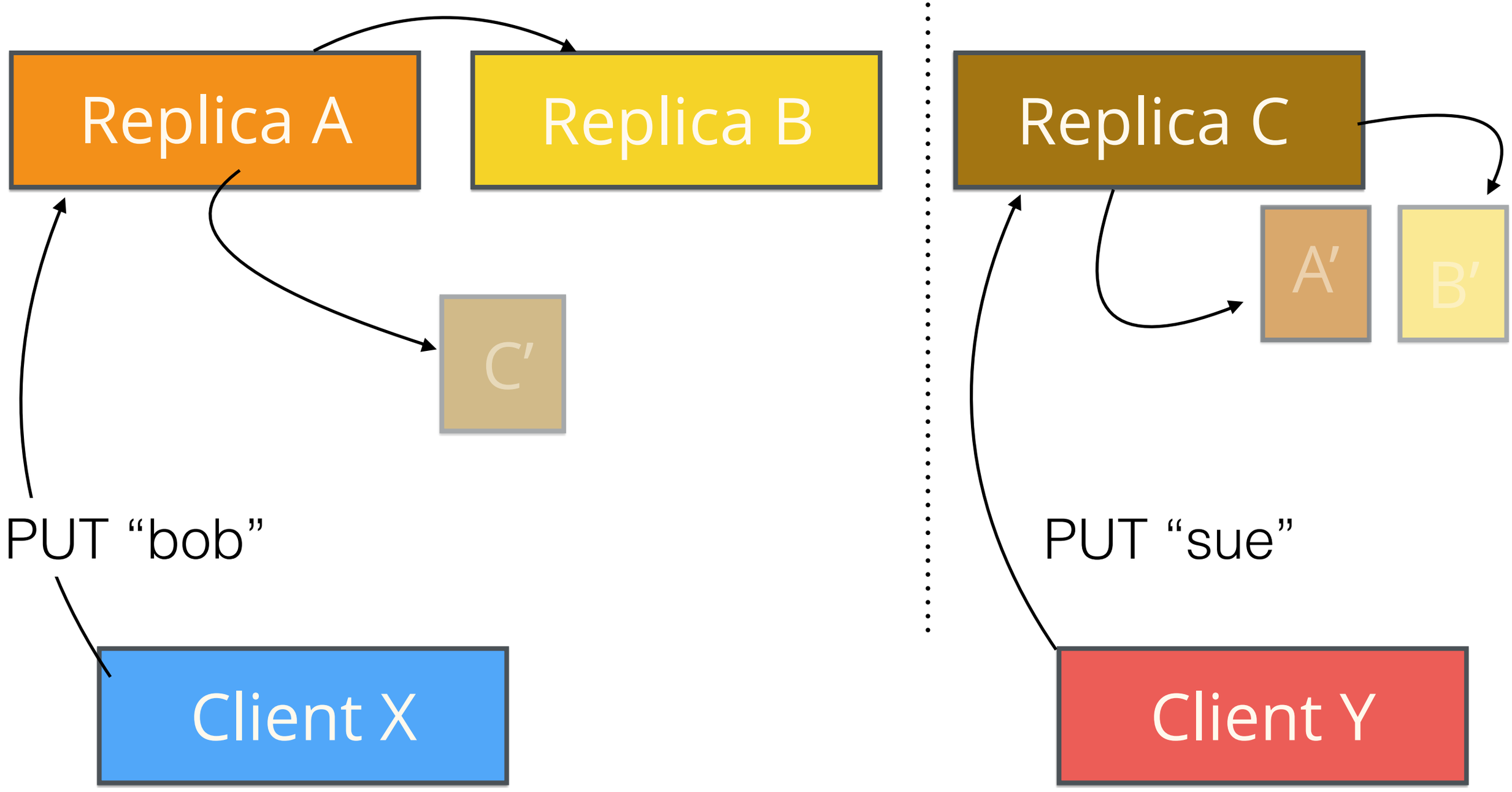
Eventual Consistency

Eventual consistency is a consistency model used in distributed computing that informally guarantees that, if no new updates are made to a given data item, eventually all accesses to that item will return the last updated value.

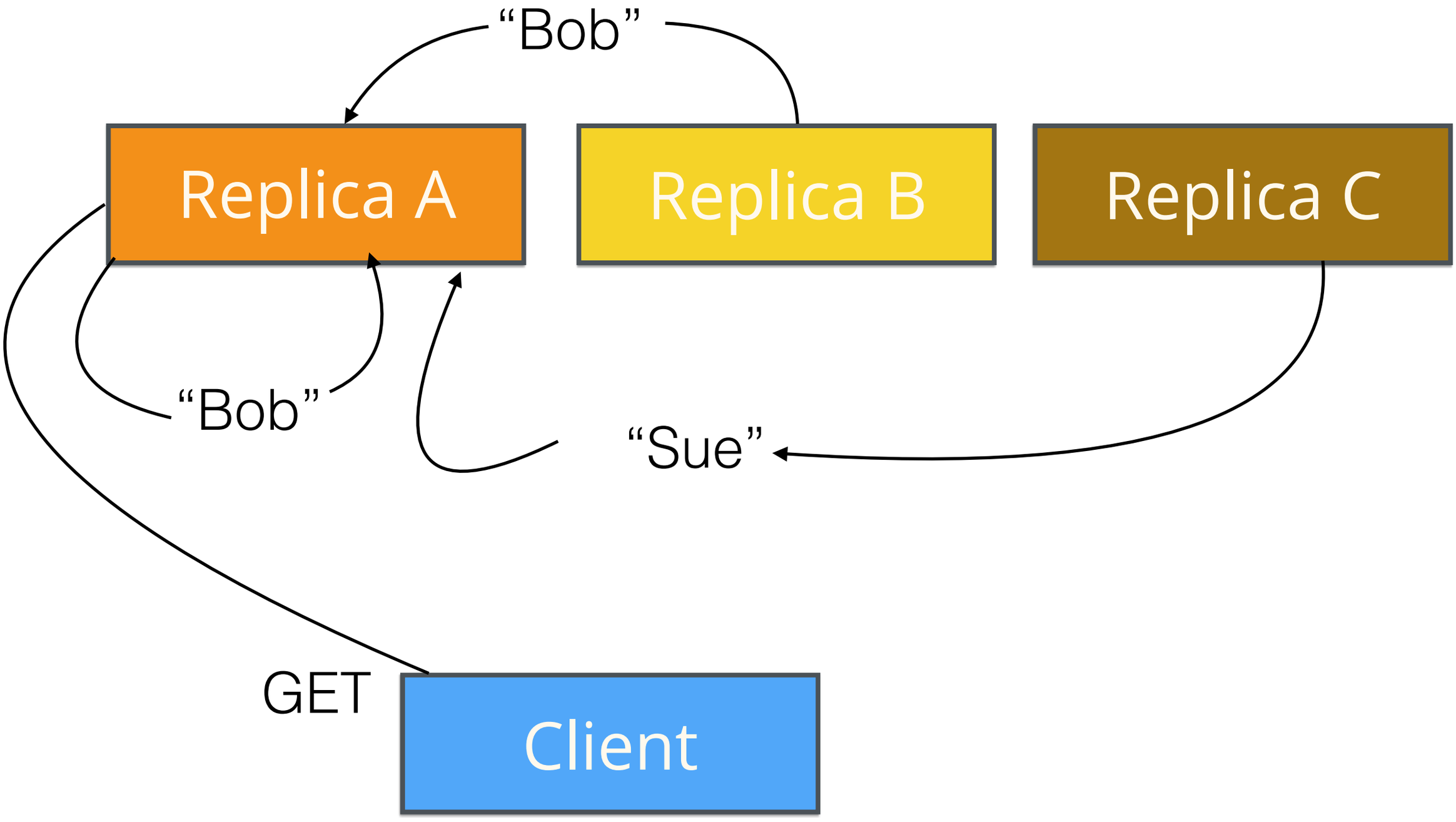
--Wikipedia


Replica A
Replica B
Replica C
NO!!!! :(
PUT “bob”
PUT “sue”
Client X
Client Y

CP


Replica A
Replica B
C′
PUT “bob”
Client X
Replica C
A′
B′
PUT “sue”
Client Y

AP

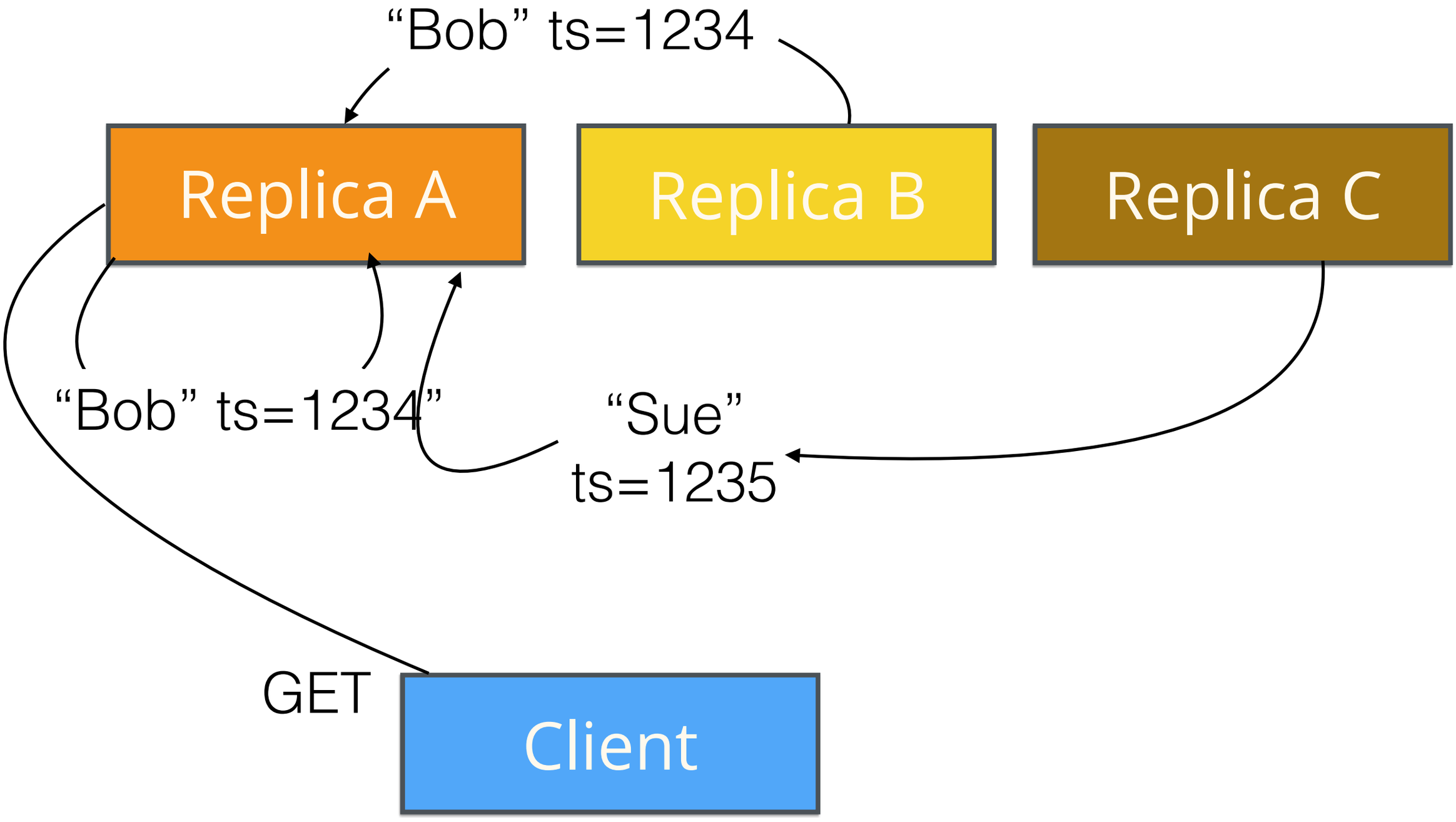


Conflict!

Google+

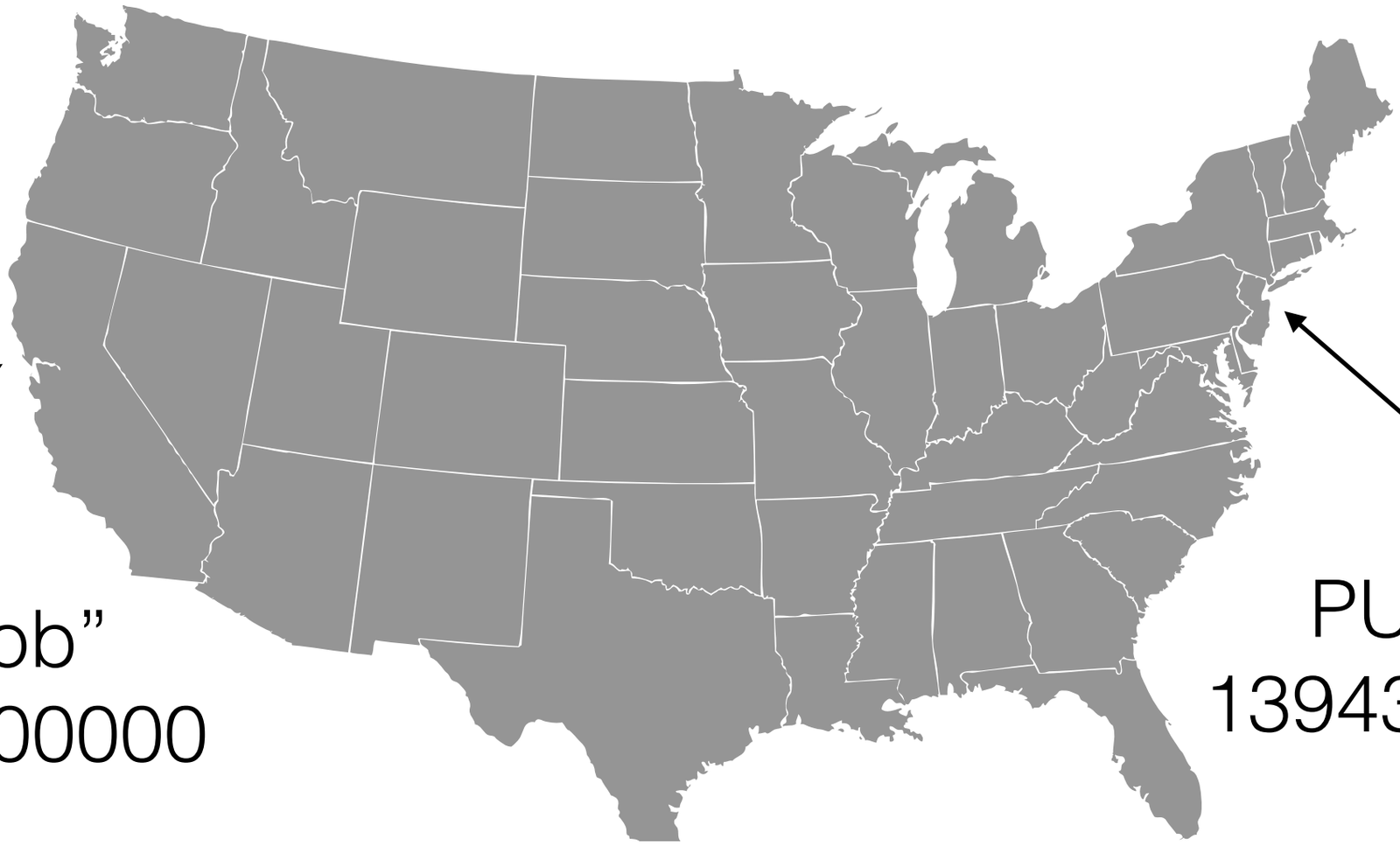
```
if (result.hasConflicts()) {
    // TODO: What should we do???
```

48:22 Live



Last Write Wins!

Physics Problem



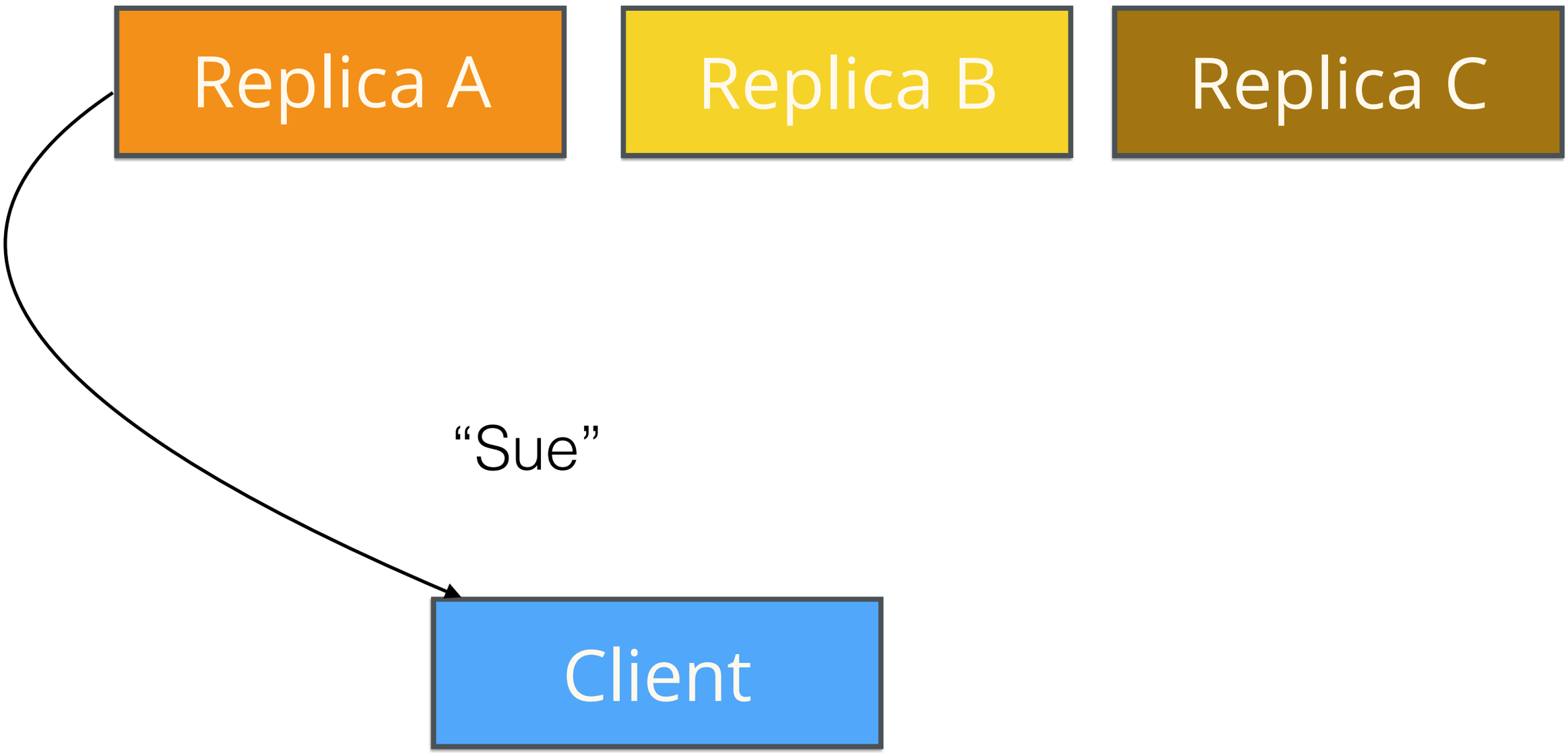
PUT “bob”
1394382600000

PUT “sue”
1394382600020

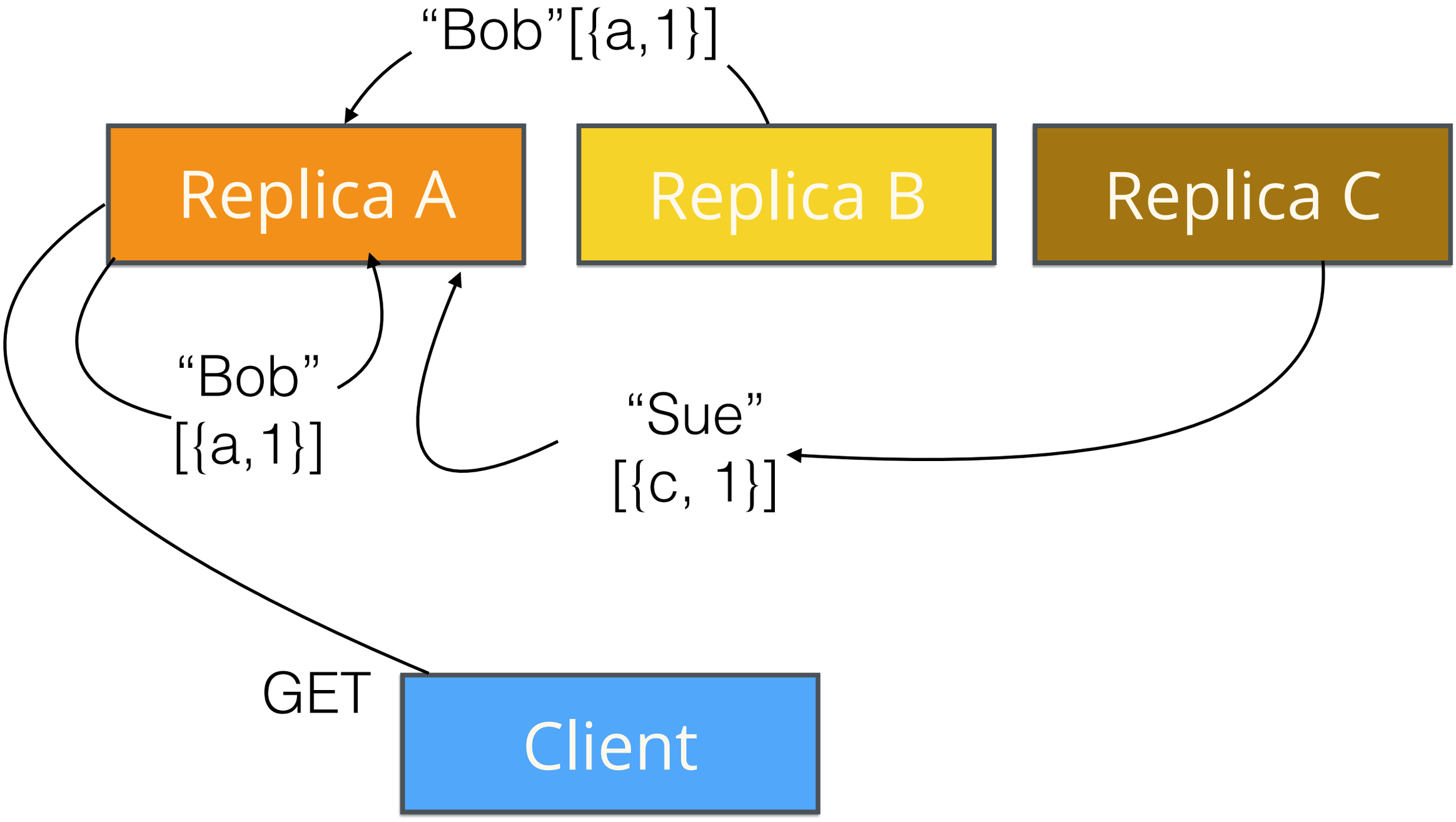
SF

NY

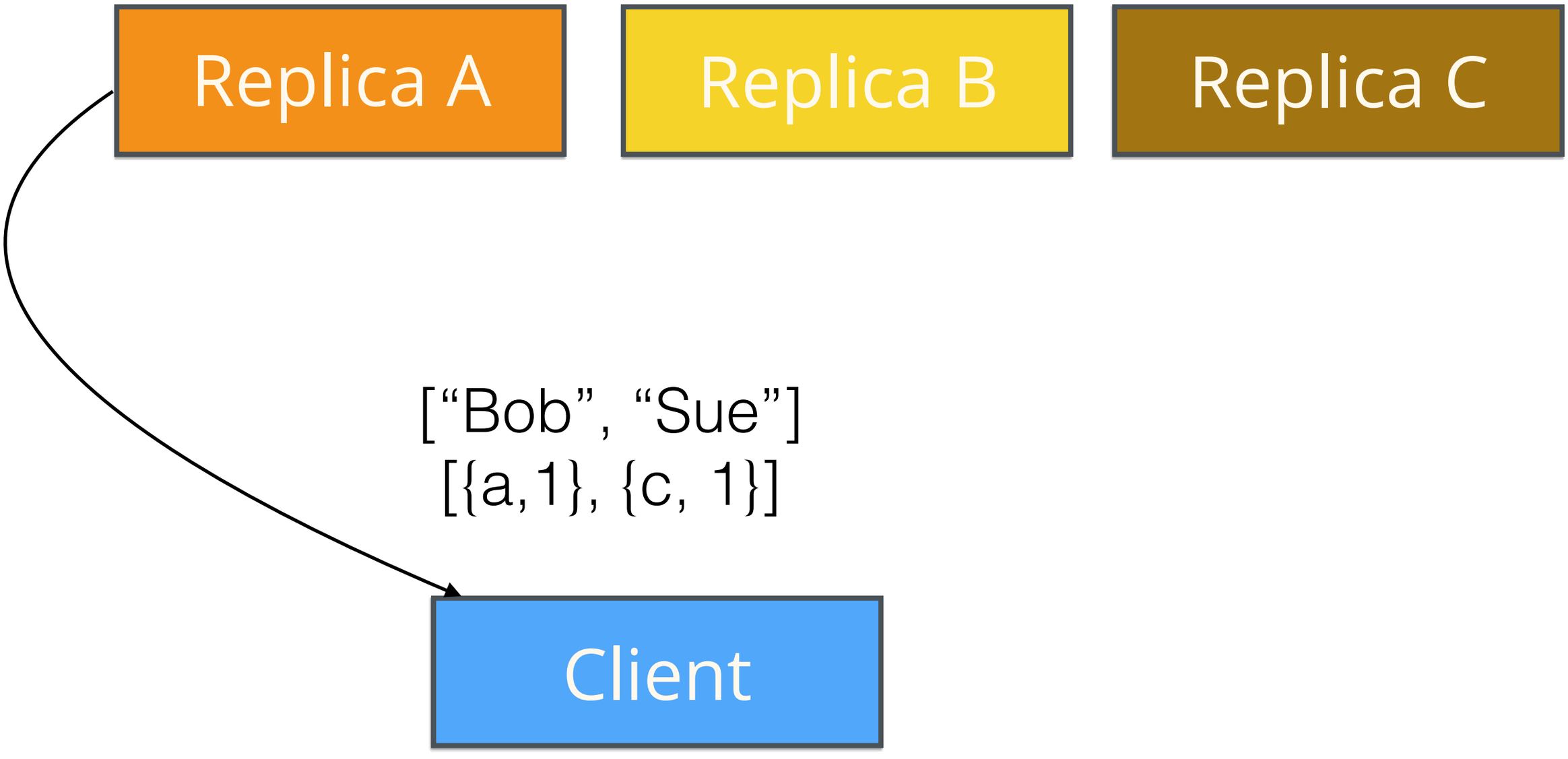
4,148 km
14 ms Light
21 ms fibre



Last Write Wins



Conflict!

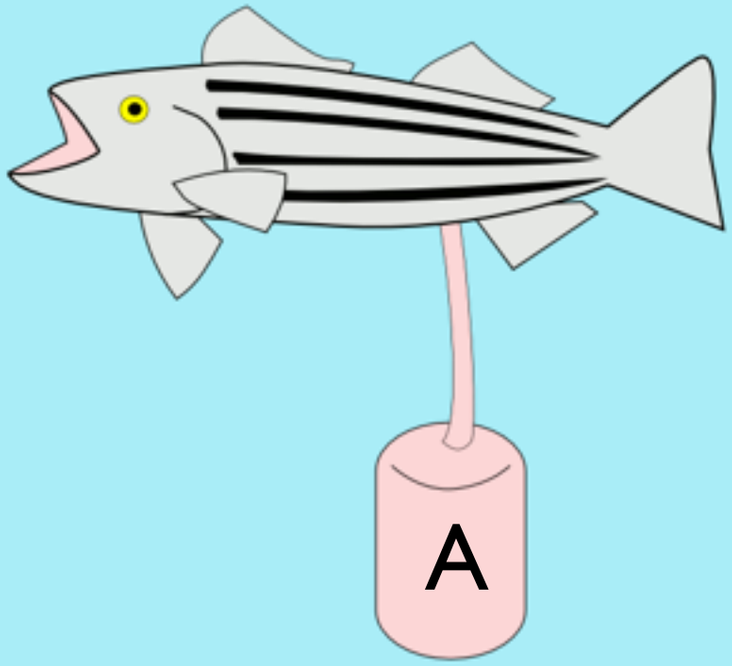


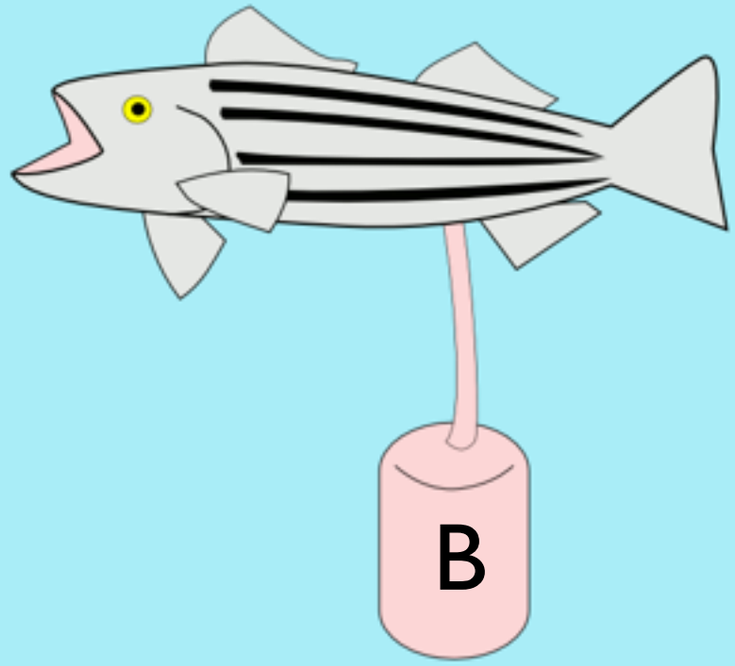
Multi-Value

Semantic Resolution

Dynamo

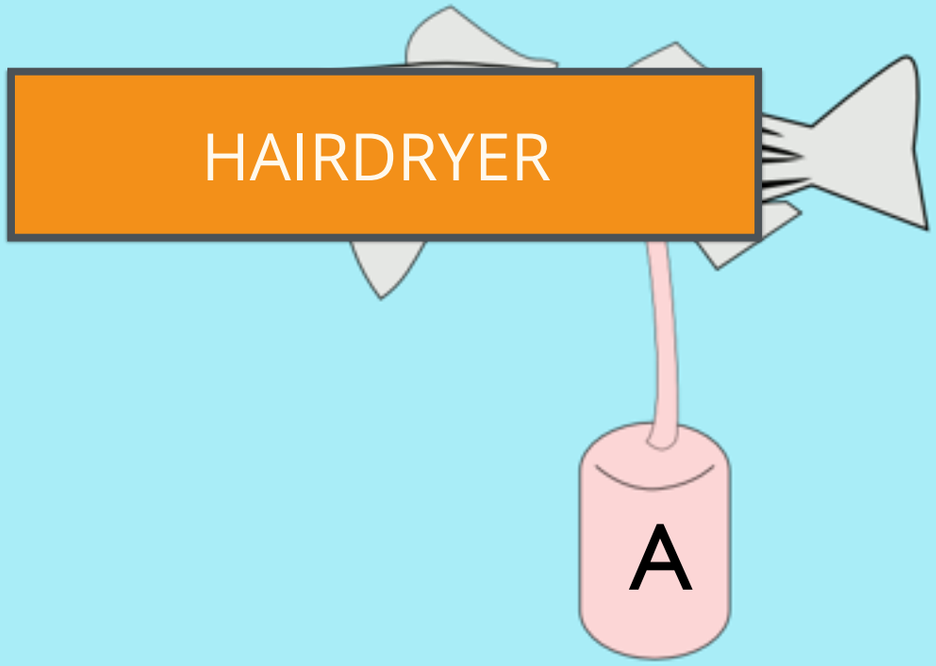
The Shopping Cart

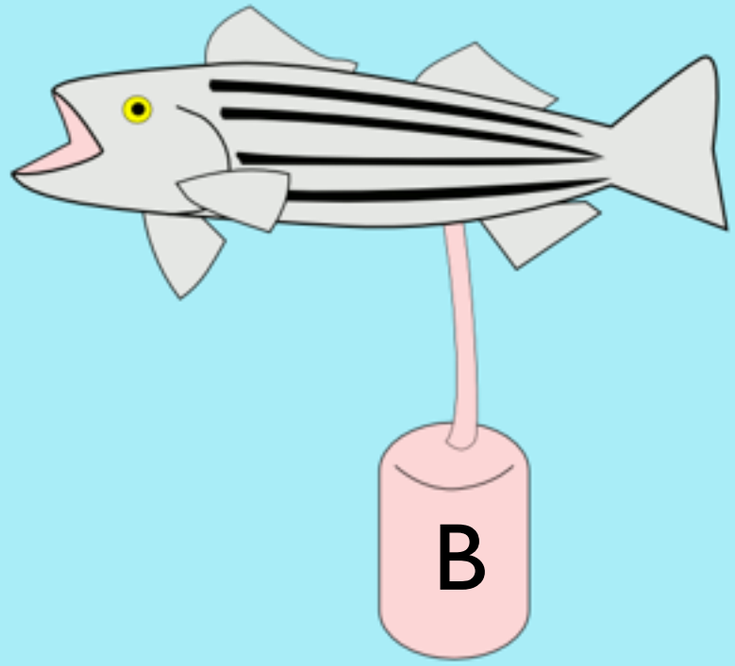




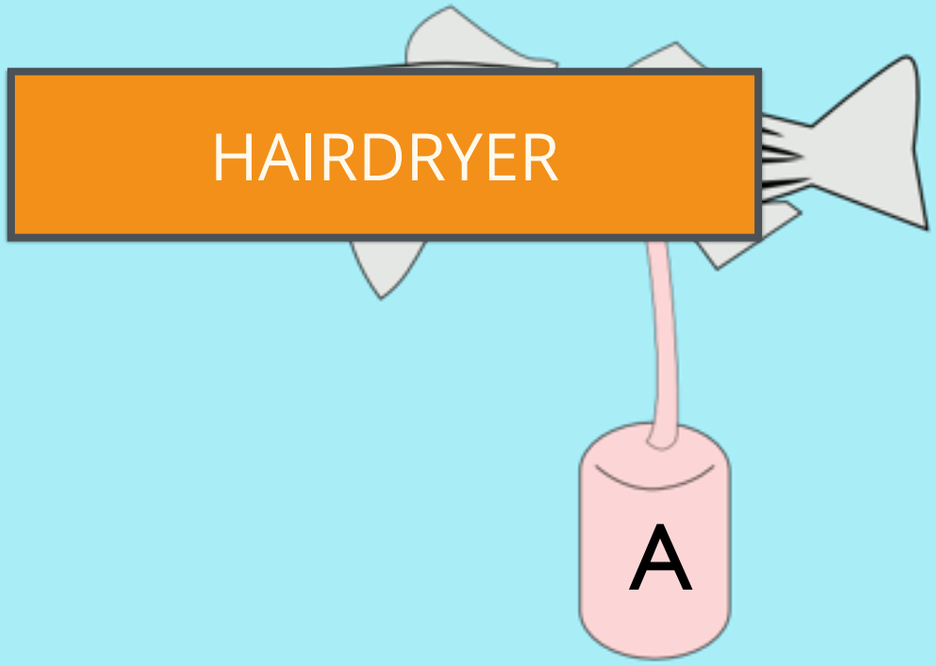
HAIRDRYER

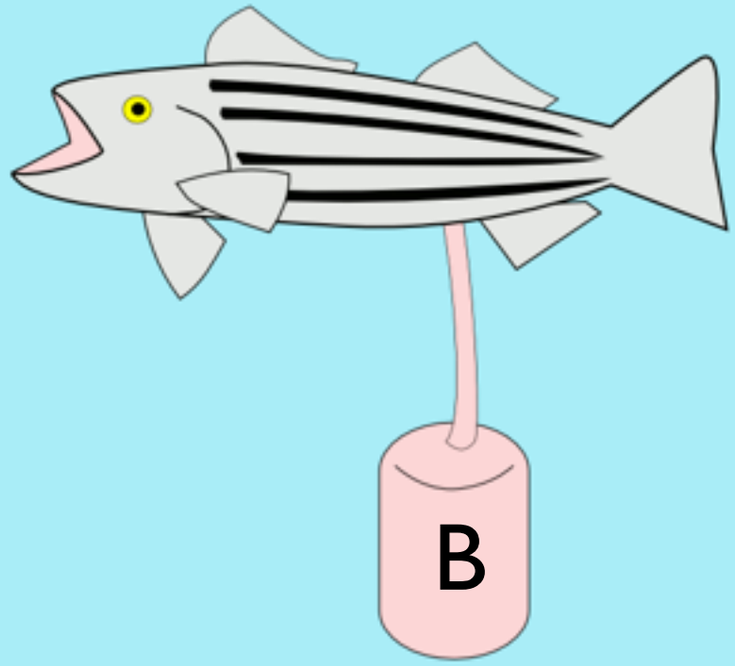
riak





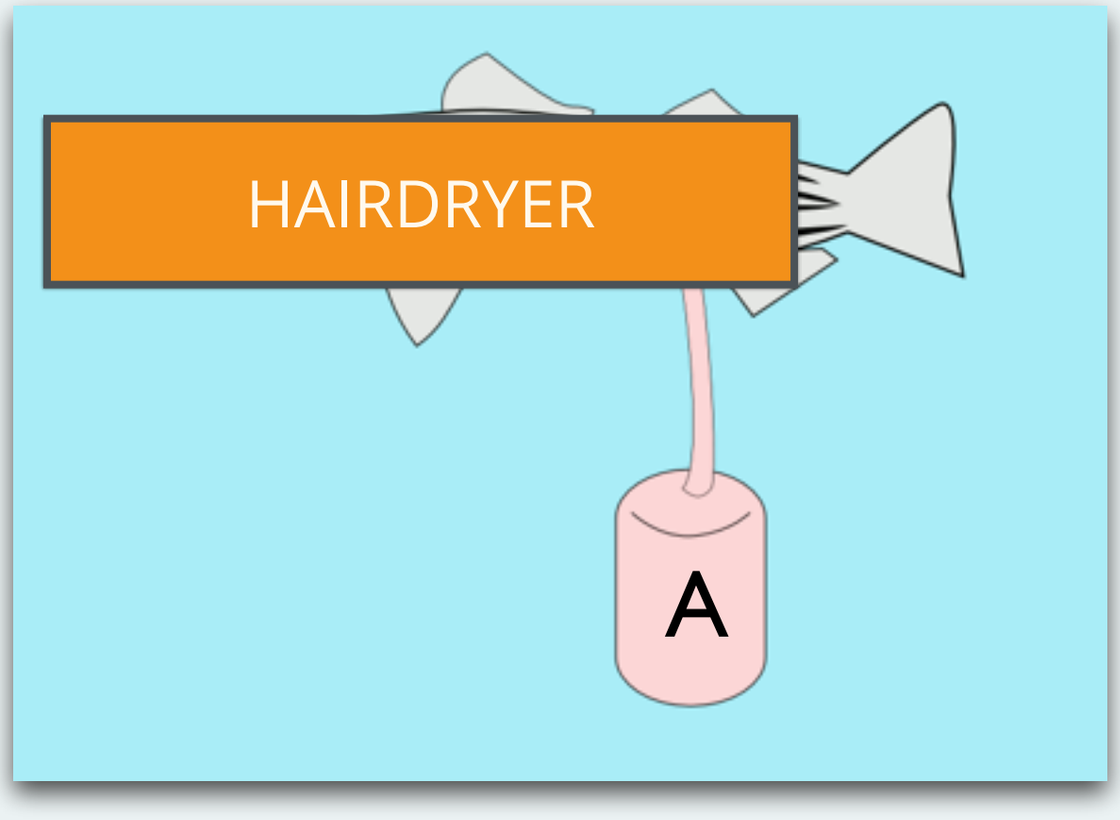
riak

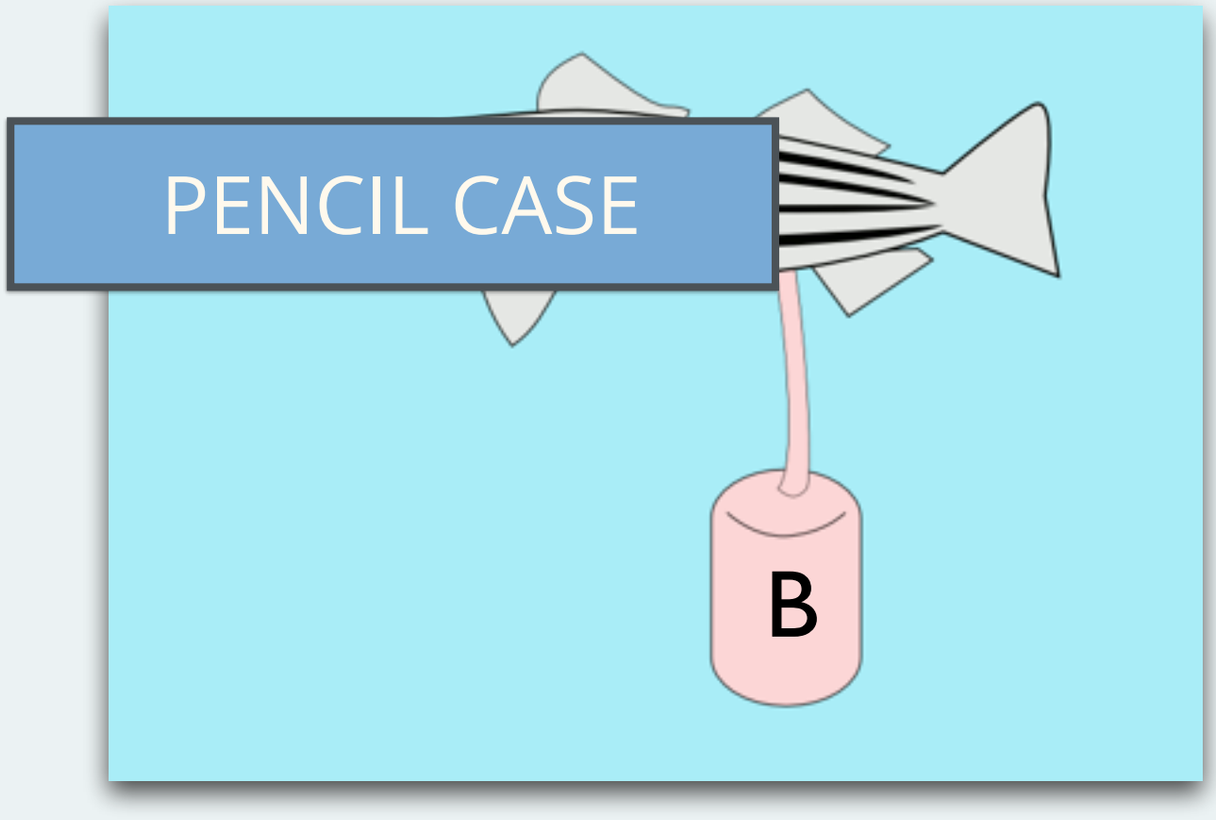




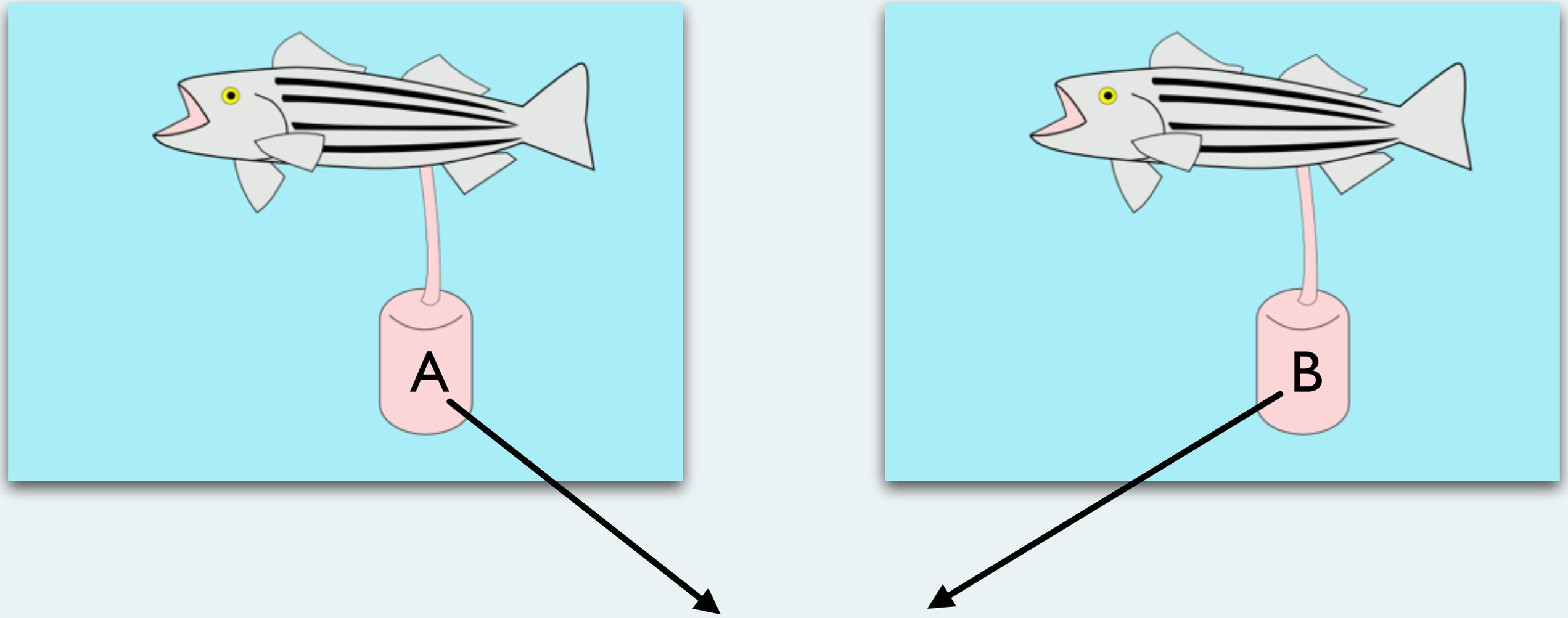
PENCIL CASE

riak





riak



[HAIRDRYER], [PENCIL CASE]

riak

Merge

Set Union of Values
Simples, right?

Removes?

Set Union?

“Anomaly” Reappear

Google F1

“We have a lot of experience with eventual consistency systems at Google.”

“We find developers spend a significant fraction of their time building extremely complex and error-prone mechanisms to cope with eventual consistency”

Google F1

“Designing applications to cope with concurrency anomalies in their data is very error-prone, time-consuming, and ultimately not worth the performance gains.”

Ad Hoc

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A comprehensive study of Convergent and Commutative Replicated Data Types

Marc Shapiro, INRIA & LIP6, Paris, France
Nuno Preguiça, CITI, Universidade Nova de Lisboa, Portugal
Carlos Baquero, Universidade do Minho, Portugal

Marek Zawirski, INRIA & UPMC, Paris, France

riak

Join Semi-lattice

Join Semi-lattice

Partially ordered set; Bottom; least upper bound

$$\langle\ S,\ \bot,\ \sqcup\ \rangle$$

riak

Join Semi-lattice

Associativity: (X⊔Y)⊔Z = X⊔(Y⊔Z)

Join Semi-lattice

Commutativity: X⊔Y = Y⊔X

Join Semi-lattice

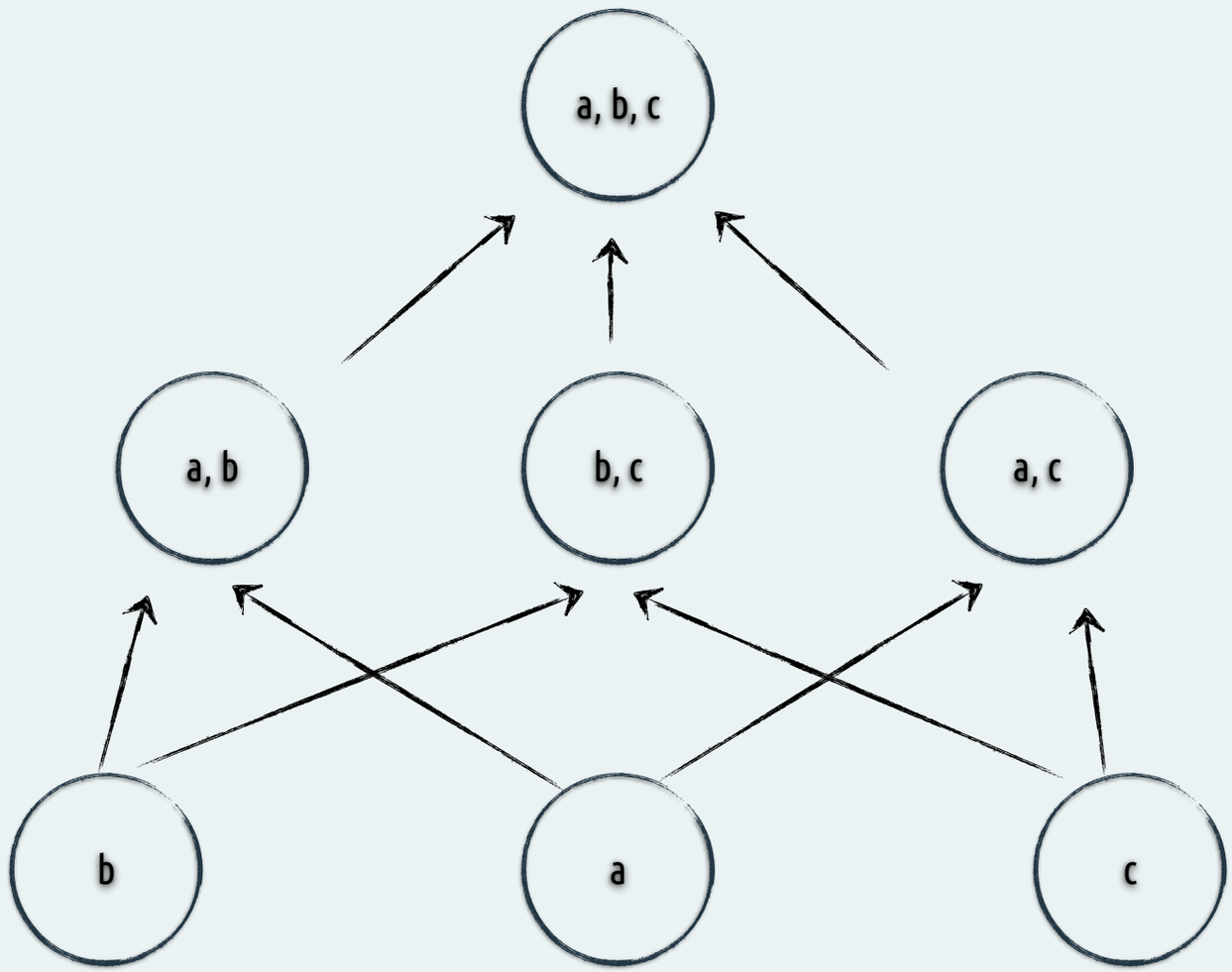
Idempotent: X⊔X = X

Join Semi-lattice

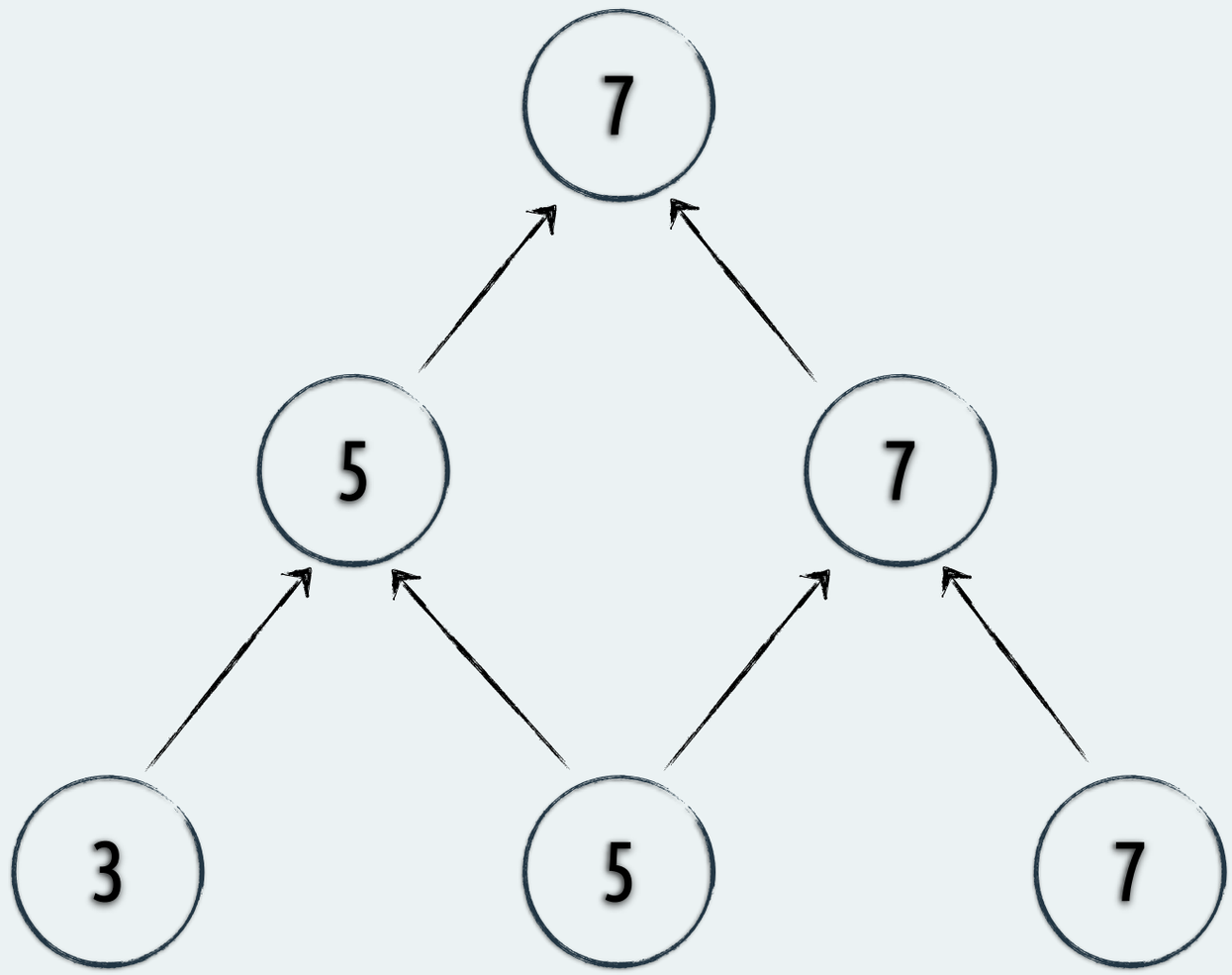
Objects grow over time; merge computes LUB

Join Semi-lattice

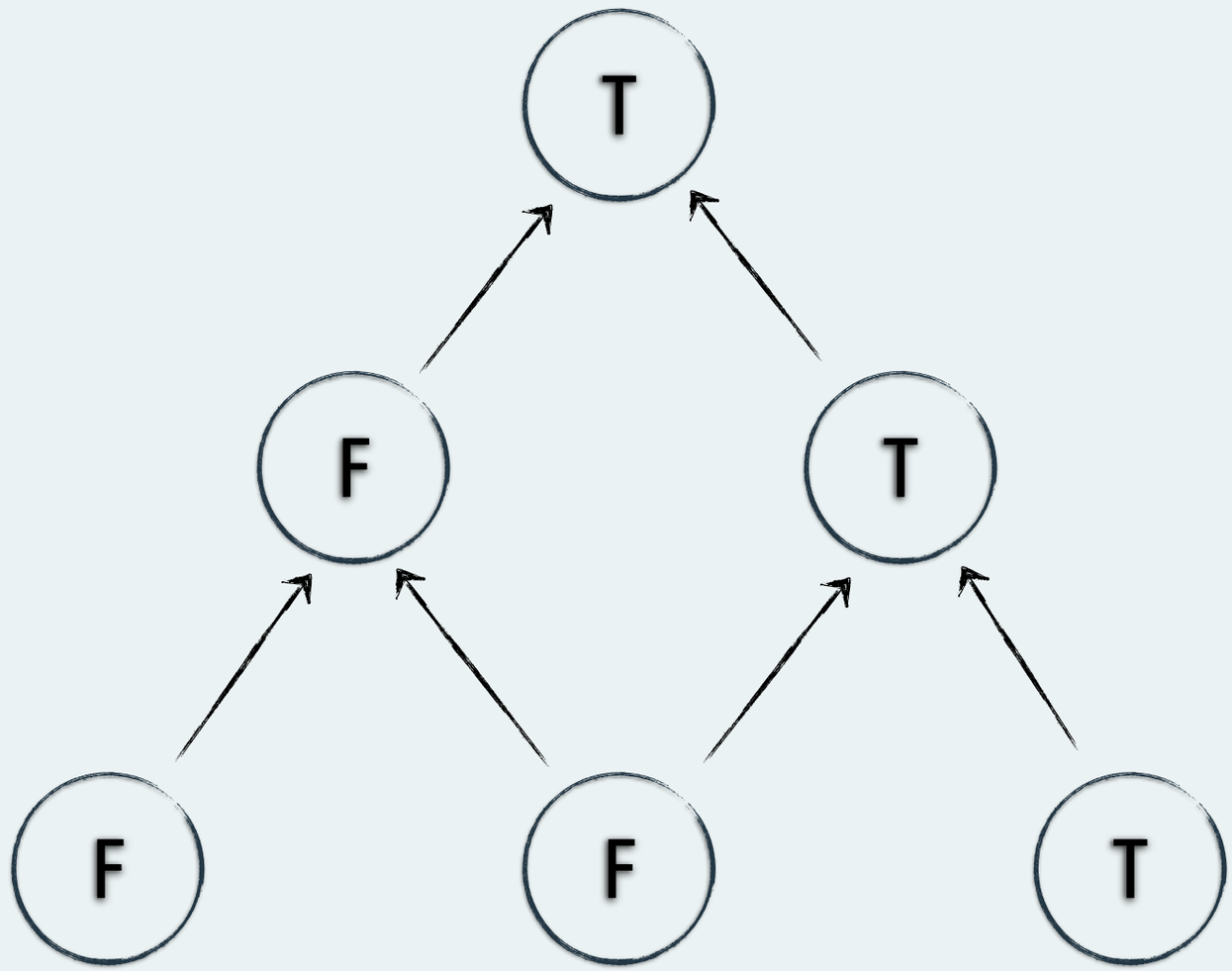
Examples



Set; merge function: union.



Increasing natural; merge function: max.



Booleans; merge function: or.

Merge

Deterministic
Idempotent
Associative
Commutative

Reusable defined semantics

Evolution of a Set

G-SET

Evolution of a Set

G-SET

Shelly

Bob

Shelly

Bob

Pete

Shelly

Bob

Pete

Anna

Joe

Shelly

Bob

Pete

Anna

Joe

Shelly

Reece

Pete

Alex

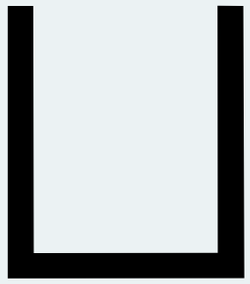
Shelly

Bob

Pete

Anna

Joe



Shelly

Reece

Pete

Alex

Shelly

Bob

Pete

Anna

Joe

Reece

Alex

Evolution of a Set

G-SET

2P-SET

Adds

Shelly

Bob

Pete

Anna

Shelly

Removes

Shelly

Bob

Pete

Adds

Shelly

Bob

Pete

Anna

Removes

Shelly

Bob

Pete

=

Anna

Evolution of a Set

U-SET

Evolution of a Set

U-SET

OR-SET

Adds	
1	Shelly
2	Bob
3	Pete
4	Anna

Removes	
1	Shelly
2	Bob
3	Pete

Adds	
1	Shelly
2	Bob
3	Pete
4	Anna
5	Shelly

Removes	
1	Shelly
2	Bob
3	Pete

Replica A

Adds

1	Shelly
2	Bob
3	Pete

Replica A

Adds

1	Shelly
2	Bob
3	Pete

Remove

Removes

1	Shelly
2	Bob
3	Pete

Replica B

Adds

4	Anna
5	Shelly

Adds	
1	Shelly
2	Bob
3	Pete

Removes	
1	Shelly
2	Bob
3	Pete

∪

Adds	
4	Anna
5	Shelly

Adds	
1	Shelly
2	Bob
3	Pete
4	Anna
5	Shelly

Removes	
1	Shelly
2	Bob
3	Pete

=

Anna
Shelly

Semantics

Add

Wins

Evolution of a Set

U-SET

OR-SET

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

An Optimized Conflict-free Replicated Set

Annette Bieniusa, INRIA & UPMC, Paris, France
Marek Zawirski, INRIA & UPMC, Paris, France
Nuno Preguiça, CITI, Universidade Nova de Lisboa, Portugal
Marc Shapiro, INRIA & LIP6, Paris, France
Carlos Baquero, HASLab, INESC TEC & Universidade do Minho, Portugal
Valter Balegas, CITI, Universidade Nova de Lisboa, Portugal

Sérgio Duarte CITI, Universidade Nova de Lisboa, Portugal

11 Oct 2012

Dotted Version Vectors: Logical Clocks for Optimistic Replication

Nuno Preguiça
CITI/DI
FCT, Universidade Nova de Lisboa
Monte da Caparica, Portugal
nmp@di.fct.unl.pt

Carlos Baquero, Paulo Sérgio Almeida,
Victor Fonte, Ricardo Gonçalves
CCTC/DI
Universidade do Minho
Braga, Portugal
{cbm,psa,vff}@di.uminho.pt, rtg@lsd.di.uminho.pt

Abstract

In cloud computing environments, a large number of users access data stored in highly available storage systems. To provide good performance to geographically disperse users and allow operation even in the presence of failures or network partitions, these systems often rely on optimistic replication solutions that guarantee only eventual consistency. In this scenario, it is important to be able to accurately and efficiently

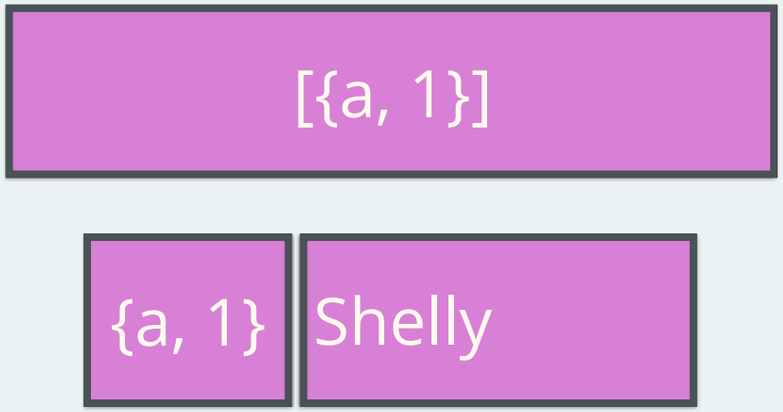
The mentioned systems follow a design where the data store is always writable. A consequence is that replicas of the same data item are allowed to diverge, and this divergence should later be repaired. Accurate tracking of concurrent data updates can be achieved by a careful use of well established causality tracking mechanisms [5], [6], [7], [8]. In particular, for data storage systems, version vectors [6] enables the system to compare any pair of replica versions and detect if

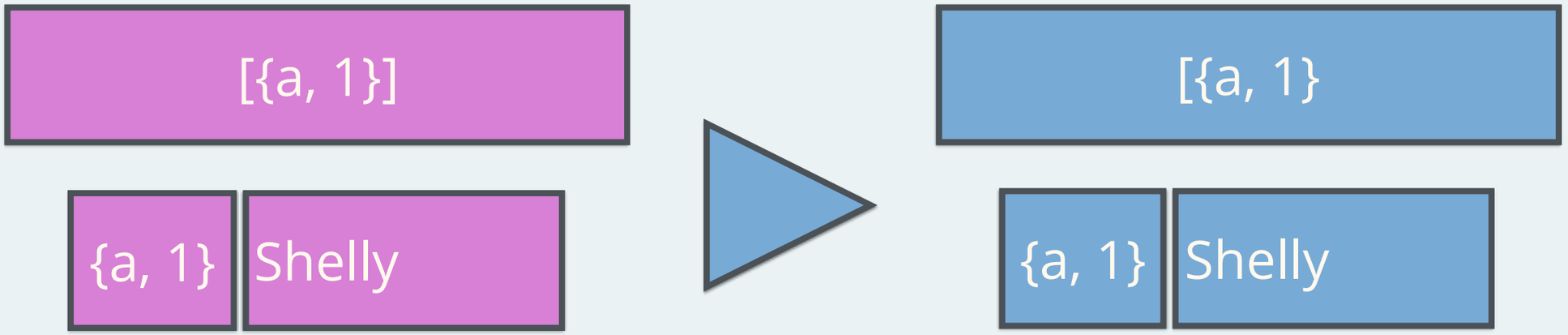
Evolution of a Set

U-SET

OR-SET

OR-SWOT


[{a, 1}]
{a, 1}
Shelly


[{a, 1}]
{a, 1}
Shelly
[{a, 1}
{a, 1}
Shelly

[{a, 1}]

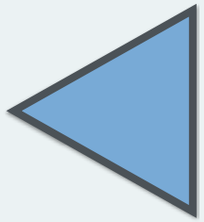
{a, 1}	Shelly

[{a, 1}, {b, 3}]

{a, 1}	Shelly
{b, 1}	Bob
{b, 2}	Phil
{b, 3}	Pete

[{a, 1}, {b,3}]

{a, 1}	Shelly
{b, 1}	Bob
{b, 2}	Phil
{b, 3}	Pete



[{a, 1}, {b, 3}]

{a, 1}	Shelly
{b, 1}	Bob
{b, 2}	Phil
{b, 3}	Pete

[{a, 2}, {b, 3}]

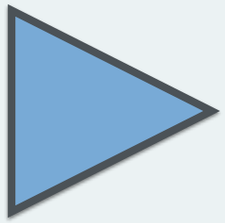
{a, 1}	Shelly
{b, 1}	Bob
{b, 3}	Pete
{a, 2}	Anna

[{a, 1}, {b, 3}]

{a, 1}	Shelly
{b, 1}	Bob
{b, 2}	Phil
{b, 3}	Pete

[{a, 2}, {b, 3}]

{a, 1}	Shelly
{b, 1}	Bob
{b, 3}	Pete
{a, 2}	Anna



[{a, 2}, {b, 3}]

{a, 1}	Shelly
{b, 1}	Bob
{b, 3}	Pete
{a, 2}	Anna

CRDTs

- Principled Merge
- Data Types with Defined Semantic
- Fine Grained Causality
- Building Block of EC Systems