

Entretien technique

Matthieu Nicolas (nicolasmatthieu57@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

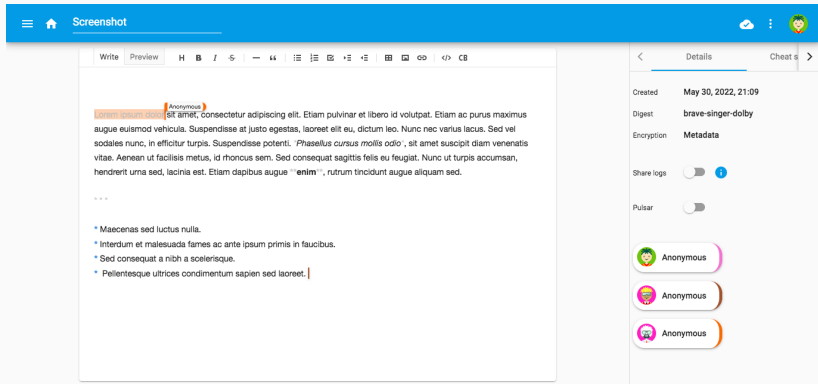
- MUTE : conception et développement d'un éditeur de texte collaboratif temps réel web pair-à-pair (<https://mute.loria.fr>)
- PLM : webification d'un environnement d'apprentissage de la programmation
(<http://people.irisa.fr/Martin.Quinson/Teaching/PLM/>)

Doctorat en informatique

2017-2022

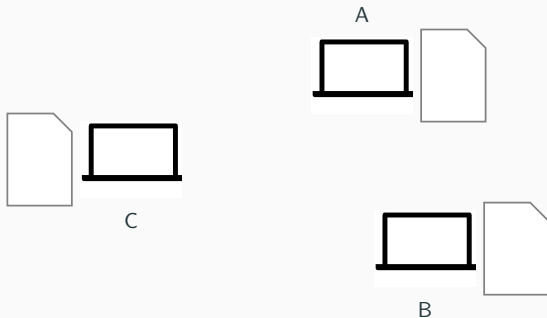
Université de Lorraine, Loria, équipe Coast, Nancy

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

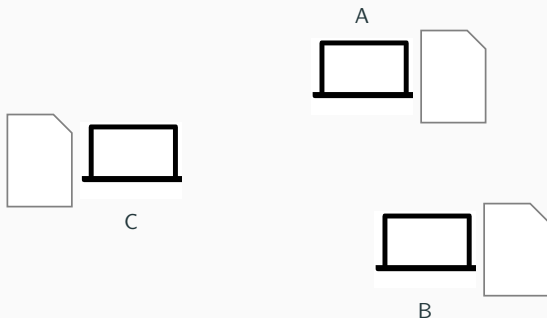


- Application pair-à-pair
- Permet de rédiger collaborativement des documents texte
- Garantit la confidentialité & souveraineté des données

Réplication dans applications collaboratives pair-à-pair

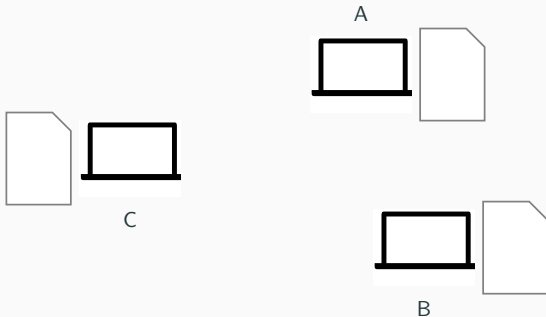


Réplication dans applications collaboratives pair-à-pair



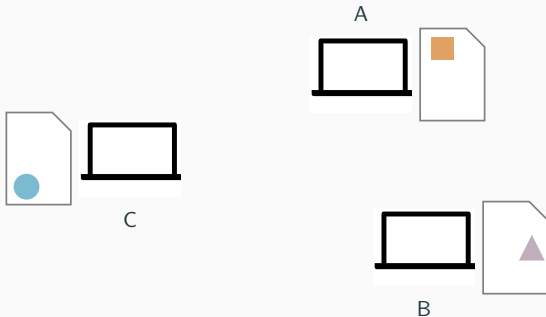
- Noeuds peuvent être **déconnectés**

Réplication dans applications collaboratives pair-à-pair



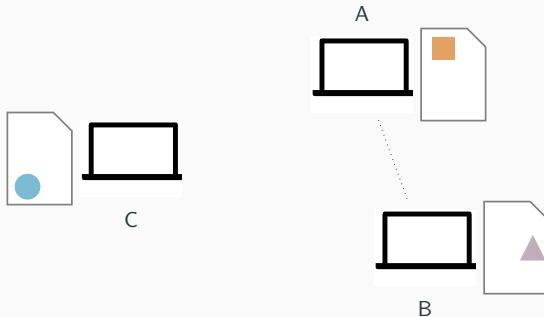
- Noeuds peuvent être **déconnectés**
- Doivent pouvoir **travailler sans coordination synchrone** préalable (par ex. consensus)

Réplication dans applications collaboratives pair-à-pair



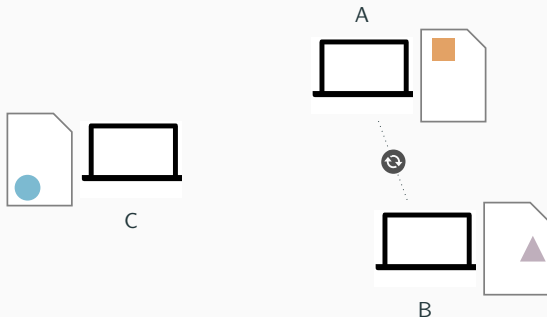
- Noeuds peuvent être **déconnectés**
- Doivent pouvoir **travailler sans coordination synchrone** préalable (par ex. consensus)

Réplication dans applications collaboratives pair-à-pair



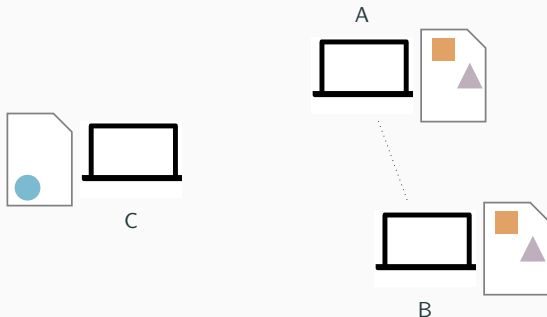
- Noeuds peuvent être **déconnectés**
- Doivent pouvoir **travailler sans coordination synchrone** préalable (par ex. consensus)

Réplication dans applications collaboratives pair-à-pair



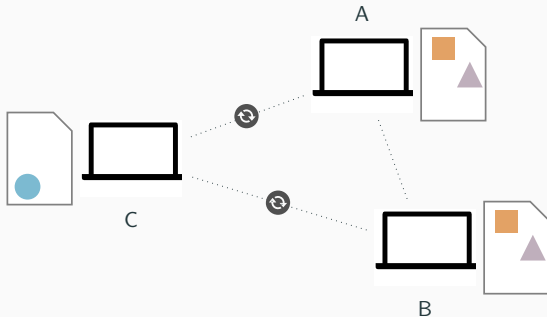
- Noeuds peuvent être **déconnectés**
- Doivent pouvoir **travailler sans coordination synchrone** préalable (par ex. consensus)

Réplication dans applications collaboratives pair-à-pair



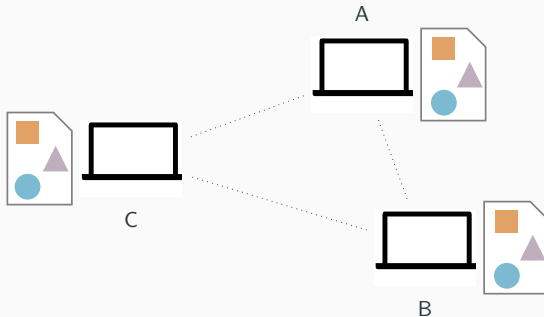
- Noeuds peuvent être **déconnectés**
- Doivent pouvoir **travailler sans coordination synchrone** préalable (par ex. consensus)

Réplication dans applications collaboratives pair-à-pair



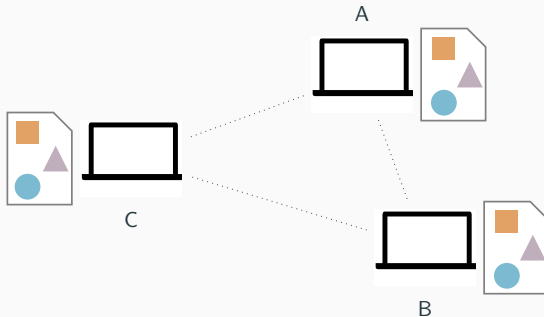
- Noeuds peuvent être **déconnectés**
- Doivent pouvoir **travailler sans coordination synchrone** préalable (par ex. consensus)

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

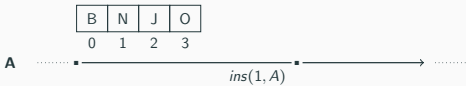
Type Séquence usuel

B	N	J	O
0	1	2	3

A ■—————→

CRDTs pour le type Séquence

Type Séquence usuel



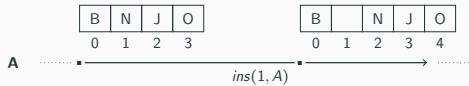
CRDTs pour le type Séquence

Type Séquence usuel



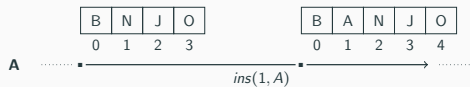
CRDTs pour le type Séquence

Type Séquence usuel



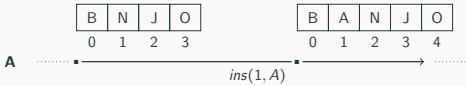
CRDTs pour le type Séquence

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

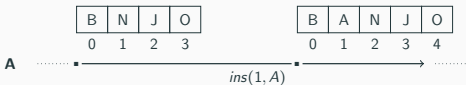
B	N	J	O
id_0	id_1	id_2	id_3

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**

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

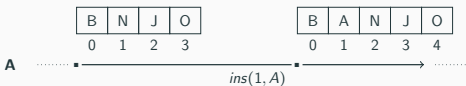


- Changements d'indices sont **source de conflits**
- CRDTs assignent des **identifiants de position** [Pre+09] à chaque élément
- Identifiants permettent d'**ordonner les éléments**

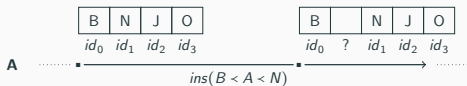
$$id_0 <_{id} id_1 <_{id} id_2 <_{id} id_3$$

CRDTs pour le type Séquence

Type Séquence usuel



CRDTs pour Séquence



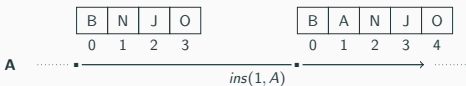
- Changements d'indices sont **source de conflits**
- CRDTs assignent des **identifiants de position** [Pre+09] à chaque élément
- Identifiants permettent d'**ordonner les éléments**

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

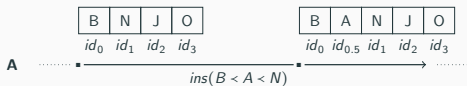
- Identifiants appartiennent à un **espace dense**

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

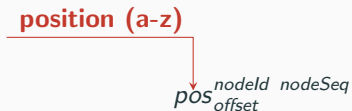
- Composé d'un ou plusieurs tuples de la forme

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

Identifiant LogootSplit

Identifiant

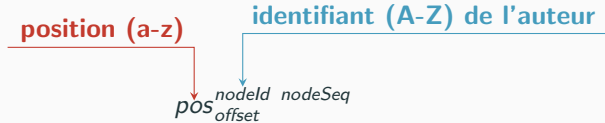
- Composé d'un ou plusieurs tuples de la forme



Identifiant LogootSplit

Identifiant

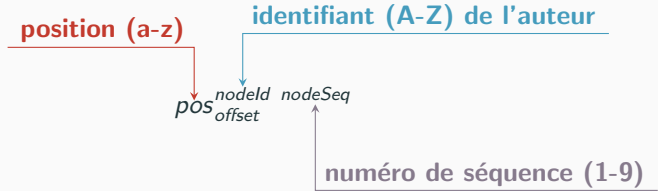
- Composé d'un ou plusieurs tuples de la forme



Identifiant LogootSplit

Identifiant

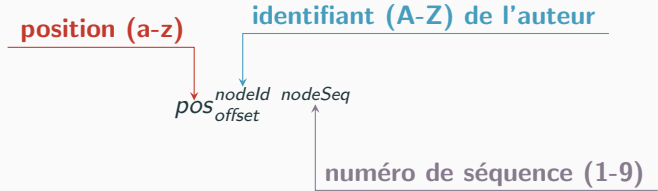
- Composé d'un ou plusieurs tuples de la forme



Identifiant LogootSplit

Identifiant

- Composé d'un ou plusieurs tuples de la forme



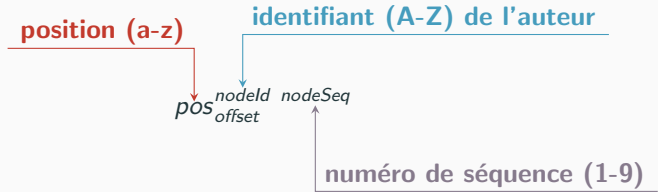
Relation d'ordre $<_{id}$

- 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

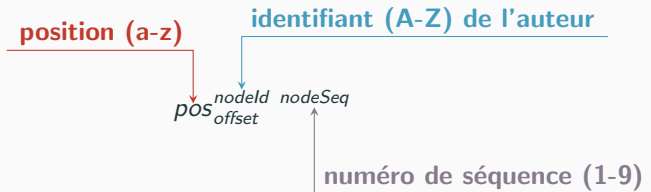
Exemples

$$d_0^{F5}$$

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

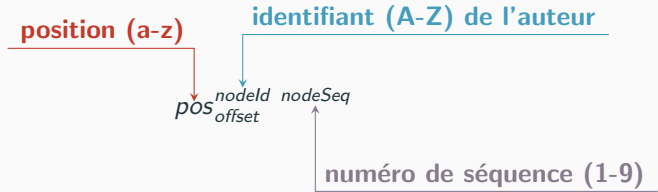
Exemples

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

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

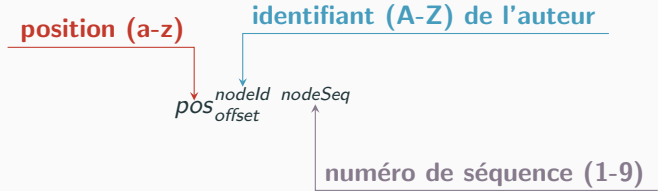
Exemples

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

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

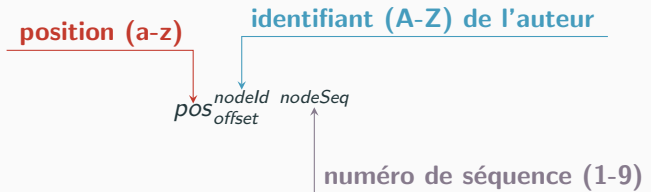
Exemples

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

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

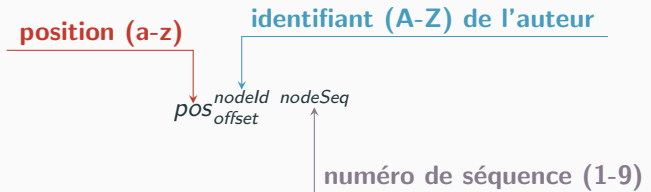
Exemples

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

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

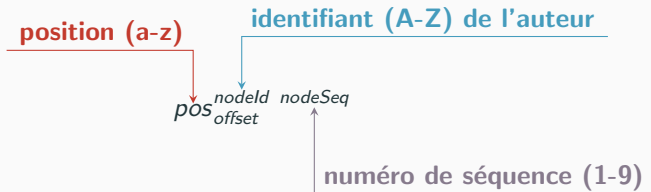
$$d_0^{F5} <_{id} m_0^{C1} <_{id} m_1^{C1}$$

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

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 :

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

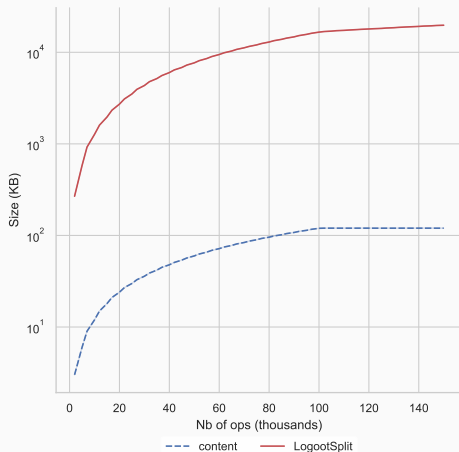
- Note l'intervalle d'identifiants d'un bloc : $pos_{begin..end}^{nodeId\ nodeSeq}$

BANJO

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

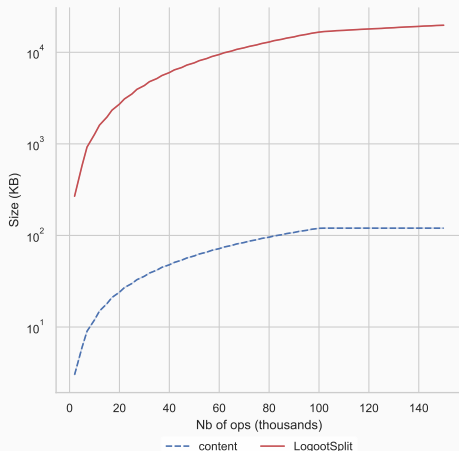
Limites de LogootSplit

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

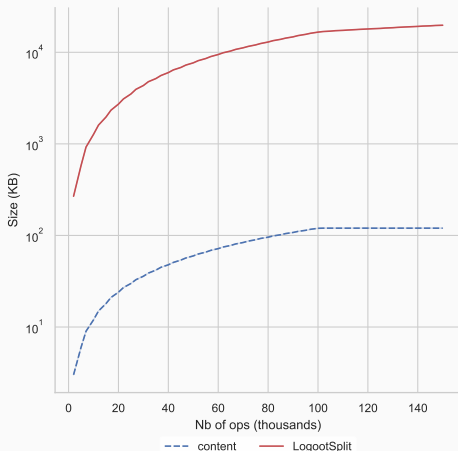


Constat

- 1% contenu...
- ... 99% métadonnées

Limites de LogootSplit

Taille du contenu comparée à la taille de la séquence LogootSplit



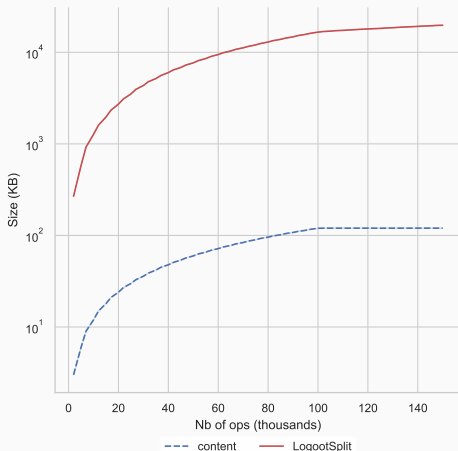
Constat

- 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

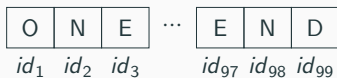
Et ça augmente !

Impact

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

Comment peut-on **réduire le surcoût** des
mécanismes de résolution de conflits dans les
applications pair-à-pair ?

Proposition



A ■ —————→

- Convertir l'état actuel...

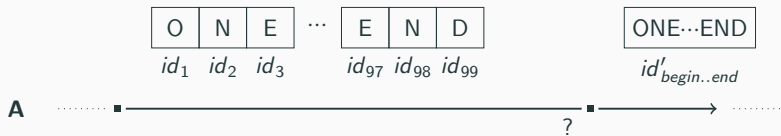
Proposition



A ■ —————→

- Convertir l'état actuel...
- ... en un état optimisé (identifiants de taille minimale, moins de blocs)...

Proposition



- Convertir l'état actuel...
- ... en un état optimisé (identifiants de taille minimale, moins de blocs)...
- ... à l'aide d'une nouvelle opération

Contribution : RenamableLogootSplit

- CRDT pour le type Séquence qui incorpore un mécanisme de renommage
- Prend la forme d'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

Décomposition de la contribution

- Mécanisme de renommage pour réassigner de nouveaux identifiants aux éléments

Décomposition de la contribution

- Mécanisme de renommage pour réassigner de nouveaux identifiants aux éléments
- Mécanisme de détection des modifications *insert* et *remove* concurrentes à une opération *rename* donnée
- Algorithme de transformation des modifications *insert* et *remove* concurrentes

Décomposition de la contribution

- Mécanisme de renommage pour réassigner de nouveaux identifiants aux éléments
- Mécanisme de détection des modifications *insert* et *remove* concurrentes à une opération *rename* donnée
- Algorithme de transformation des modifications *insert* et *remove* concurrentes
- Mécanisme de résolution de conflits en cas de *rename* concurrents

Décomposition de la contribution

- Mécanisme de renommage pour réassigner de nouveaux identifiants aux éléments
- Mécanisme de détection des modifications *insert* et *remove* concurrentes à une opération *rename* donnée
- Algorithme de transformation des modifications *insert* et *remove* concurrentes
- Mécanisme de résolution de conflits en cas de *rename* concurrents
- Mécanisme de *garbage collection* des métadonnées introduites par le mécanisme de renommage

Décomposition de la contribution

- Mécanisme de renommage pour réassigner de nouveaux identifiants aux éléments
- Mécanisme de détection des modifications *insert* et *remove* concurrentes à une opération *rename* donnée
- Algorithme de transformation des modifications *insert* et *remove* concurrentes
- Mécanisme de résolution de conflits en cas de *rename* concurrents
- Mécanisme de *garbage collection* des métadonnées introduites par le mécanisme de renommage
- 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)

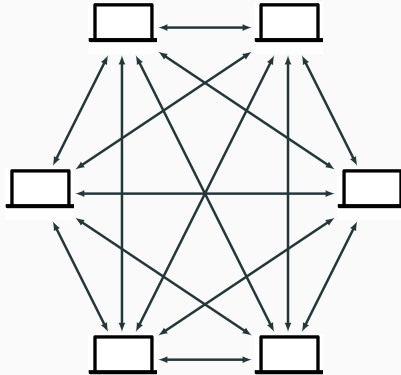
Conduite d'une évaluation expérimentale

**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 - Architecture



- 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)

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 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

- **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^{*}

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

*. Code des simulations et benchmarks :

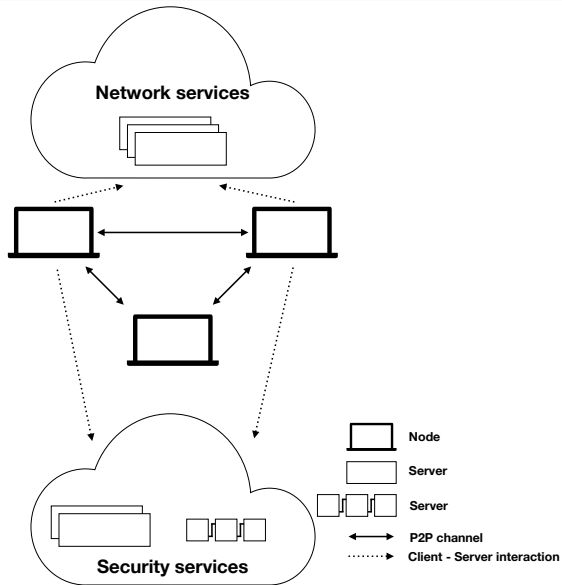
<https://github.com/coast-team/mute-bot-random>

- **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.

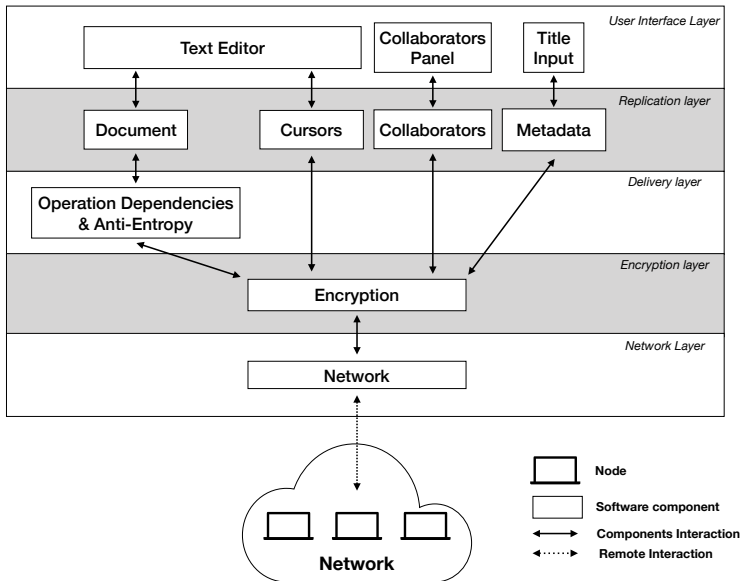
- [Ter+95] Douglas B TERRY et al. « **Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System** ». In : *SIGOPS Oper. Syst. Rev.* 29.5 (déc. 1995), p. 172-182. ISSN : 0163-5980. DOI : 10.1145/224057.224070. URL : <https://doi.org/10.1145/224057.224070>.
- [Sha+11] Marc SHAPIRO 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.

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`^{*} et `mute-core`^{*}

*. <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