

# Entretien technique

---

**Matthieu Nicolas** (nicolasmattthieu57@gmail.com)

19 octobre 2023

## **Ingénieur Recherche & Développement**

2014-2017

*Inria, Loria, équipe Coast, Nancy*

- MUTE : conception et développement d'un éditeur de texte collaboratif temps réel web pair-à-pair (<https://mute.loria.fr>)
- PLM : webification d'un environnement d'apprentissage de la programmation  
(<http://people.irisa.fr/Martin.Quinson/Teaching/PLM/>)

## **Doctorat en informatique**

2017-2022

*Université de Lorraine, Loria, équipe Coast, Nancy*

- (Ré)Identification efficace dans les types de données répliquées sans conflit (CRDTs)

## **Ingénieur Recherche & Développement**

2014-2017

*Inria, Loria, équipe Coast, Nancy*

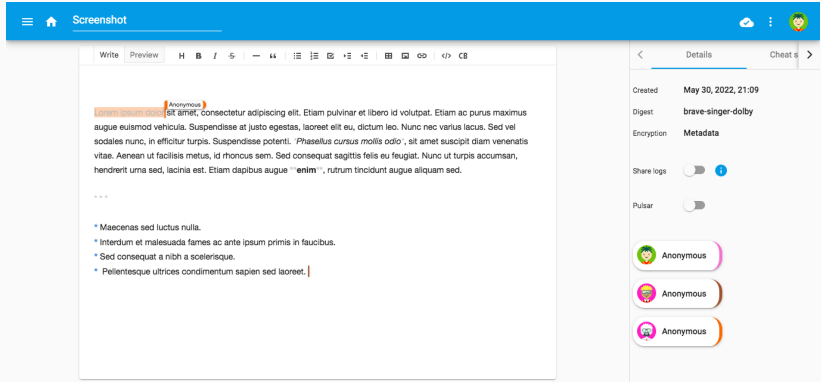
- MUTE : conception et développement d'un éditeur de texte collaboratif temps réel web pair-à-pair (<https://mute.loria.fr>)
- PLM : webification d'un environnement d'apprentissage de la programmation  
(<http://people.irisa.fr/Martin.Quinson/Teaching/PLM/>)

## **Doctorat en informatique**

2017-2022

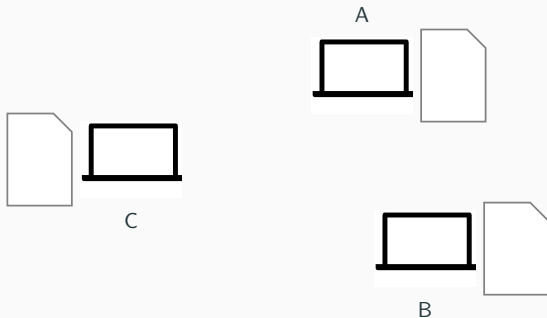
*Université de Lorraine, Loria, équipe Coast, Nancy*

- (Ré)Identification efficace dans les types de données répliquées sans conflit (CRDTs)

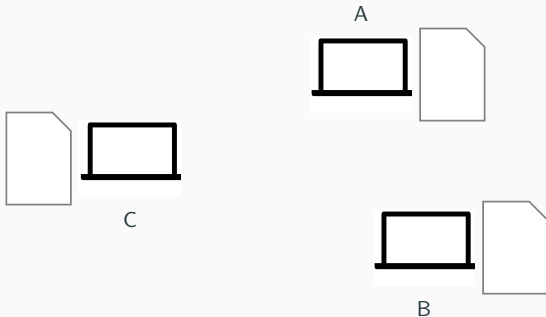


- Application pair-à-pair
- Permet de rédiger collaborativement des documents texte
- Garantit la confidentialité & souveraineté des données

# Réplication dans applications collaboratives pair-à-pair

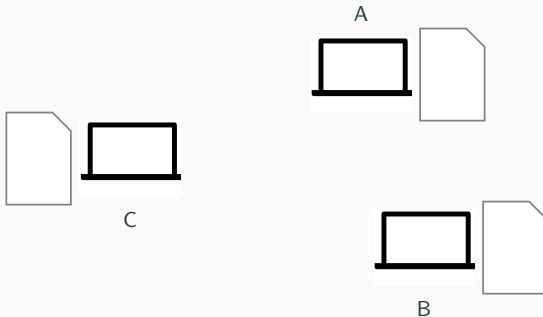


# Réplication dans applications collaboratives pair-à-pair



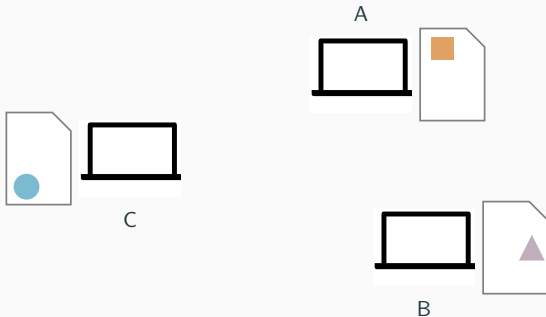
- Noeuds peuvent être **déconnectés**

# Réplication dans applications collaboratives pair-à-pair



- Noeuds peuvent être **déconnectés**
- Doivent pouvoir **travailler sans coordination synchrone** préalable (par ex. consensus)

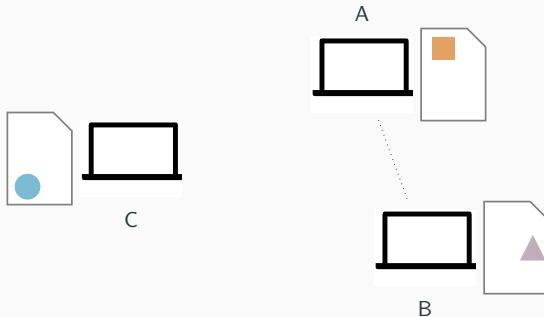
# Réplication dans applications collaboratives pair-à-pair



- Noeuds peuvent être **déconnectés**
- Doivent pouvoir **travailler sans coordination synchrone** préalable (par ex. consensus)

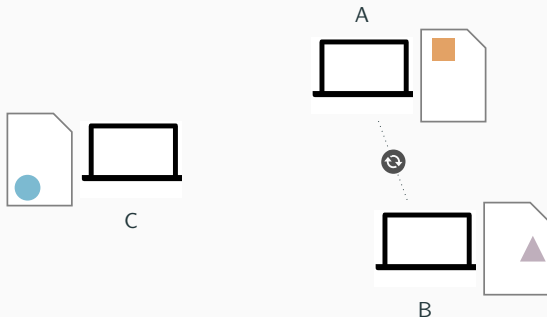


# Réplication dans applications collaboratives pair-à-pair



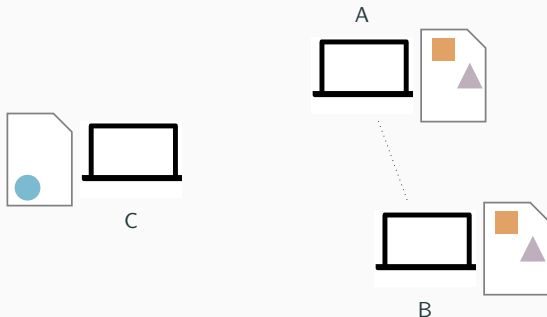
- Noeuds peuvent être **déconnectés**
- Doivent pouvoir **travailler sans coordination synchrone** préalable (par ex. consensus)

# Réplication dans applications collaboratives pair-à-pair



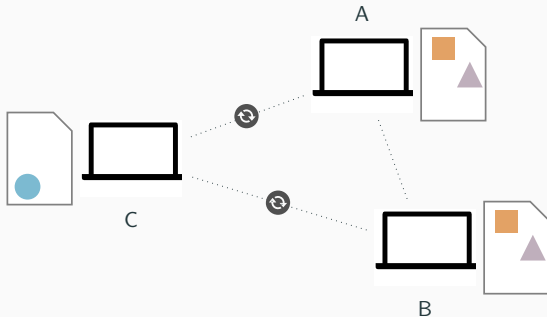
- Noeuds peuvent être **déconnectés**
- Doivent pouvoir **travailler sans coordination synchrone** préalable (par ex. consensus)

# Réplication dans applications collaboratives pair-à-pair



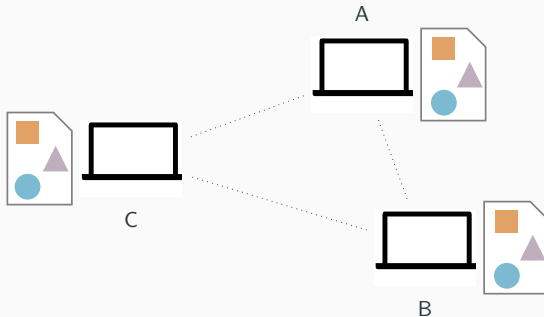
- Noeuds peuvent être **déconnectés**
- Doivent pouvoir **travailler sans coordination synchrone** préalable (par ex. consensus)

# Réplication dans applications collaboratives pair-à-pair



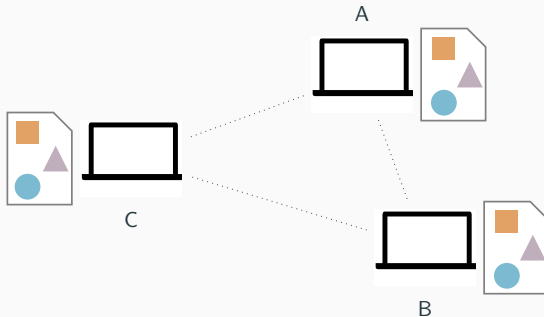
- Noeuds peuvent être **déconnectés**
- Doivent pouvoir **travailler sans coordination synchrone** préalable (par ex. consensus)

# Réplication dans applications collaboratives pair-à-pair



- Noeuds peuvent être **déconnectés**
- Doivent pouvoir **travailler sans coordination synchrone** préalable (par ex. consensus)
- Doit garantir **cohérence à terme** [Ter+95]...
- ... malgré ordres différents d'intégration des modifications

# Réplication dans applications collaboratives pair-à-pair



- Noeuds peuvent être **déconnectés**
- Doivent pouvoir **travailler sans coordination synchrone** préalable (par ex. consensus)
- Doit garantir **cohérence à terme** [Ter+95]...
- ... malgré ordres différents d'intégration des modifications

*Nécessite des mécanismes de résolution de conflits*

# Conflict-free Replicated Data Types (CRDTs) [Sha+11]

- Nouvelles spécifications des types de données, e.g. *Ensemble* ou *Séquence*
- Incorpore nativement mécanisme de résolution de conflits

# Conflict-free Replicated Data Types (CRDTs) [Sha+11]

- Nouvelles spécifications des types de données, e.g. *Ensemble* ou *Séquence*
- Incorpore nativement mécanisme de résolution de conflits

## Propriétés des CRDTs

- Permettent modifications **sans coordination**
- Garantissent la **cohérence forte à terme**



# Conflict-free Replicated Data Types (CRDTs) [Sha+11]

- Nouvelles spécifications des types de données, e.g. *Ensemble* ou *Séquence*
- Incorpore nativement mécanisme de résolution de conflits

## Propriétés des CRDTs

- Permettent modifications **sans coordination**
- Garantissent la **cohérence forte à terme**

## Cohérence forte à terme

Ensemble des noeuds ayant intégrés le même ensemble de modifications obtient des états équivalents, **sans nécessiter d'actions ou messages supplémentaires**

# CRDTs pour le type Séquence

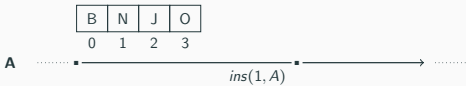
## Type Séquence usuel

B	N	J	O
0	1	2	3

A ..... ■—————→ .....

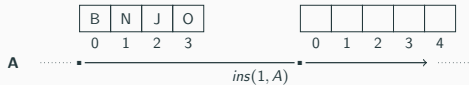
# CRDTs pour le type Séquence

## Type Séquence usuel



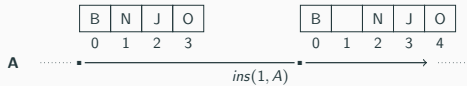
# CRDTs pour le type Séquence

## Type Séquence usuel



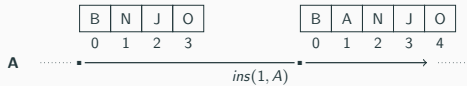
# CRDTs pour le type Séquence

## Type Séquence usuel



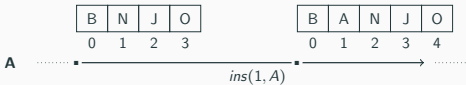
# CRDTs pour le type Séquence

## Type Séquence usuel



# CRDTs pour le type Séquence

## Type Séquence usuel



- Changements d'indices sont **source de conflits**

# CRDTs pour le type Séquence

## Type Séquence usuel

B	N	J	O
0	1	2	3

B	A	N	J	O
0	1	2	3	4

A ..... ■ —————→ .....  
 $ins(1, A)$

## CRDTs pour Séquence

B	N	J	O
$id_0$	$id_1$	$id_2$	$id_3$

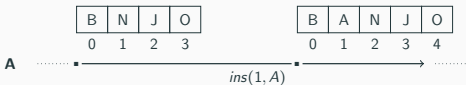
A ..... ■ —————→ .....

- Changements d'indices sont **source de conflits**
- CRDTs assignent des **identifiants de position** [Pre+09] à chaque élément
- Identifiants permettent d'**ordonner les éléments**



# CRDTs pour le type Séquence

## Type Séquence usuel



## CRDTs pour Séquence

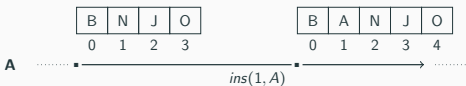


- Changements d'indices sont **source de conflits**
- CRDTs assignent des **identifiants de position** [Pre+09] à chaque élément
- Identifiants permettent d'**ordonner les éléments**

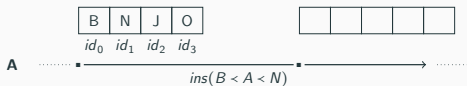
$$id_0 <_{id} id_1 <_{id} id_2 <_{id} id_3$$

# CRDTs pour le type Séquence

## Type Séquence usuel



## CRDTs pour Séquence

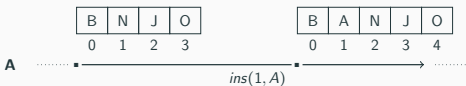


- Changements d'indices sont **source de conflits**
- CRDTs assignent des **identifiants de position** [Pre+09] à chaque élément
- Identifiants permettent d'**ordonner les éléments**

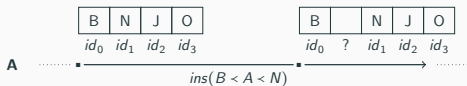
$$id_0 <_{id} id_1 <_{id} id_2 <_{id} id_3$$

# CRDTs pour le type Séquence

## Type Séquence usuel



## CRDTs pour Séquence



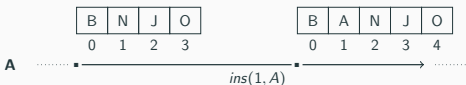
- Changements d'indices sont **source de conflits**
- CRDTs assignent des **identifiants de position** [Pre+09] à chaque élément
- Identifiants permettent d'**ordonner les éléments**

$$id_0 <_{id} id_1 <_{id} id_2 <_{id} id_3$$

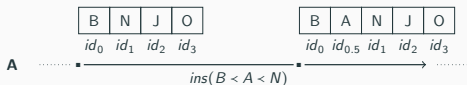
- Identifiants appartiennent à un **espace dense**

# CRDTs pour le type Séquence

## Type Séquence usuel



## CRDTs pour Séquence



- Changements d'indices sont **source de conflits**
- CRDTs assignent des **identifiants de position** [Pre+09] à chaque élément
- Identifiants permettent d'**ordonner les éléments**

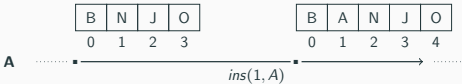
$$id_0 <_{id} id_1 <_{id} id_2 <_{id} id_3$$

- Identifiants appartiennent à un **espace dense**

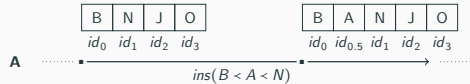
$$id_0 <_{id} id_{0.5} <_{id} id_1$$

# CRDTs pour le type Séquence

## Type Séquence usuel



## CRDTs pour Séquence



- Changements d'indices sont **source de conflits**
- CRDTs assignent des **identifiants de position** [Pre+09] à chaque élément
- Identifiants permettent d'**ordonner les éléments**

$$id_0 <_{id} id_1 <_{id} id_2 <_{id} id_3$$

- Identifiants appartiennent à un **espace dense**

$$id_0 <_{id} id_{0.5} <_{id} id_1$$

Utilise LogootSplit [And+13] comme base

# Identifiant LogootSplit

## Identifiant

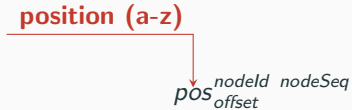
- Composé d'un ou plusieurs tuples de la forme

$pos_{offset}^{nodeId \ nodeSeq}$

# Identifiant LogootSplit

## Identifiant

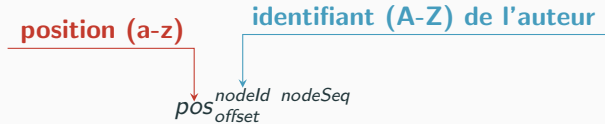
- Composé d'un ou plusieurs tuples de la forme



# Identifiant LogootSplit

## Identifiant

- Composé d'un ou plusieurs tuples de la forme

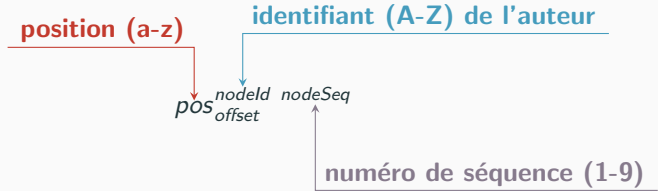




# Identifiant LogootSplit

## Identifiant

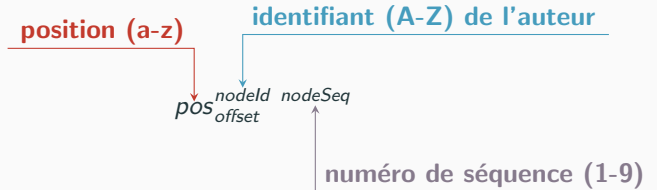
- Composé d'un ou plusieurs tuples de la forme



# Identifiant LogootSplit

## Identifiant

- Composé d'un ou plusieurs tuples de la forme



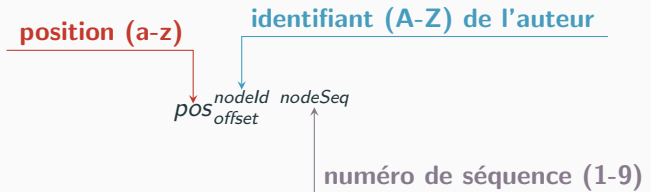
## Relation d'ordre $<_{id}$

- Se base sur l'ordre lexicographique sur les éléments des tuples

# Identifiant LogootSplit

## Identifiant

- Composé d'un ou plusieurs tuples de la forme



## Relation d'ordre $<_{id}$

- Se base sur l'ordre lexicographique sur les éléments des tuples

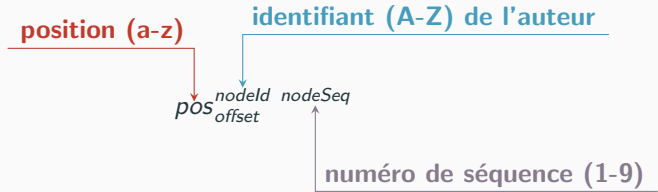
## Exemples

$d_0^{F5}$

# Identifiant LogootSplit

## Identifiant

- Composé d'un ou plusieurs tuples de la forme



## Relation d'ordre $<_{id}$

- Se base sur l'ordre lexicographique sur les éléments des tuples

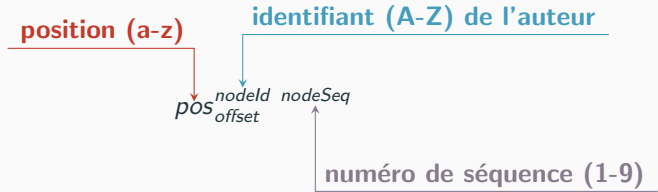
## Exemples

$$d_0^{F5} <_{id} m_0^{C1}$$

# Identifiant LogootSplit

## Identifiant

- Composé d'un ou plusieurs tuples de la forme



## Relation d'ordre $<_{id}$

- Se base sur l'ordre lexicographique sur les éléments des tuples

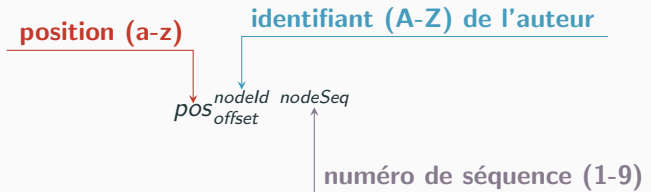
## Exemples

$$\mathbf{d}_0^{F5} <_{id} \mathbf{m}_0^{C1}$$

# Identifiant LogootSplit

## Identifiant

- Composé d'un ou plusieurs tuples de la forme



## Relation d'ordre $<_{id}$

- Se base sur l'ordre lexicographique sur les éléments des tuples

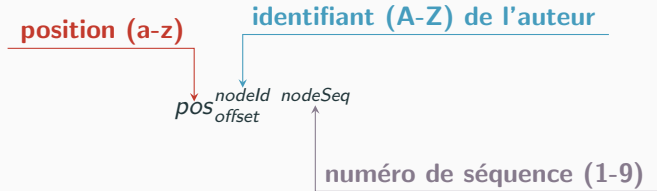
## Exemples

$$d_0^{F5} <_{id} m_0^{C1} <_{id} m_1^{C1}$$

# Identifiant LogootSplit

## Identifiant

- Composé d'un ou plusieurs tuples de la forme



## Relation d'ordre $<_{id}$

- Se base sur l'ordre lexicographique sur les éléments des tuples

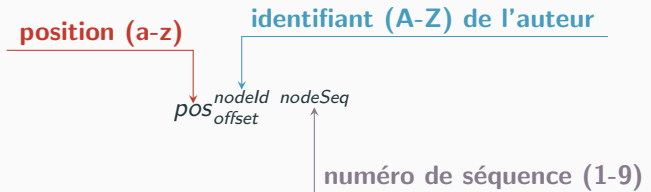
## Exemples

$$d_0^{F5} <_{id} m_0^{C1} <_{id} m_1^{C1}$$

# Identifiant LogootSplit

## Identifiant

- Composé d'un ou plusieurs tuples de la forme



## Relation d'ordre $<_{id}$

- Se base sur l'ordre lexicographique sur les éléments des tuples

## Exemples

$$d_0^{F5} <_{id} m_0^{C1} <_{id} m_1^{C1}$$

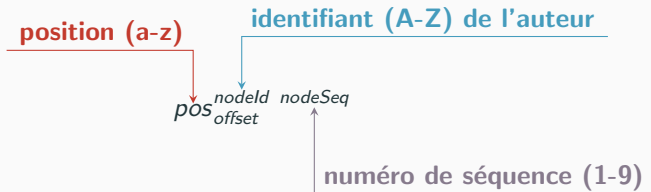
$$i_0^{B1} <_{id} ? <_{id} i_1^{B1}$$



# Identifiant LogootSplit

## Identifiant

- Composé d'un ou plusieurs tuples de la forme



## Relation d'ordre $<_{id}$

- Se base sur l'ordre lexicographique sur les éléments des tuples

## Exemples

$$d_0^{F5} <_{id} m_0^{C1} <_{id} m_1^{C1}$$

$$i_0^{B1} <_{id} i_0^{B1} f_0^{A1} <_{id} i_1^{B1}$$

- Coûteux de stocker les identifiants de chaque élément

B	A	N	J	O
$m_0^{C1}$	$m_1^{C1}$	$m_2^{C1}$	$m_3^{C1}$	$m_4^{C1}$

# Bloc LogootSplit

- Coûteux de stocker les identifiants de chaque élément

B	A	N	J	O
$m_0^{C1}$	$m_1^{C1}$	$m_2^{C1}$	$m_3^{C1}$	$m_4^{C1}$

- Aggrège en un **bloc** les éléments ayant des **identifiants contigus**

## Identifiants contigus

Deux identifiants sont contigus si et seulement si :

1. les deux identifiants sont identiques à l'exception de leur dernier offset
2. ces deux derniers offsets sont consécutifs

# Bloc LogootSplit

- Coûteux de stocker les identifiants de chaque élément

B	A	N	J	O
$m_0^{C1}$	$m_1^{C1}$	$m_2^{C1}$	$m_3^{C1}$	$m_4^{C1}$

- Aggrège en un **bloc** les éléments ayant des **identifiants contigus**

## Identifiants contigus

Deux identifiants sont contigus si et seulement si :

1. les deux identifiants sont identiques à l'exception de leur dernier offset
2. ces deux derniers offsets sont consécutifs

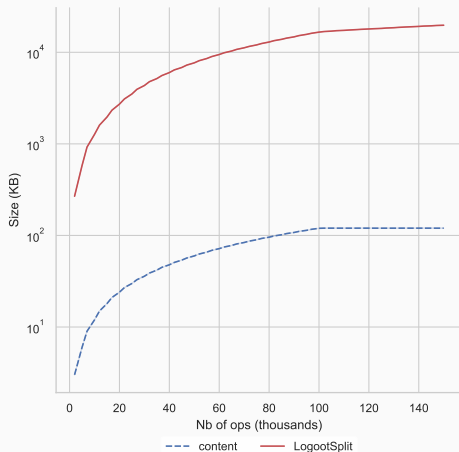
- Note l'intervalle d'identifiants d'un bloc :  $pos_{begin..end}^{nodeId\ nodeSeq}$

BANJO
-------

$m_{0..4}^{C1}$

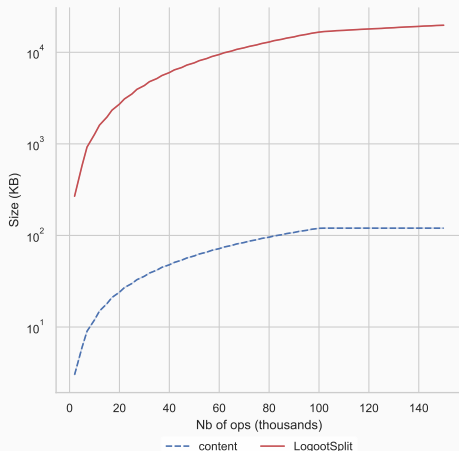
# Limites de LogootSplit

## Taille du contenu comparée à la taille de la séquence LogootSplit



# Limites de LogootSplit

## Taille du contenu comparée à la taille de la séquence LogootSplit

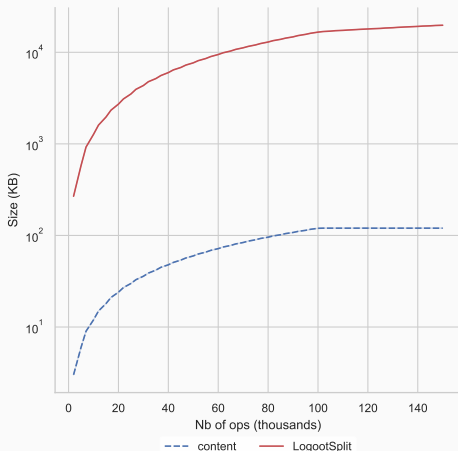


### Constat

- 1% contenu...
- ... 99% métadonnées

# Limites de LogootSplit

## Taille du contenu comparée à la taille de la séquence LogootSplit



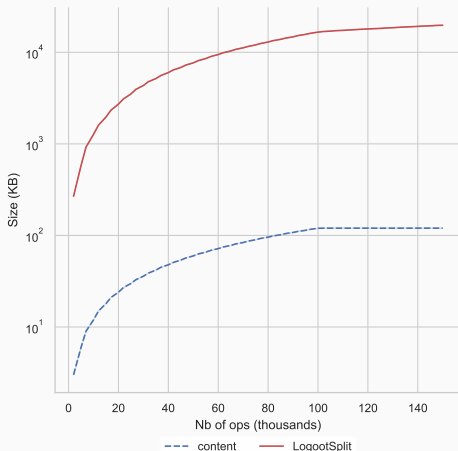
### Constat

- 1% contenu...
- ... 99% métadonnées

Et ça augmente !

# Limites de LogootSplit

## Taille du contenu comparée à la taille de la séquence LogootSplit



### Constat

- 1% contenu...
- ... 99% métadonnées

Et ça augmente !

### Impact

- Surcoût **mémoire**...
- ... mais aussi surcoût en **calculs** et en **bande-passante**



Comment réduire le surcoût des CRDTs pour  
le type Séquence, dans le cadre de systèmes  
pair-à-pair ?

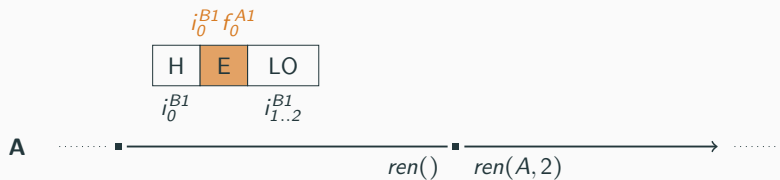
# Contribution : RenamableLogootSplit

- CRDT pour le type Séquence qui incorpore un mécanisme de renommage
- Réassigne de nouveaux identifiants aux éléments via une nouvelle opération : *rename*

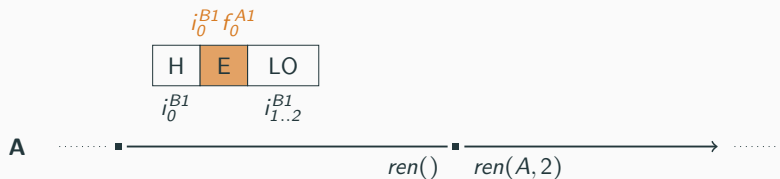
## Propriétés de l'opération *rename*

- Est déterministe
- Préserve l'intention des utilisateur-rices
- Préserve les propriétés de la séquence, c.-à-d. l'unicité et l'ordre de ses identifiants
- Est commutative avec les opérations *insert*, *remove* mais aussi *rename* concurrentes

# Opération rename

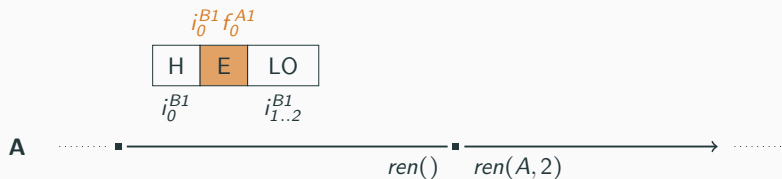


# Opération rename



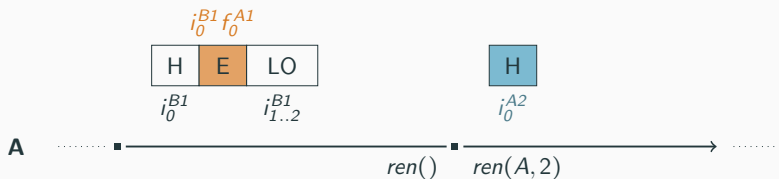
- Génère nouvel identifiant pour le 1er élément :

# Opération rename



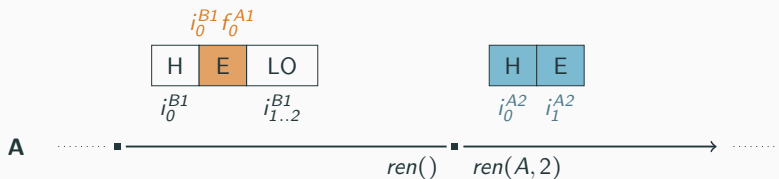
- Génère nouvel identifiant pour le 1er élément :  $i_0^{B1} \rightarrow i_0^{A2}$

# Opération rename



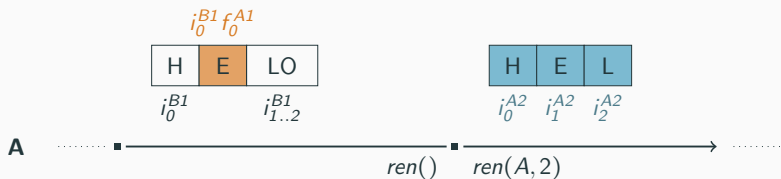
- Génère nouvel identifiant pour le 1er élément :  $i_0^{B1} \rightarrow i_0^{A2}$
- Puis génère identifiants contigus pour éléments suivants :

# Opération rename



- Génère nouvel identifiant pour le 1er élément :  $i_0^{B1} \rightarrow i_0^{A2}$
- Puis génère identifiants contigus pour éléments suivants :  $i_1^{A2}$

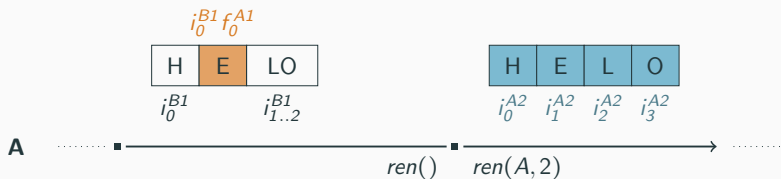
# Opération rename



- Génère nouvel identifiant pour le 1er élément :  $i_0^{B1} \rightarrow i_0^{A2}$
- Puis génère identifiants contigus pour éléments suivants :  $i_1^{A2}, i_2^{A2}$

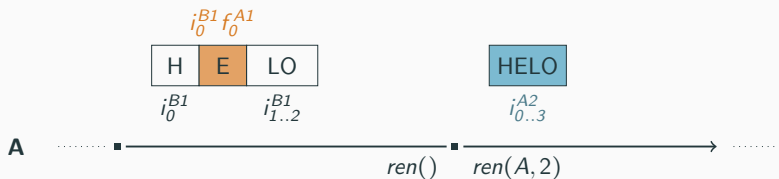


# Opération rename



- Génère nouvel identifiant pour le 1er élément :  $i_0^{B1} \rightarrow i_0^{A2}$
- Puis génère identifiants contigus pour éléments suivants :  $i_1^{A2}$ ,  $i_2^{A2}$ ,  
...

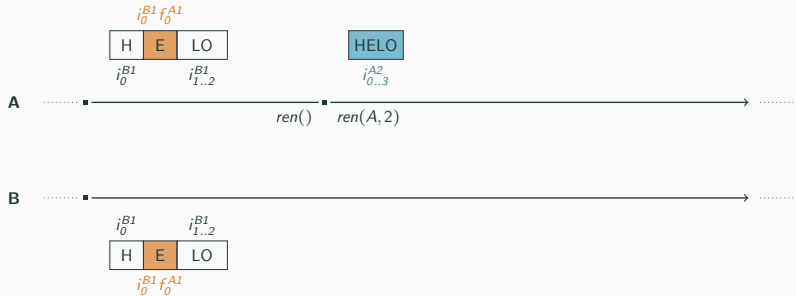
# Opération rename



- Génère nouvel identifiant pour le 1er élément :  $i_0^{B1} \rightarrow i_0^{A2}$
- Puis génère identifiants contigus pour éléments suivants :  $i_1^{A2}$ ,  $i_2^{A2}$ ,  
...

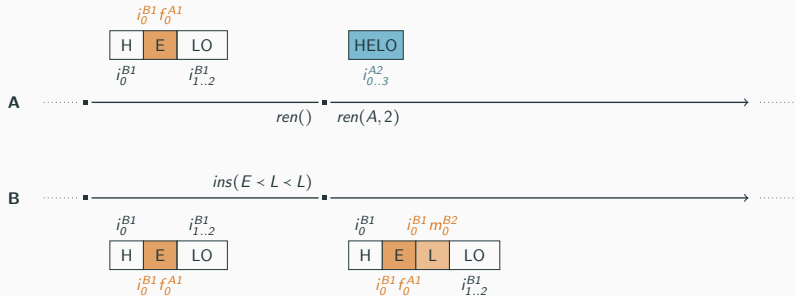
Regroupe tous les éléments en 1 unique bloc

# Interactions avec opérations insert et remove concurrentes



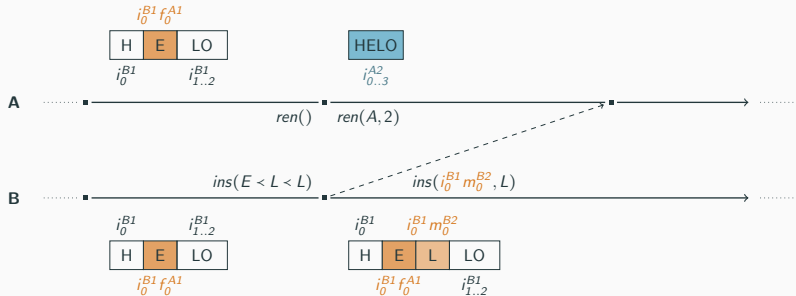
- Noeuds peuvent générer opérations concurrentes aux opérations *rename*

# Interactions avec opérations insert et remove concurrentes



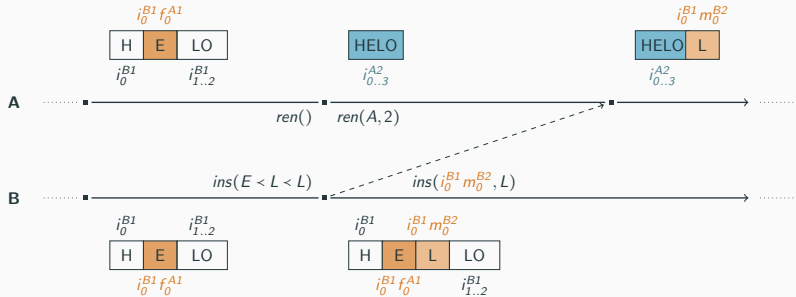
- Noeuds peuvent générer opérations concurrentes aux opérations *rename*

# Interactions avec opérations insert et remove concurrentes



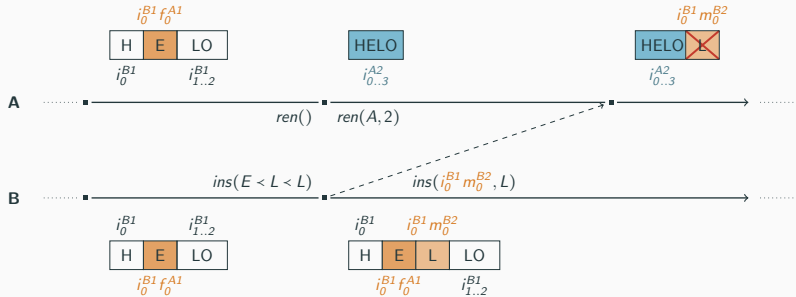
- Noeuds peuvent générer opérations concurrentes aux opérations *rename*

# Interactions avec opérations insert et remove concurrentes



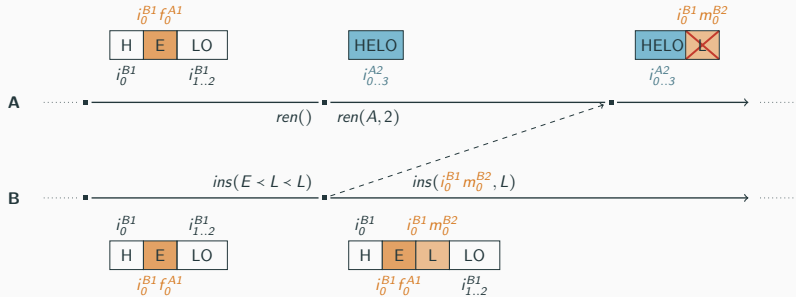
- Noeuds peuvent générer opérations concurrentes aux opérations *rename*

# Interactions avec opérations insert et remove concurrentes



- Noeuds peuvent générer opérations concurrentes aux opérations *rename*
- Opérations produisent anomalies si intégrées naïvement

# Interactions avec opérations insert et remove concurrentes



- Noeuds peuvent générer opérations concurrentes aux opérations *rename*
- Opérations produisent anomalies si intégrées naïvement

Nécessité d'un mécanisme dédié



# Décomposition de la contribution

## Besoins

1. Détecter les opérations concurrents aux opérations *rename*
2. Prendre en compte les effets des opérations *rename* lors de l'intégration des opérations concurrentes

## Besoins

1. Détecter les opérations concurrents aux opérations *rename*
2. Prendre en compte les effets des opérations *rename* lors de l'intégration des opérations concurrentes
3. Résoudre les conflits provoqués par des opérations *rename* concurrentes

## Besoins

1. Détecter les opérations concurrents aux opérations *rename*
2. Prendre en compte les effets des opérations *rename* lors de l'intégration des opérations concurrentes
3. Résoudre les conflits provoqués par des opérations *rename* concurrentes
4. Supprimer les métadonnées introduites par le mécanisme de renommage lui-même

# Décomposition de la contribution

## Besoins

1. Détecter les opérations concurrents aux opérations *rename*
2. Prendre en compte les effets des opérations *rename* lors de l'intégration des opérations concurrentes
3. Résoudre les conflits provoqués par des opérations *rename* concurrentes
4. Supprimer les métadonnées introduites par le mécanisme de renommage lui-même

- Implémentation disponible à l'adresse suivante :  
<https://github.com/coast-team/mute-structs>

- Montrer que RenamableLogootSplit satisfait la cohérence forte à terme
- Montrer que le mécanisme de renommage améliore les performances de la séquence répliquée (mémoire, calculs, bande-passante)

- Montrer que RenamableLogootSplit satisfait la cohérence forte à terme
- Montrer que le mécanisme de renommage améliore les performances de la séquence répliquée (mémoire, calculs, bande-passante)

Conduite d'une évaluation expérimentale

**Absence d'un jeu de données de sessions  
d'édition collaborative**

**Absence d'un jeu de données de sessions  
d'édition collaborative**

**Mise en place de simulations pour générer un  
jeu de données**



# Simulations - Principe

- Simulation d'une session d'édition collaborative
- 10 noeuds communiquant via un réseau entièrement maillé
- Noeuds effectuent modifications périodiquement et intègrent modifications reçues dès que possible, sans coordination
- 2 phases : génération de contenu (80% d'*insert*, 20% de *remove*) puis édition (50/50%)

- **Instantané de l'état** de chaque noeud à différents points de la simulation (10k opérations et état final)
- **Journal des opérations** de chaque noeud

---

\*. Code des simulations et benchmarks :

<https://github.com/coast-team/mute-bot-random>

- **Instantané de l'état** de chaque noeud à différents points de la simulation (10k opérations et état final)
- **Journal des opérations** de chaque noeud

## Conduite d'évaluations sur ces données<sup>\*</sup>

- Validation de l'amélioration des performances de la séquence répliquée (mémoire, calculs, bande-passante)

---

\*. Code des simulations et benchmarks :

<https://github.com/coast-team/mute-bot-random>

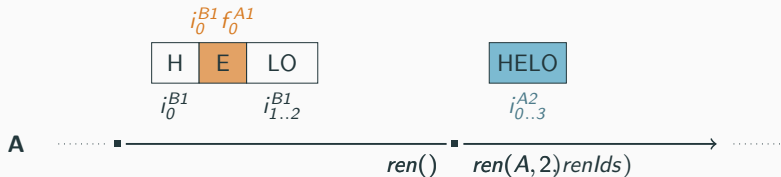
- **Article de position** : *Efficient renaming in CRDTs*, à Middleware 2018 - 19th ACM/IFIP International Middleware Conference (Doctoral Symposium), Dec 2018, Rennes, France.
- **Article d'atelier** : *Efficient renaming in Sequence CRDTs*, avec Gérald Oster et Olivier Perrin à PaPoC 2020 - 7th Workshop on Principles and Practice of Consistency for Distributed Data, Apr 2020, Heraklion / Virtual, Greece.
- **Article de revue** : *Efficient renaming in Sequence CRDTs*, avec Gérald Oster et Olivier Perrin dans IEEE Transactions on Parallel and Distributed Systems, Institute of Electrical and Electronics Engineers, 2022, 33 (12), pp.3870-3885.

- [Ter+95] Douglas B TERRY et al. « **Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System** ». In : *SIGOPS Oper. Syst. Rev.* 29.5 (déc. 1995), p. 172-182. ISSN : 0163-5980. DOI : 10.1145/224057.224070. URL : <https://doi.org/10.1145/224057.224070>.
- [Sha+11] Marc SHAPIRO et al. « **Conflict-Free Replicated Data Types** ». In : *Proceedings of the 13th International Symposium on Stabilization, Safety, and Security of Distributed Systems. SSS* 2011. 2011, p. 386-400. DOI : 10.1007/978-3-642-24550-3\_29.
- [Pre+09] Nuno PREGUICA et al. « **A Commutative Replicated Data Type for Cooperative Editing** ». In : *2009 29th IEEE International Conference on Distributed Computing Systems*. Juin 2009, p. 395-403. DOI : 10.1109/ICDCS.2009.20.

- [And+13] Luc ANDRÉ et al. « **Supporting Adaptable Granularity of Changes for Massive-Scale Collaborative Editing** ». In : *International Conference on Collaborative Computing: Networking, Applications and Worksharing - CollaborateCom 2013*. Austin, TX, USA : IEEE Computer Society, oct. 2013, p. 50-59. DOI : 10.4108/icst.collaboratecom.2013.254123.

**Back-up slides**

# Opération rename



- Génère nouvel identifiant pour le 1er élément :  $i_0^{B1} \rightarrow i_0^{A2}$
- Puis génère identifiants contigus pour éléments suivants :  $i_1^{A2}$ ,  $i_2^{A2}$ , ...

Regroupe tous les éléments en 1 unique bloc

Pour plus tard :

- Stocke identifiants ( $[i_0^{B1}, i_0^{B1} f_0^{A1}, \dots]$ ) de l'état d'origine :  $renlds$

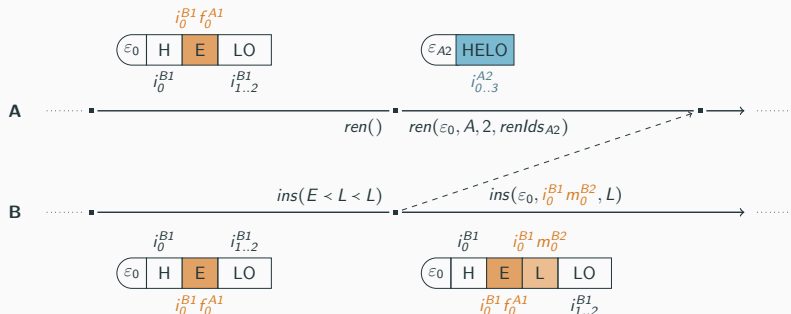


# Mécanisme de résolution de conflits entre une opération *rename* et une opération *insert* ou *remove*

## Besoins

1. Détecter les opérations concurrentes aux opérations *rename*
2. Prendre en compte effet des opérations *rename* lors de l'intégration des opérations concurrentes

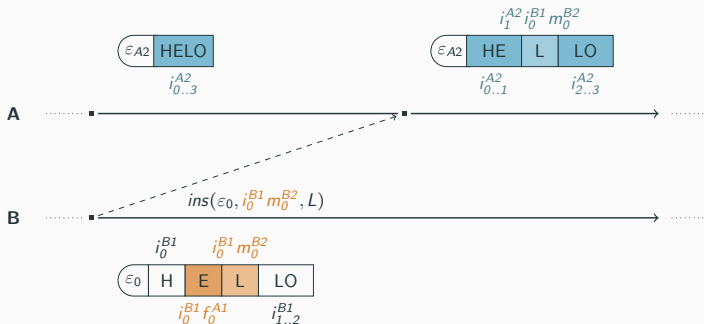
# Détection des opérations concurrentes à opération rename



## Ajout mécanisme d'époques

- Séquence commence à époque d'origine, notée  $\varepsilon_0$
- *rename* font progresser à nouvelle époque,  $\varepsilon_{nodeId} \text{ nodeSeq}$
- Opérations labellisées avec époque de génération

# Intégration des opérations insert et remove concurrentes

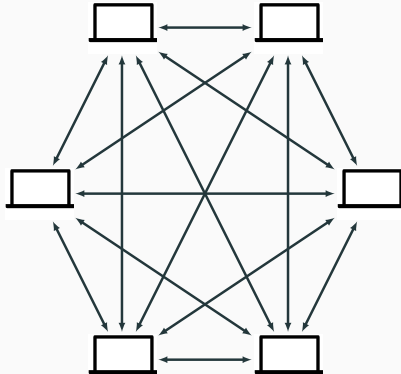


Rappel :

$$renlds_{A2} = [i_0^{B1}, i_0^{B1} f_0^{A1}, i_1^{B1}, i_2^{B1}]$$

Exemple avec  $i_0^{B1} m_0^{B2}$

- Rechercher son prédécesseur dans  $renlds_{A2}$  :  $i_0^{B1} f_0^{A1}$
- Utiliser son index (1) pour calculer équivalent à époque  $\epsilon_{A2}$  :  $i_1^{A2}$
- Préfixer  $i_0^{B1} m_0^{B2}$  par ce dernier :  $i_1^{A2} i_0^{B1} m_0^{B2}$



- 10 noeuds éditent collaborativement un document
- Topologie réseau entièrement maillée
- Ne considère pas de pannes ou de pertes de message

## Se décompose en 2 phases

1. **Génération du contenu** (80% d'*insert*, 20% de *remove*)
2. **Édition** (50/50%)

Noeuds passent à la phase 2 quand leur copie locale atteint une taille donnée (15 pages - 60k caractères)

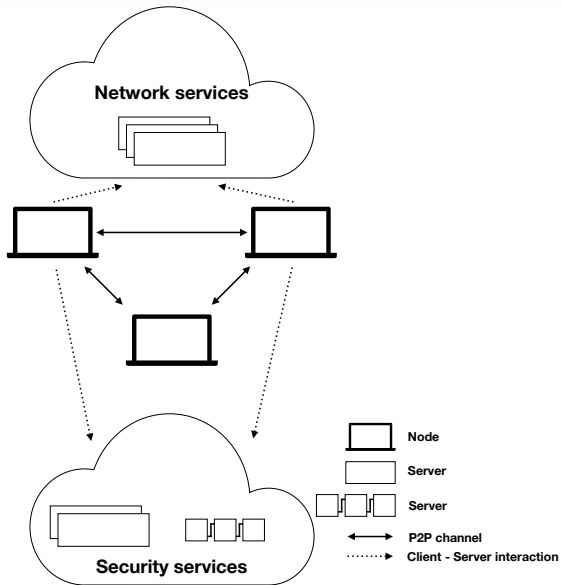
**Nombre d'opérations :** 15k par noeud, 150k au total

Noeuds utilisent LogootSplit (LS) ou RenamableLogootSplit (RLS)

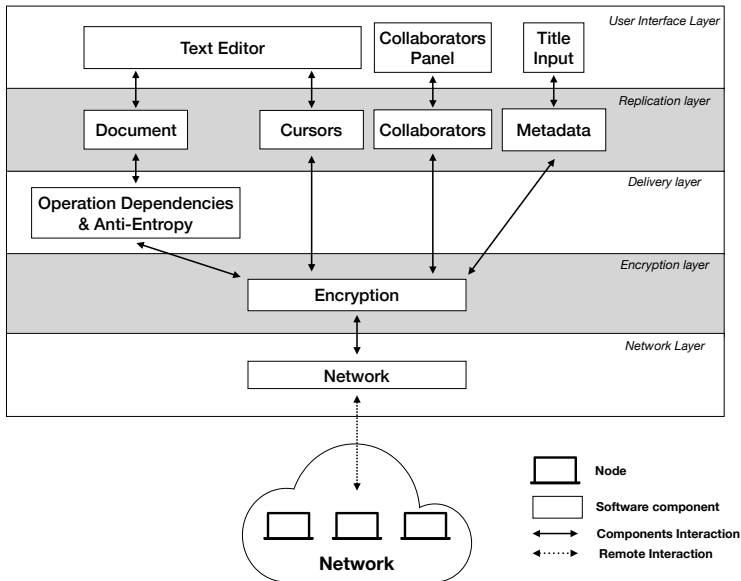
## Noeuds de renommage

- 1 à 4 noeuds effectuent une opération *rename* toutes les 30k opérations
- Opérations *rename* générées à un point donné sont concurrentes

# Architecture système de MUTE



# Architecture logicielle de MUTE





## Document

- Implémentation des CRDTs LogootSplit et RenamableLogootSplit

## Operation Dependancies & Anti-Entropy

- Implémentation des modèles de livraison pour LogootSplit et RenamableLogootSplit
- Implémentation d'un mécanisme d'anti-entropie (détection et échange des opérations perdues)

## Ingénierie logicielle

- Mise en place des processus d'intégration continue et de livraison continue pour les librairies `mute-structs`<sup>\*</sup> et `mute-core`<sup>\*</sup>

---

\*. <https://github.com/coast-team/mute-structs>

\*. <https://github.com/coast-team/mute-core>

### Network

- Supervision de la réalisation d'un *Proof of Concept* basé sur l'utilisation d'un *log-based message broker*

### Collaborators

- Supervision de l'adaptation et l'implémentation de SWIM, un protocole d'appartenance au réseau