Conflict-Free Replicated Data Type

Lattice Theory for Parallel Programming

Pierre Talbot

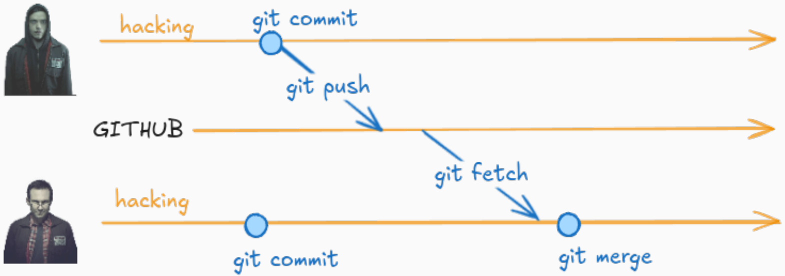
pierre.talbot@uni.lu

1st October 2025

University of Luxembourg

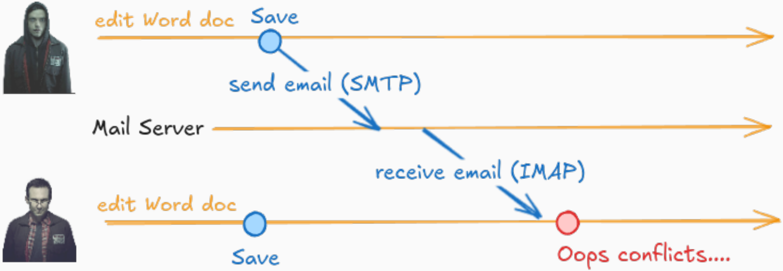
Motivation

How to Collaborate? (Git Example)



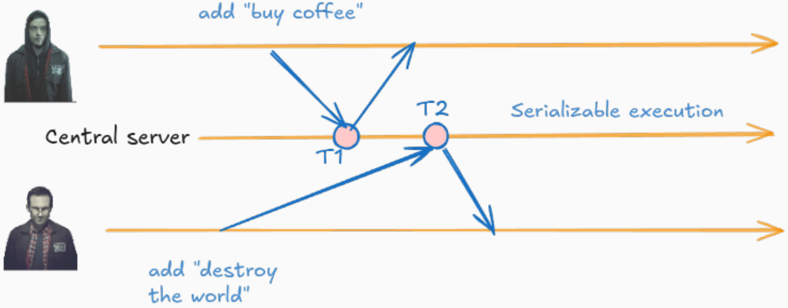
Conflicts are (semi-automatically) resolved by the users!

How to Collaborate? (Word Example)

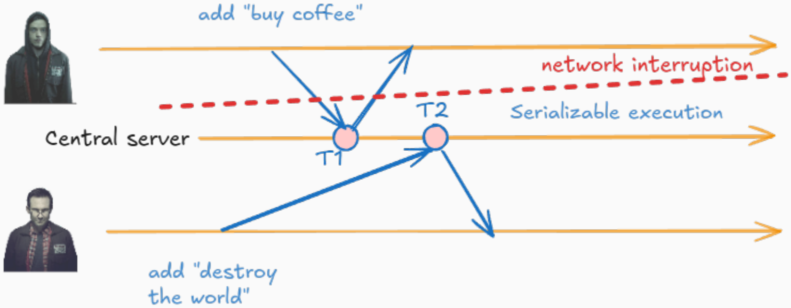


Conflicts are resolved by the users!

How to Collaborate? (Todolist Example)



How to Collaborate? (Todolist Example)

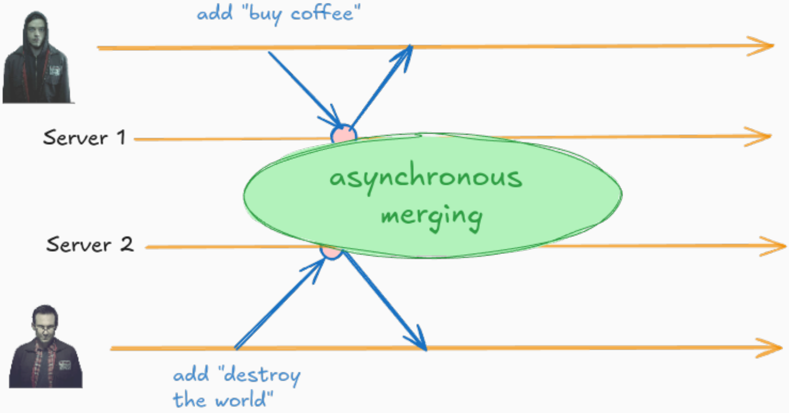


System fails under network partition.

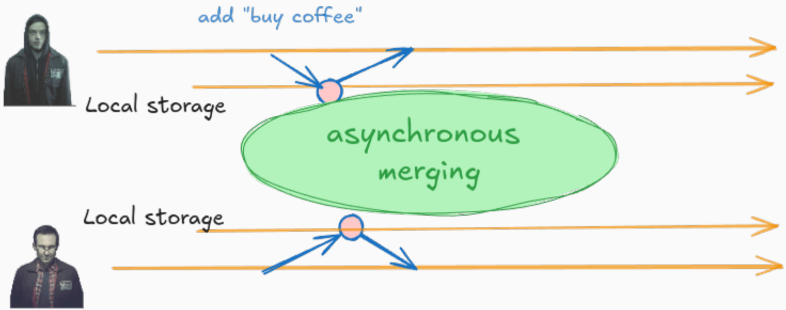
Central Server Issues

1. User waits for round-trip (latency).
2. Single point of failure (DDOS).
3. Require constant connectivity.

Fixing Latency and DDOS

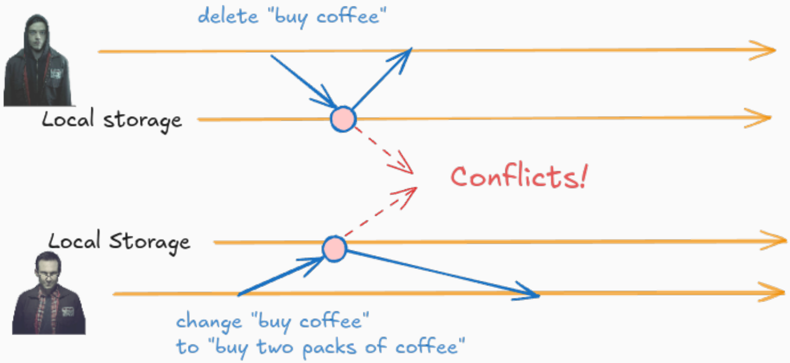


Fixing Constant Connectivity

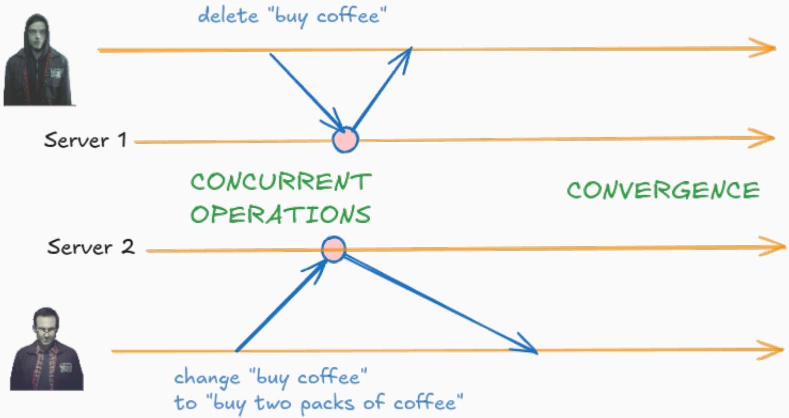


A paradigm sometimes called "*local-first application*".

But Conflicts...



What Do We Want?



Eventual Consistency

What do we want?

- **Concurrent operations**: happens without knowing about each other.
- **Convergence**: same eventual state.

Properties

- **Eventual delivery**: eventually, every operation is seen by every node. But asynchronous: no assumption on the order.
- **Convergence**: Seen same operations $\Rightarrow$ have same state.
- **Don't lose data**: can happen in some systems (e.g., last writer wins).

CRDTs to the Rescue!

We are going to explore three frameworks:

- **Operational Transformation**: historical approach.
- **Operation-based CRDTs**: communicating the operations.
- **State-based CRDTs**: communicating the states.

Operational Transformation

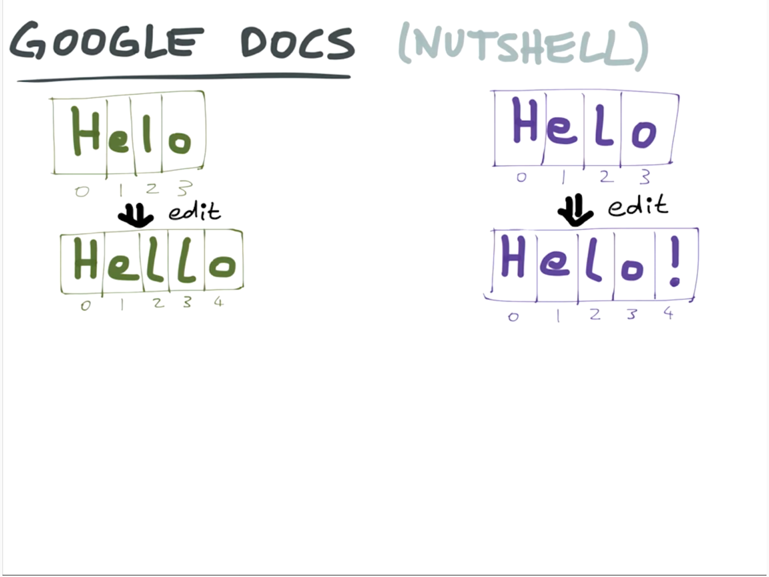
The nice drawings in this section are taken from the CodeMesh 2016 talk of Martin Kleppmann.

Source: `https://www.youtube.com/watch?v=8_DfwEpHE88`

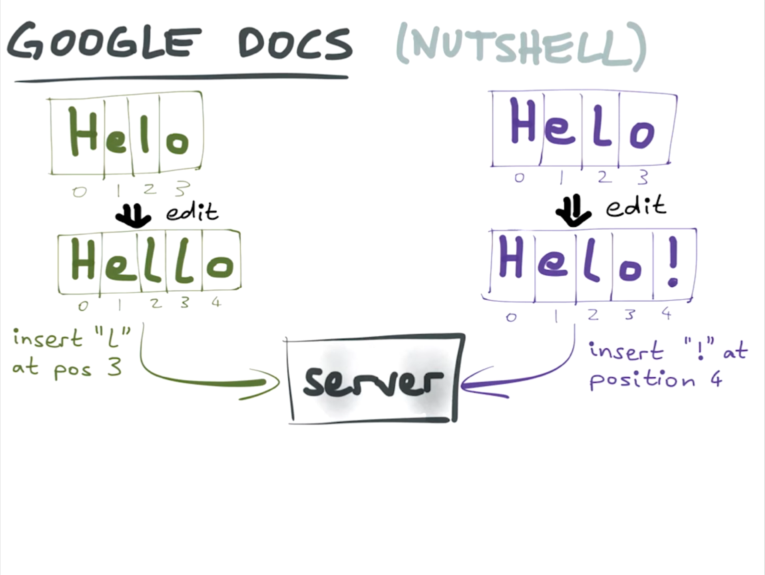
Collaborative Applications

- (1989–): Operational Transformation (OT): Google Docs, MS Office Online
- (2006–): Conflict-Free Replicated Data Types (CRDTs): Riak, League of Legends (chat system), Angry Birds, TomTom GPS, ...
- See also `https://christophermeiklejohn.com/erlang/lasp/2019/03/08/monotonicity.html`

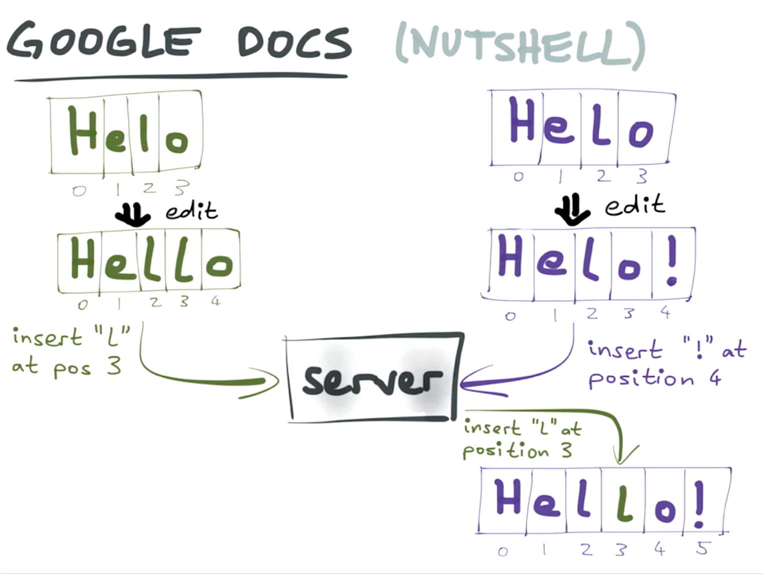
Collaborative Text Editing



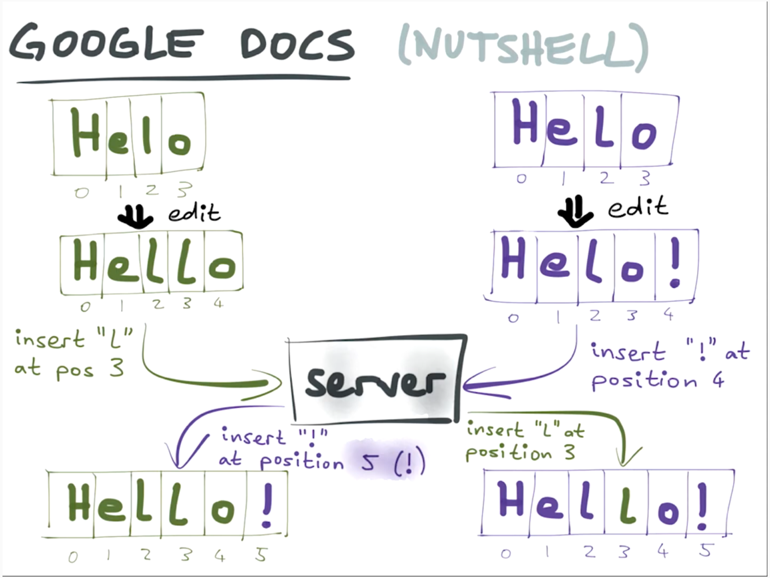
Collaborative Text Editing



Collaborative Text Editing



Collaborative Text Editing



Operational Transformation

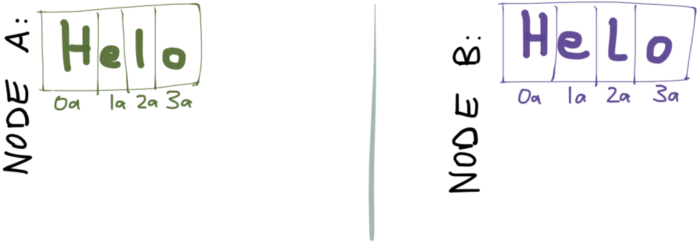
- The “insert ! at position 4” has been *transformed* to “insert ! at position **5**”.
- Most of the papers on OT are wrong!! (Including the first one by Ellis & Gibbs, 1989).
- The ones correct usually assume all operations needs to go through a central server (Google Docs).
- Key role of the server: sequencing the operations.
- Jupiter (Nichols et al. 1995) the basis of Google docs, Etherpad, ...

Operation-based CRDTs

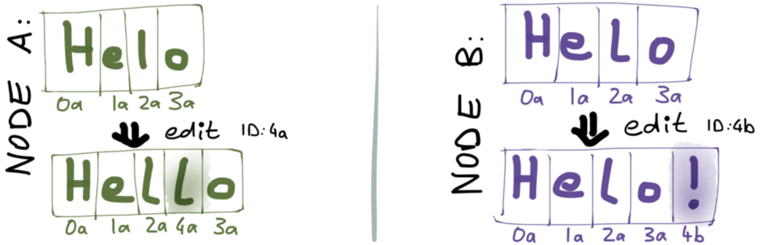
This section is based on the slides of Martin Kleppmann.

Source: `https://www.infoq.com/presentations/crdt-distributed-consistency/`

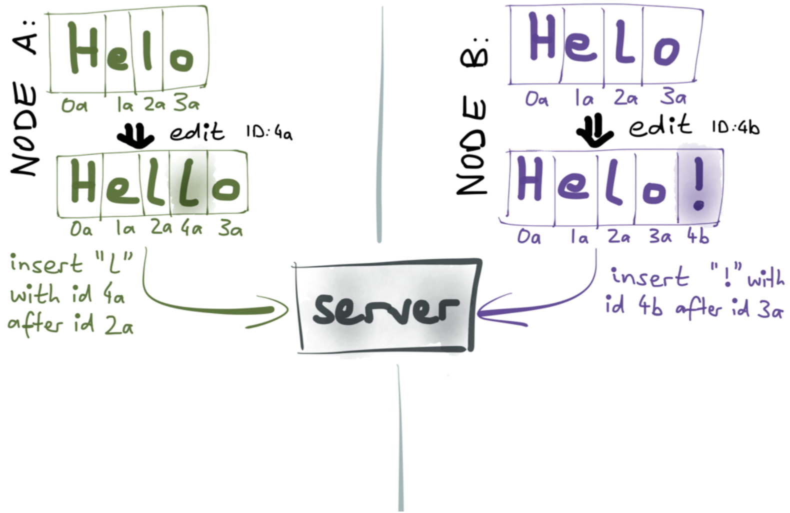
ORDERED LIST CRDT (NUTSHELL)



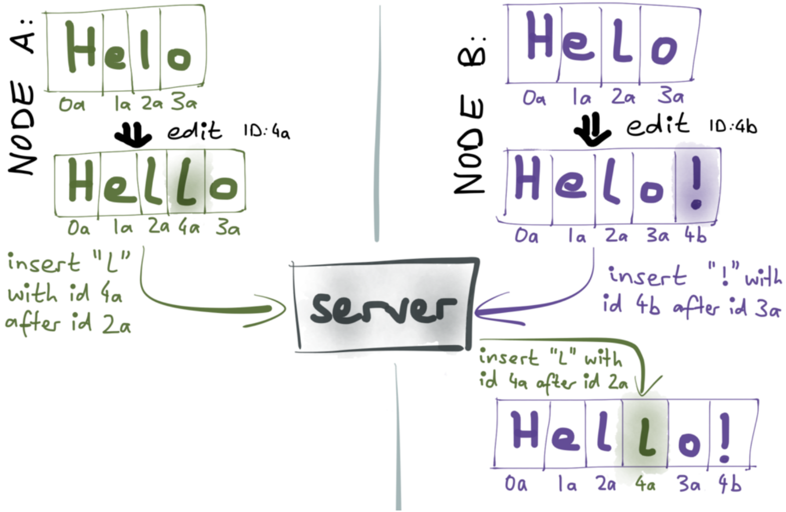
ORDERED LIST CRDT (NUTSHELL)



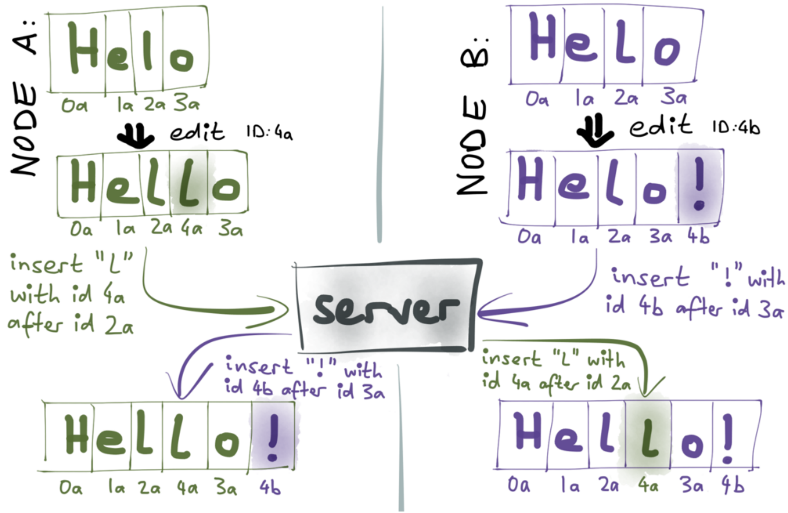
ORDERED LIST CRDT (NUTSHELL)



ORDERED LIST CRDT (NUTSHELL)



ORDERED LIST CRDT (NUTSHELL)



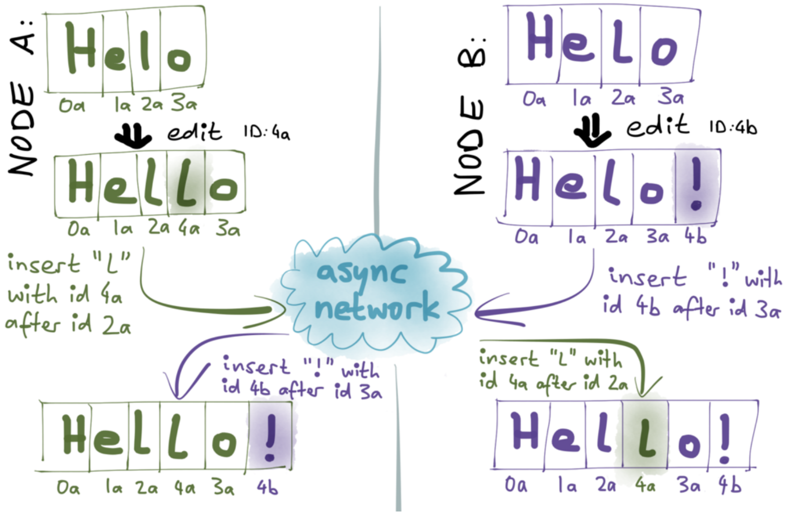
insert "L" with id 4a after id 2a

insert "!" with id 4b after id 3a

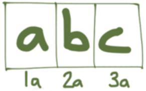
insert "!" with id 4b after id 3a

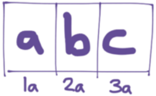
insert "L" with id 4a after id 2a

ORDERED LIST CRDT (NUTSHELL)

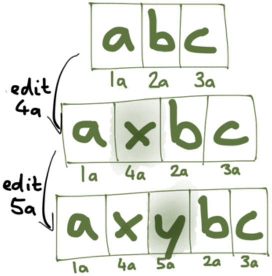


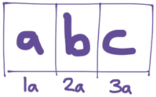
INSERTING IN THE SAME PLACE



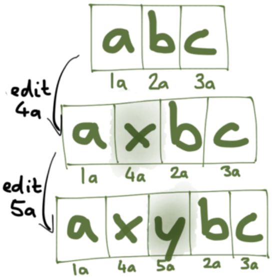


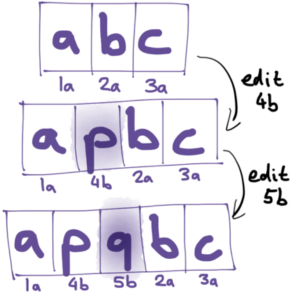
INSERTING IN THE SAME PLACE



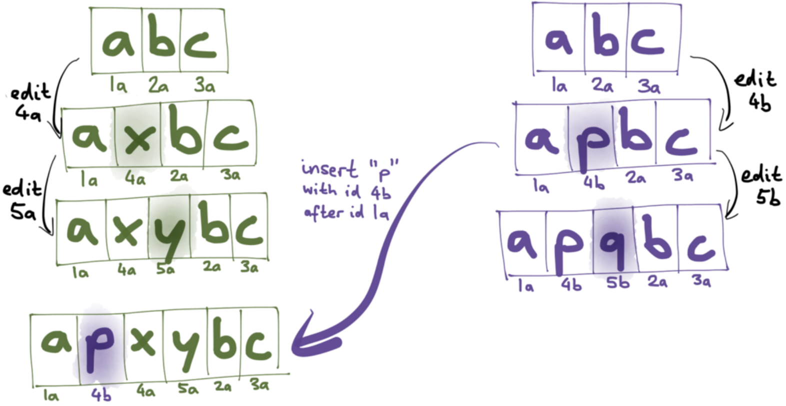


INSERTING IN THE SAME PLACE

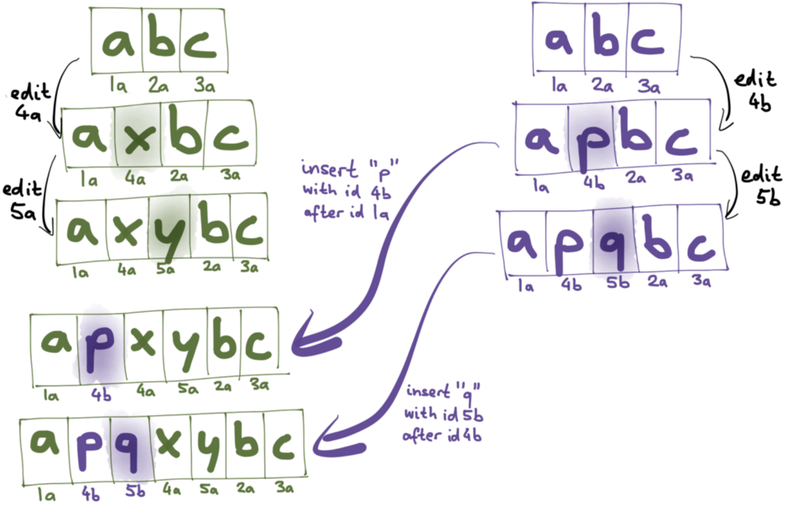




INSERTING IN THE SAME PLACE



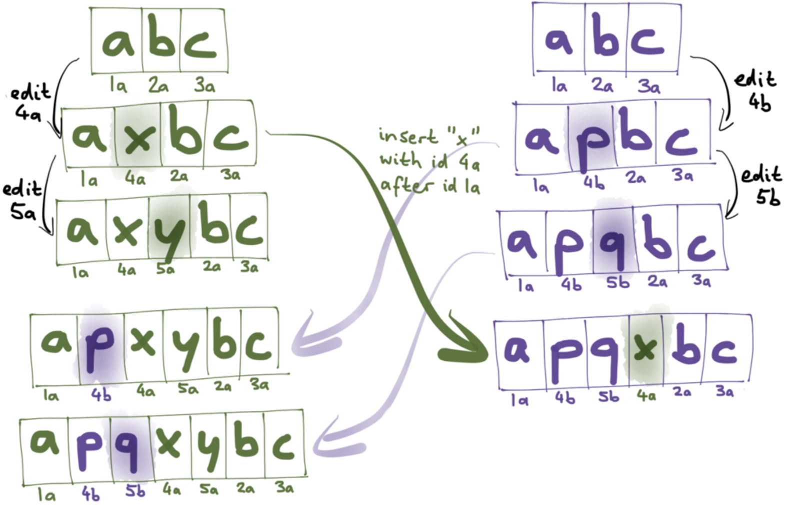
INSERTING IN THE SAME PLACE



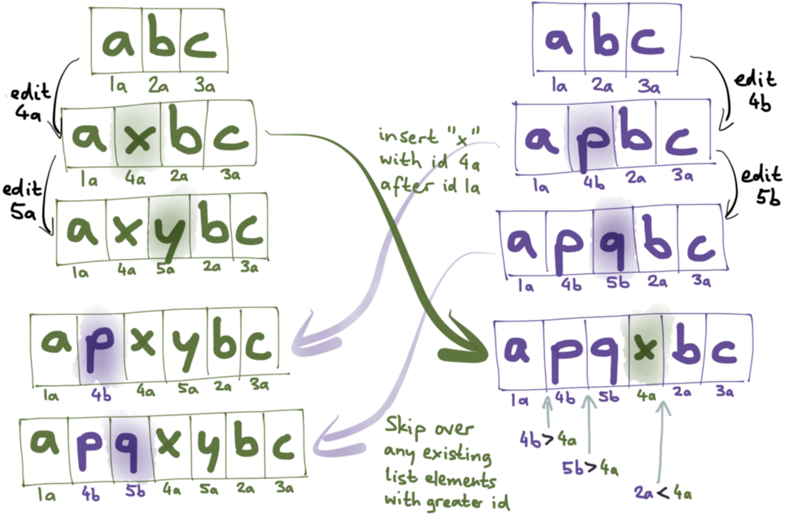
insert "p" with id 4b after id 1a

insert "q" with id 5b after id 4b

INSERTING IN THE SAME PLACE



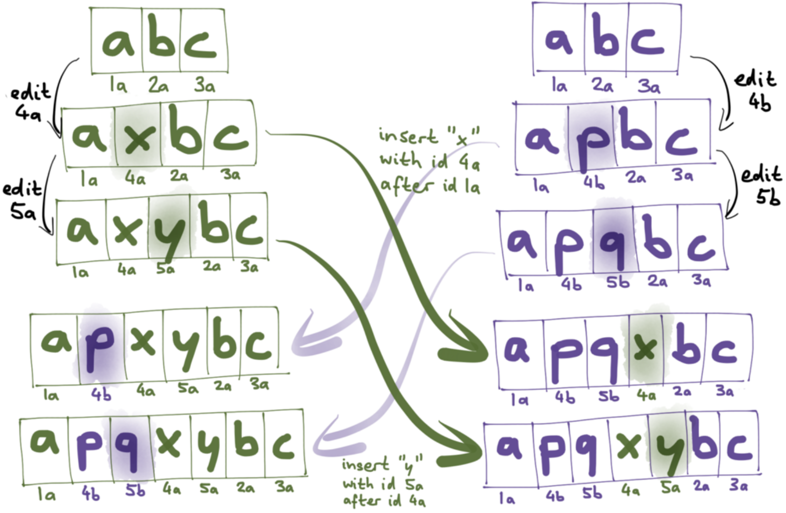
INSERTING IN THE SAME PLACE



insert "x" with id 4a after id 1a

Skip over any existing list elements with greater id

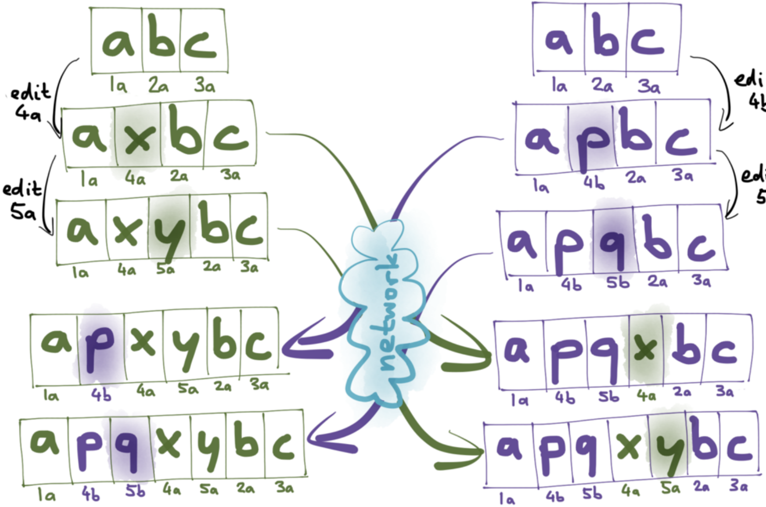
INSERTING IN THE SAME PLACE



insert "x" with id 4a after id 1a

insert "y" with id 5a after id 4a

INSERTING IN THE SAME PLACE



CRDT Properties

CRDTs allow collaboration without assumptions about the network topology.

Theorem: Convergence guarantee

Whenever two nodes have seen the same set of operations, possibly in a different order, they are in the same state.

Proof.

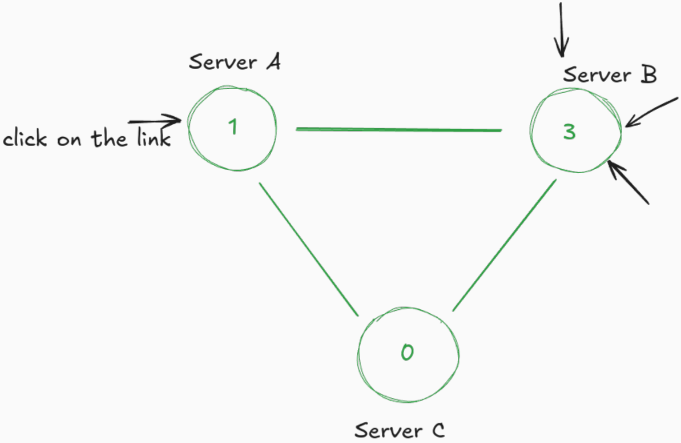
Essentially relying on commutative property of the operations. □

Intuitive properties

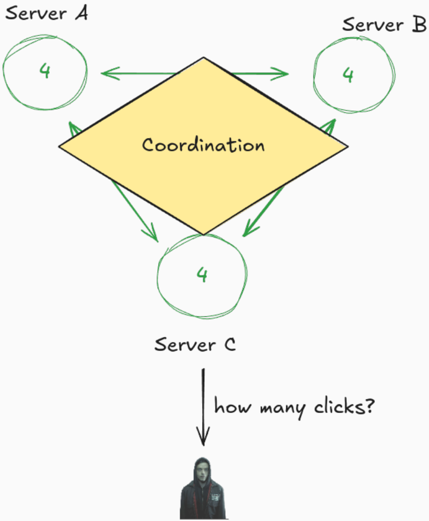
CRDTs work even if messages are delayed, duplicated and reordered.

State-based CRDTs

Distributed Counter



Distributed Counter



Definition of CRDT

Key Observation: It is not necessary to have the true latest global value, as long as we obtain it eventually.

Definition

A conflict-free replicated data type (CRDT) is a tuple $\langle L, \leq, \sqcup, S, value, f_1, \ldots, f_n \rangle$ where:

- $\langle L, \leq, \sqcup \rangle$ is a lattice[1].
- S is a set of values.
- $f_1, \ldots, f_n$ are monotone and extensive functions over L (extensive: $\forall x, x \leq f(x)$).
- $value : L \to S$ returns the value modelled by the CRDT.

[1] A join semi-lattice to be precise, because we don't need the meet operation $\sqcap$.

Connections Between Distributed Computing and Lattice Theory

In distributed computing terms, we have replicas on different nodes.
An element of a CRDT $r \in L$ is only one replica (all replicas is a collection of elements in L).

- A value $v \in L$ is called the **payload**.
- The *join operation* $\sqcup$ is called **merge**.
- We have one or more **update** operations mutating our state locally and monotonically.
- A function **value** is used to get the current value modelled by the CRDT. Note that if $L = S$, we can have *value* to be the identity function.

Synchronization among the replicas is not part of the “public API” and happens in the background. New values are merged in the current state using the *merge* function.

First Try: Grow-only Counter (G-Counter)

Suppose the lattice $\langle \mathbb{Z}, \leq \rangle$, that is the chain of integers.

Let's try to design a CRDT from this lattice!

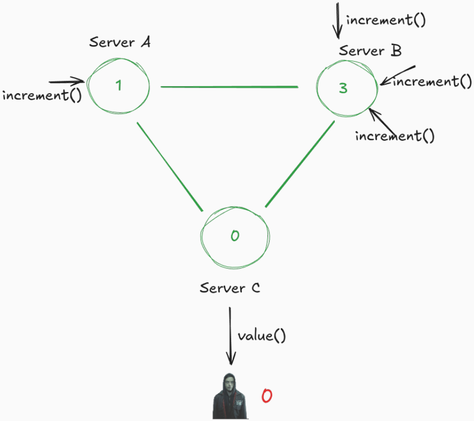
First Try: Grow-only Counter (G-Counter)

Suppose the lattice $\langle \mathbb{Z}, \leq \rangle$, that is the chain of integers.
Let's try to design a CRDT from this lattice!

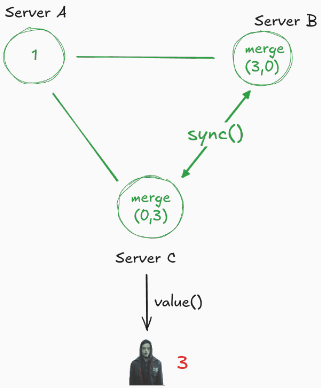
First try: Let the CRDT $G_1 \triangleq \langle \mathbb{Z}, \leq, \mathit{merge}, \mathbb{Z}, \mathit{value}, \mathit{increment} \rangle$ where:

- $\mathit{increment}(x) \triangleq x + 1$
- $\mathit{merge}(x, y) \triangleq x + y$
- $\mathit{value}(x) \triangleq x$

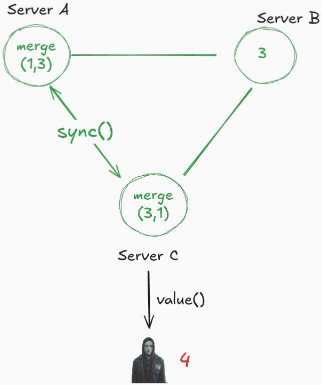
First Try: Grow-only Counter (G-Counter)



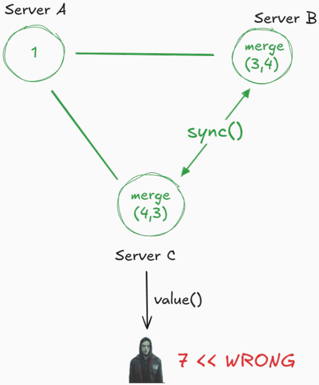
First Try: Grow-only Counter (G-Counter)



First Try: Grow-only Counter (G-Counter)



First Try: Grow-only Counter (G-Counter)



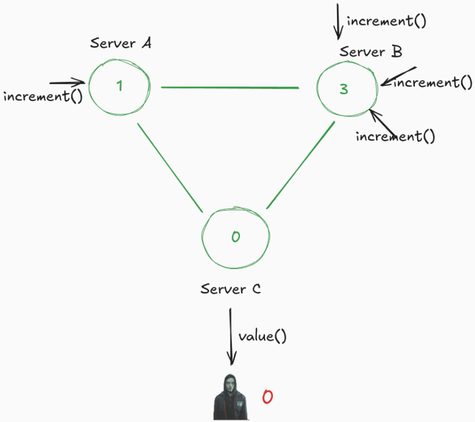
Second Try: Grow-only Counter (G-Counter)

Suppose the lattice $\langle \mathbb{Z}, \leq \rangle$, that is the chain of integers.
Let's try to design a CRDT from this lattice!

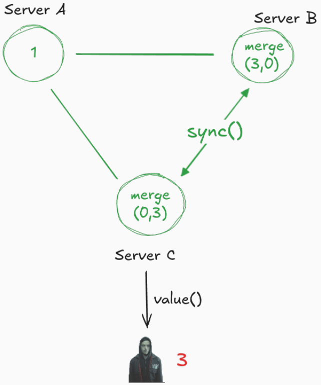
Second try: Let the CRDT $G_2 \triangleq \langle \mathbb{Z}, \leq, \mathit{merge}, \mathbb{Z}, \mathit{value}, \mathit{increment} \rangle$ where:

- $\mathit{increment}(x) \triangleq x + 1$
- $\mathit{merge}(x, y) \triangleq \mathit{max}(x, y)$
- $\mathit{value}(x) \triangleq x$

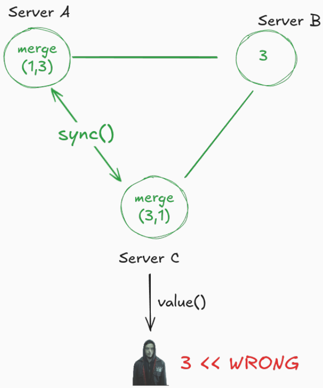
Second Try: Grow-only Counter (G-Counter)



Second Try: Grow-only Counter (G-Counter)



Second Try: Grow-only Counter (G-Counter)



Third Try: Grow-only Counter (G-Counter)

Let n be the number of replicas (nodes of the distributed system).
Suppose the Cartesian product lattice $\langle \mathbb{Z}^n, \dot{\leq}, \dot{\sqcup} \rangle$, where:

- $(x_1, \ldots, x_n) \dot{\leq} (y_1, \ldots, y_n) \Leftrightarrow \forall 1 \leq i \leq n, x_i \leq y_i$.
- $(x_1, \ldots, x_n) \dot{\sqcup} (y_1, \ldots, y_n) \triangleq (x_1 \sqcup y_1, \ldots, x_n \sqcup y_n)$.

Third Try: Grow-only Counter (G-Counter)

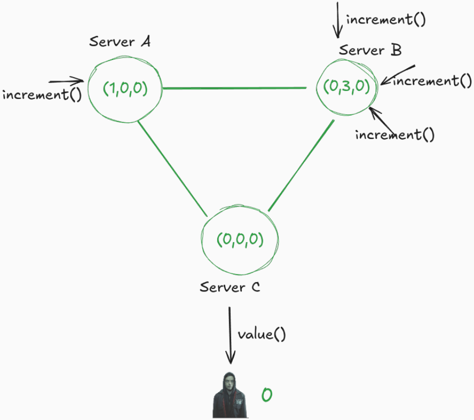
Let n be the number of replicas (nodes of the distributed system).
Suppose the Cartesian product lattice $\langle \mathbb{Z}^n, \dot{\leq}, \dot{\sqcup} \rangle$, where:

- $(x_1, \ldots, x_n) \;\dot{\leq}\; (y_1, \ldots, y_n) \Leftrightarrow \forall 1 \leq i \leq n, x_i \leq y_i$.
- $(x_1, \ldots, x_n) \;\dot{\sqcup}\; (y_1, \ldots, y_n) \triangleq (x_1 \sqcup y_1, \ldots, x_n \sqcup y_n)$.

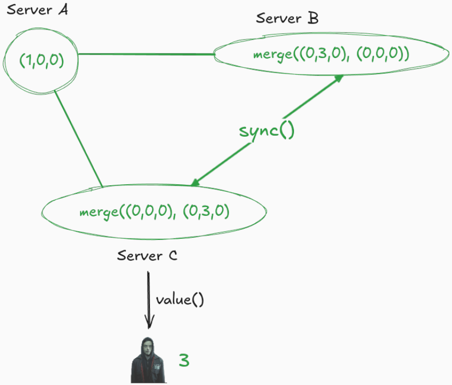
Third try: Let the CRDT $G_3 \triangleq \langle \mathbb{Z}^n, \dot{\leq}, \mathit{merge}, \mathbb{Z}, \mathit{value}, \mathit{increment} \rangle$ where:

- $\mathit{increment}(\langle x_1, \ldots, x_n \rangle) \triangleq \langle x_1, \ldots, x_{id} + 1, \ldots, x_n \rangle$ where *id* is the ID of the current node.
- $\mathit{merge}(\langle x_1, \ldots, x_n \rangle, \langle y_1, \ldots, y_n \rangle) \triangleq \langle \mathit{max}(x_1, y_1), \ldots, \mathit{max}(x_n, y_n) \rangle$.
- $\mathit{value}(\langle x_1, \ldots, x_n \rangle) \triangleq \sum_{1 \leq i \leq n} x_i$.

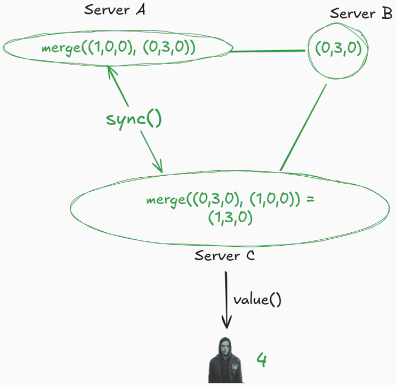
Third Try: Grow-only Counter (G-Counter)



Third Try: Grow-only Counter (G-Counter)



Third Try: Grow-only Counter (G-Counter)



Resources

Resources

- Website about CRDT: `crdt.tech`
- Martin Kleppmann presentation (operation-based CRDTs): `https://www.youtube.com/watch?v=8_DfwEpHE88`
- John Mumm presentation (state-based CRDTs): `https://www.youtube.com/watch?v=OOlnp2bZVRs`
- *Conflict-free Replicated Data Types*, Nuno Preguiça, Carlos Baquero, and Marc Shapiro (2018)
- *A comprehensive study of Convergent and Commutative Replicated Data Types*, Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski (2011).