

# Efficient renaming in Conflict-free Replicated Data Types (CRDTs)

---

Matthieu Nicolas

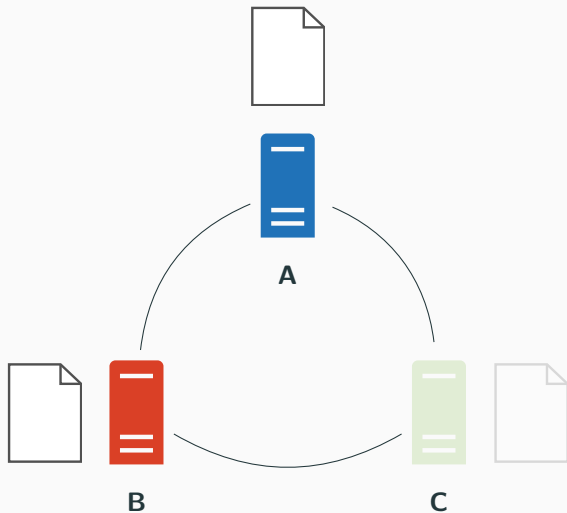
COAST team

**Supervised by** Gérald Oster and Olivier Perrin

December 4, 2018

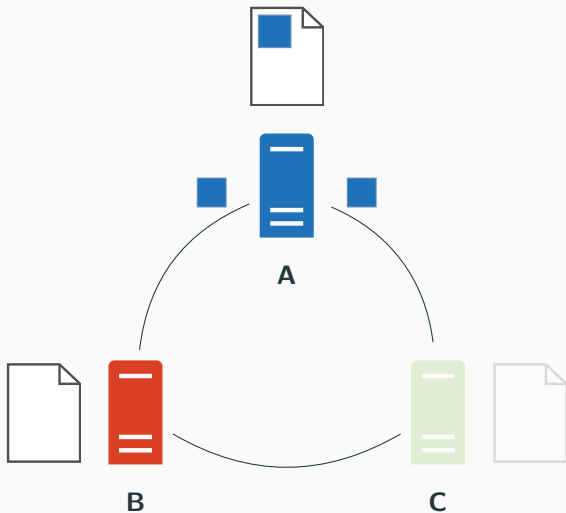


# Conflict-free Replicated Data Types (CRDTs) [3]



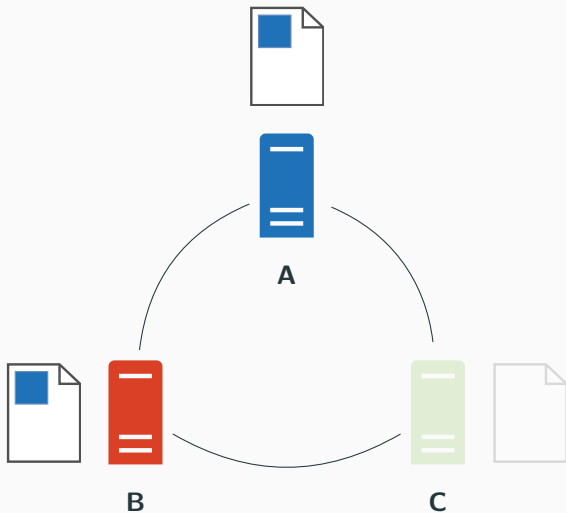
- Replicated data structure

# Conflict-free Replicated Data Types (CRDTs) [3]



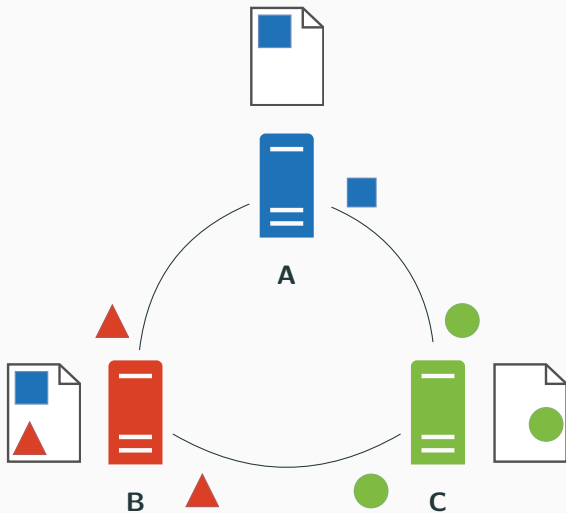
- Replicated data structure
- Updates performed without coordination

# Conflict-free Replicated Data Types (CRDTs) [3]



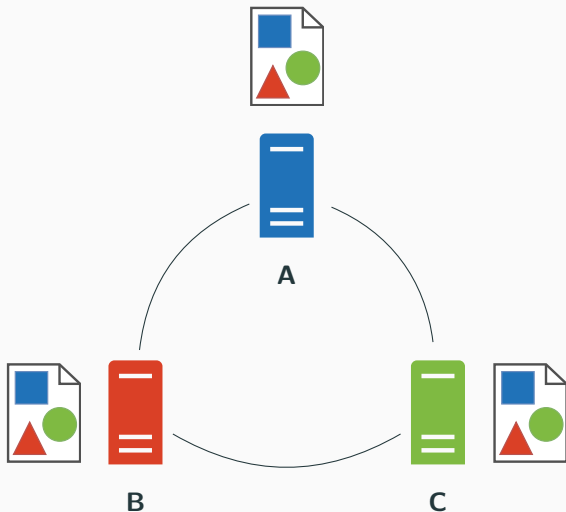
- Replicated data structure
- Updates performed without coordination

# Conflict-free Replicated Data Types (CRDTs) [3]



- Replicated data structure
- Updates performed without coordination

# Conflict-free Replicated Data Types (CRDTs) [3]



- Replicated data structure
- Updates performed without coordination
- Strong Eventual Consistency [3]

# Identifier-based CRDTs

## Main idea

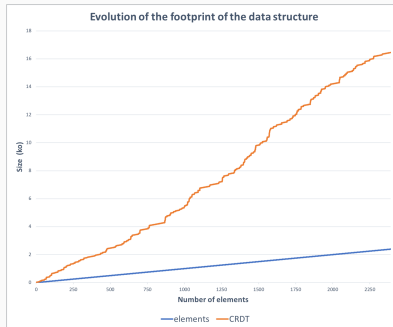
- Attach an identifier to each element

## Allow to design commutative updates

- Identifying uniquely elements
- Ordering updates causally
- ...

## Limits

- Unbounded size of identifiers
- Overhead of the data structure increasing over time



**Figure 1:** Evolution of the footprint of the data structure



**How to reduce the overhead introduced by  
the data structure ?**

**How to reduce the overhead introduced by  
the data structure ?**

**Reassign shorter identifiers in a fully  
distributed manner**

- State of the art of *Sequence CRDTs*
- Elements are ordered by their identifier, noted here as lowercase letters

- State of the art of *Sequence CRDTs*
- Elements are ordered by their identifier, noted here as lowercase letters



**Figure 2:** State of a sequence which contains the elements "helo" and their corresponding identifiers

- State of the art of *Sequence CRDTs*
- Elements are ordered by their identifier, noted here as lowercase letters

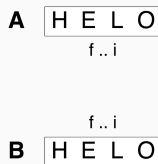


**Figure 2:** State of a sequence which contains the elements "helo" and their corresponding identifiers



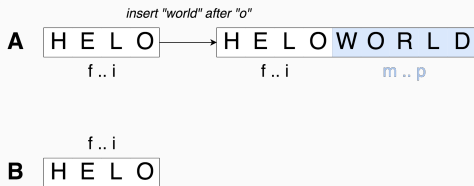
**Figure 3:** State of a sequence which contains the block "helo"

# Example



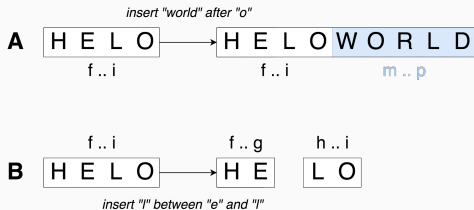
**Figure 4:** Example of concurrent *insert* operations

# Example



**Figure 4:** Example of concurrent *insert* operations

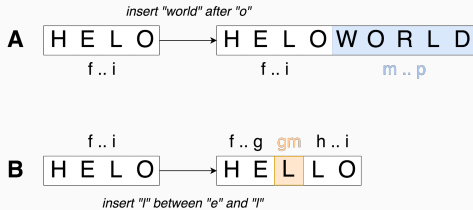
# Example



**Figure 4:** Example of concurrent *insert* operations

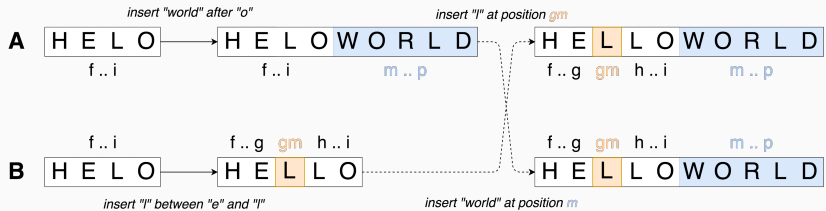


# Example



**Figure 4:** Example of concurrent *insert* operations

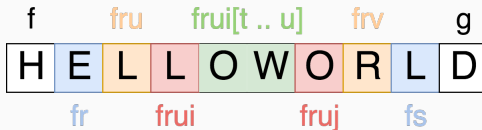
# Example



**Figure 4:** Example of concurrent *insert* operations

# Declining performances

- Updates performed may lead to an inefficient internal representation



**Figure 5:** Example of inefficient internal representation

- The more blocks we have:
  - The more metadata we store
  - The longer it takes to browse the sequence to *insert* or *delete* an element

# Renaming mechanism

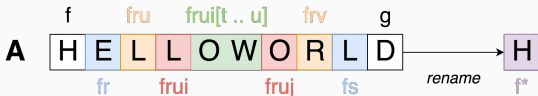
- Introduce a *rename* operation



**Figure 6:** Example of renaming

# Renaming mechanism

- Introduce a *rename* operation



**Figure 6:** Example of renaming

- Generates a new identifier to the first element, based on its previous identifier

# Renaming mechanism

- Introduce a *rename* operation



**Figure 6:** Example of renaming

- Generates a new identifier to the first element, based on its previous identifier
- Then generates contiguous identifiers for all following elements

# Renaming mechanism

- Introduce a *rename* operation

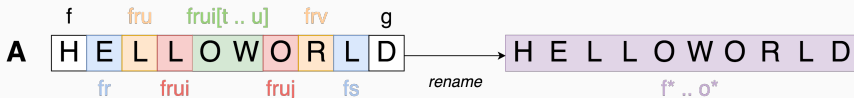


**Figure 6:** Example of renaming

- Generates a new identifier to the first element, based on its previous identifier
- Then generates contiguous identifiers for all following elements

# Renaming mechanism

- Introduce a *rename* operation



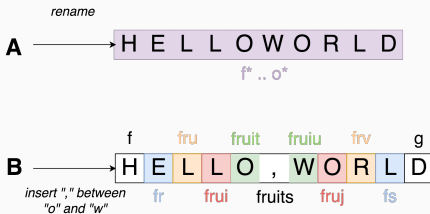
**Figure 6:** Example of renaming

- Generates a new identifier to the first element, based on its previous identifier
- Then generates contiguous identifiers for all following elements



# Handling concurrent operations

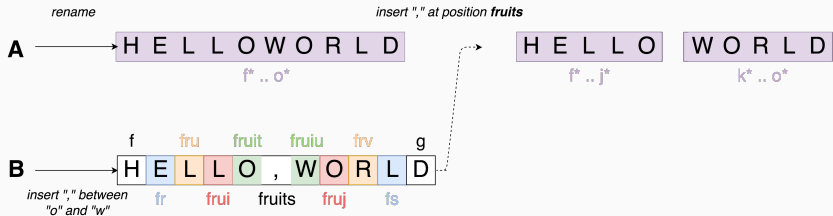
- Others may perform updates concurrently to a *rename* operation



**Figure 7:** Example of concurrent insert

# Handling concurrent operations

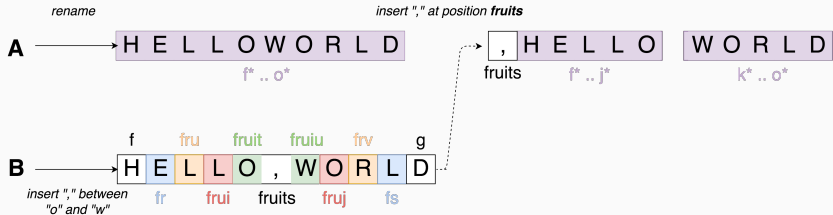
- Others may perform updates concurrently to a *rename* operation



**Figure 7:** Example of concurrent insert

# Handling concurrent operations

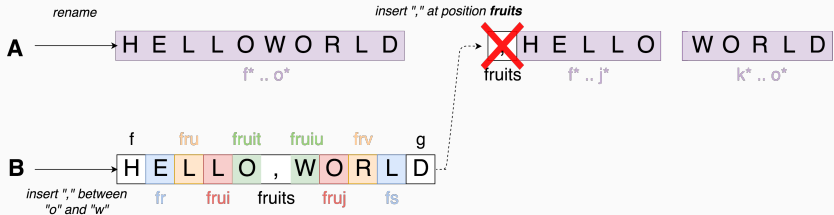
- Others may perform updates concurrently to a *rename* operation



**Figure 7:** Example of concurrent insert

# Handling concurrent operations

- Others may perform updates concurrently to a *rename* operation
- May lead to inconsistencies



**Figure 7:** Example of concurrent insert

# Handling concurrent operations

- Others may perform updates concurrently to a *rename* operation
- May lead to inconsistencies

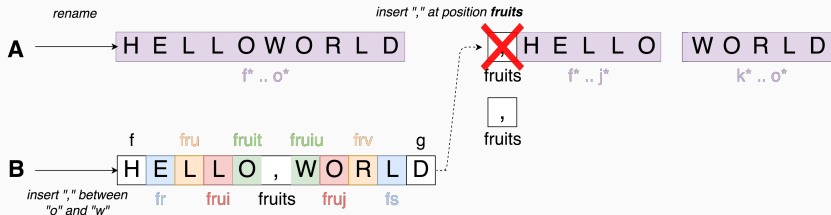


Figure 7: Example of concurrent insert

- Define rewriting rules to transform identifiers from one *epoch* to another

# Handling concurrent operations

- Others may perform updates concurrently to a *rename* operation
- May lead to inconsistencies

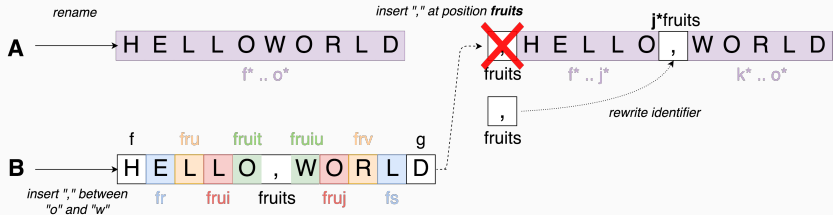


Figure 7: Example of concurrent insert

- Define rewriting rules to transform identifiers from one *epoch* to another

# Handling concurrent rename

- Define a total order between *rename* operations
- Pick a "winner" operation between concurrent *renames*
- Define additional rewriting rules to *undo* the effect of "losing" ones

# To wrap up

## Done

- Designed a *rename* operation for LogootSplit
- Defined rewriting rules to deal with concurrent updates

## Work in progress

- Implementation in MUTE [2], our P2P collaborative text editor
- Design the strategy to trigger the renaming

## To do

- Prove formally the correctness of the mechanism
- Benchmark its performances



## Generalize the approach

- To other Sequence CRDTs
- To other types
  - Counter
  - Set
  - ...

Thanks for your attention, any questions?



- [1] L. André, S. Martin, G. Oster, and C.-L. Ignat.

**Supporting adaptable granularity of changes for massive-scale collaborative editing.**

*In International Conference on Collaborative Computing: Networking, Applications and Worksharing - CollaborateCom 2013*, pages 50–59, Austin, TX, USA, Oct. 2013. IEEE Computer Society.

- [2] M. Nicolas, V. Elvinger, G. Oster, C.-L. Ignat, and F. Charoy.

**MUTE: A Peer-to-Peer Web-based Real-time Collaborative Editor.**

*In ECSCW 2017 - 15th European Conference on Computer-Supported Cooperative Work*, volume 1 of *Proceedings of 15th European Conference on Computer-Supported Cooperative Work - Panels, Posters and Demos*, pages 1–4, Sheffield, United Kingdom, Aug. 2017. EUSSET.

- [3] M. Shapiro, N. M. Preguiça, C. Baquero, and M. Zawirski.  
**Conflict-free replicated data types.**  
In *Proceedings of the 13th International Symposium on Stabilization, Safety, and Security of Distributed Systems, SSS 2011*, pages 386–400, 2011.