UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Ré-identification sans coordination dans les types de données répliquées sans conflits (CRDTs)

THÈSE

présentée et soutenue publiquement le 16 Décembre 2022

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Matthieu Nicolas

Composition du jury

Président :	À déterminer	
Rapporteurs :	Hanifa Boucheneb	Professeure, Polytechnique Montréal
	Davide Frey	Chargé de recherche, HdR, Inria Rennes Bretagne-Atlantique
Examinateurs :	Hala Skaf-Molli	Maîtresse de conférences, HdR, Nantes Université, LS2N
	Stephan Merz	Directeur de Recherche, Inria Nancy - Grand Est
Encadrants :	Olivier Perrin	Professeur des Universités, Université de Lorraine, LORIA
	Gérald Oster	Maître de conférences, Université de Lorraine, LORIA

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Remerciements

WIP

WIP

Sommaire

Table des figures

Chapitre 1

Introduction

Sommaire

1.1 Contexte

L'évolution des technologies du web a conduit à l'avènement de ce qui est communément appelé le Web 2.0. La principale caractéristique de ce média est la possibilité aux utilisateur-rices non plus seulement de le consulter, mais aussi d'y contribuer.

Cette évolution a permis l'apparition d'applications incitant les utilisateur-rices à créer et partager leur propre contenu, ainsi que d'échanger avec d'autres utilisateur-rices à ce sujet. Un cas particulier de ces applications proposent aux utilisateur-rices de travailler ensemble pour la création d'un même contenu, en d'autres termes de collaborer. Nous appelons ces applications des *systèmes collaboratifs* :

Définition 1 (Système collaboratif)**.** Un système collaboratif est un système supportant ses utilisateur-rices dans leurs processus de collaboration pour la réalisation de tâches.

De nos jours, ces systèmes font parties des applications les plus populaires du paysage internet, e.g. la suite logicielle dont fait partie GoogleDocs compte 2 milliards d'utilisateur-rices [1], Wikipedia 788 millions [2], Quora 300 millions [3] ou encore GitHub 60 millions [4]. De leur côté, d'autres plateformes fédèrent leur communautés en organisant ponctuellement des collaborations éphémères impliquant des millions d'utilisateur-rices, e.g. r/Place [5] ou TwitchPlaysPokemon [6].

En raison de leur popularité, les systèmes collaboratifs doivent assurer plusieurs propriétés pour garantir leur bon fonctionnement et qualité de service : une haute disponibilité, tolérance aux pannes et capacité de passage à l'échelle.

Définition 2 (Disponibilité)**.** La disponibilité d'un système indique sa capacité à répondre à tout moment à une requête d'un-e utilisateur-rice.

Définition 3 (Tolérance aux pannes)**.** La tolérance aux pannes d'un système indique sa capacité à continuer à répondre aux requêtes malgré l'absence de réponse d'un ou plusieurs de ses composants.

Définition 4 (Capacité de passage à l'échelle)**.** La capacité de passage à l'échelle d'un système indique sa capacité à traiter un volume toujours plus conséquent de requêtes.

Pour cela, la plupart de ces systèmes adoptent une architecture décentralisée, que nous illustrons par la Figure 1.1. Dans cette figure, les noeuds aux extrêmités du graphe correspondent à des clients, les noeuds internes à des serveurs et les arêtes du graphe représentent les connexions entre appareils.

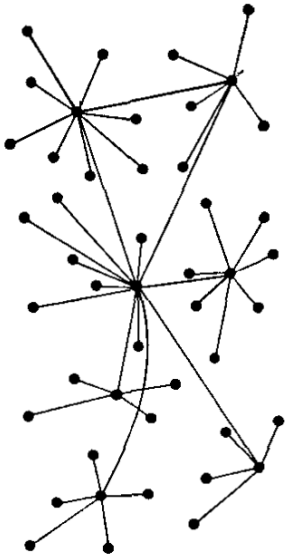

FIGURE 1.1 – Représentation d'une architecture décentralisée [7]

Dans ce type d'architecture, les responsabilités, tâches et la charge travail sont réparties entre un ensemble de serveurs. Malgré ce que le nom de cette architecture peut suggérer, il convient de noter que les serveurs jouent toujours un rôle central dans ces systèmes,. En effet, ces systèmes reposent toujours sur leurs serveurs pour authentifier les utilisateur-rices, stocker leurs données ou encore fusionner les modifications effectuées.

Bien que cette architecture système permette de répondre aux problèmes d'ordre technique que nous présentons précédemment, elle souffre néanmoins de limites. Notamment, de part le rôle prédominant que jouent les serveurs dans les systèmes décentralisés, ces derniers échouent à assurer un second ensemble de propriétés que nous jugeons néanmoins fondamentales :

Définition 5 (Confidentialité des données)**.** La confidentialité des données d'un système indique sa capacité à garantir à ses utilisateur-rices que leurs données ne seront pas accessibles par des tiers non autorisés ou par le système lui-même.

Définition 6 (Souveraineté des données)**.** La souveraineté des données d'un système indique sa capacité à garantir à ses utilisateur-rices leur maîtrise de leurs données, c.-à-d. leur capacité à les consulter, modifier, partager, exporter ; supprimer ou encore à décider de l'usage qui en est fait.

Définition 7 (Pérennité)**.** La pérennité d'un système indique sa capacité à garantir à ses utilisateur-rices son fonctionnement continu dans le temps.

Définition 8 (Résistance à la censure)**.** La résistance à la censure d'un système indique sa capacité à garantir à ses utilisateur-rices son fonctionnement malgré des actions de contrôle de l'information par des autorités.

De plus, les serveurs ne sont pas une ressource libre. En effet, ils sont déployés et maintenus par la ou les organisations qui proposent le système collaboratif. Ces organisations font alors office d'*autorités centrales* du système, e.g. en se portant garantes de l'identité des utilisateur-rices, de l'authenticité d'un contenu ou encore de la disponibilité dudit contenu.

De part le fait que les autorités centrales possèdent les serveurs hébergeant le système, elles ont tout pouvoir sur ces derniers. Ainsi, les utilisateur-rices de systèmes collaboratifs prennent, de manière consciente ou non, le risque que les propriétés présentées précédemment soient transgressées par les autorités auxquelles appartiennent ces applications ou par des tiers avec lesquelles ces autorités interagissent, e.g. des gouvernements. Plusieurs faits d'actualités nous ont malheureusement montré de tels faits, e.g. la censure de Wikipedia par des gouvernements [8], la fermeture de services par les entreprises les proposant [9] ou encore la mise à disposition des données hébergées par des applications aux services de renseignement de différentes nations [10, 11]. Cependant, le coût conséquent de l'infrastructure nécessaire pour déployer des systèmes à large échelle équivalents entrave la mise en place d'alternatives, plus respectueuses de leurs utilisateur-rices.

Ainsi, il nous paraît fondamental de proposer des moyens technologiques rendant accessible la conception et le déploiement des systèmes collaboratifs alternatifs. Ces derniers devraient minimiser le rôle des autorités centrales, voire l'éliminer, de façon à protéger et privilégier les intérêts de leurs utilisateur-rices.

Dans cette optique, une piste de recherche que nous jugeons intéressante est celle des systèmes collaboratifs pair-à-pair (P2P). Cette architecture système, que nous illustrons par la Figure 1.2, place les utilisateur-rices au centre du système et relègue les éventuels serveurs à un simple rôle de support de la collaboration, e.g. la mise en relation des pairs.

Récemment, la conception de systèmes collaboratifs P2P a gagné en traction suite à [12]. Dans cet article, les auteurs définissent un ensemble de propriétés qui correspondent à celles que nous avons établies précédemment, de la Définition 1 à la Définition 8. En

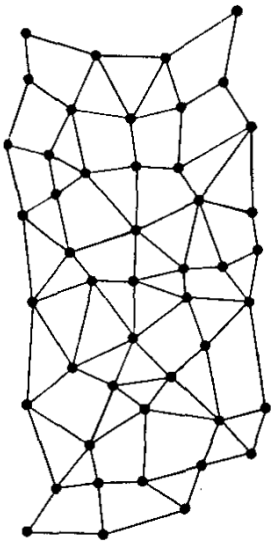

FIGURE 1.2 – Représentation d'une architecture distribuée [7]

utilisant ces propriétés comme critères, les auteurs comparent les fonctionnalités et garanties offertes par les différents types d'applications, notamment les applications lourdes et les applications basées sur le cloud.

La Figure 1.3 détaille le résultat de cette comparaison : alors que les applications basées sur le cloud permettent de nouveaux usages, notamment la collaboration entre utilisateur-rices ou la synchronisation automatique entre appareils, elles retirent à leurs utilisateur-rices toute garantie de pérennité, confidentialité des données et souveraineté des données. Ces dernières propriétés sont pourtant communément offertes par les applications lourdes.

Malgré ce que ce résultat pourrait suggérer, les auteurs affirment que les nouveaux usages offerts par les applications basées sur le cloud ne sont pas antinomiques avec les propriétés de confidentialité, souveraineté, pérennité.

Ainsi, ils proposent un nouveau paradigme de conception d'applications collaboratives P2P, nommées Local-First Software (LFS). Ce paradigme vise à la conception d'applications offrant le meilleur des approches existantes, c.-à-d. des applications cochant l'intégralité des critères de la Figure 1.3. Nous partageons cette vision.

Cependant, de nombreuses problématiques de recherche identifiées dans [12] sont encore non résolues et entravent la démocratisation des applications LFS, notamment celles à large échelle. Spécifiquement, les applications LFS se doivent de répliquer les données entre les appareils pour permettre :

(i) Le fonctionnement en mode hors-ligne et le fonctionnement avec une faible latence.

(ii) Le partage de contenu entre appareils d'un-e même utilisateur-rice.

(iii) Le partage de contenu entre utilisateur-rices pour la collaboration.

Toutefois, compte tenu des propriétés visées par les applications LFS, plusieurs contraintes restreignent le choix des méthodes de réplication possibles. Ainsi, pour permettre le fonc-

Technology	Section	1. Fast (§ 2.1)	2. Multi-device (§ 2.2)	3. Offline (§ 2.3)	4. Collaboration (§ 2.4)	5. Longevity (§ 2.5)	6. Privacy (§ 2.6)	7. User control (§ 2.7)
Applications employed by end users								
Files + email attachments	§ 3.1.1	✓	–	✓	●	✓	–	✓
Google Docs	§ 3.1.2	–	✓	–	✓	–	●	–
Trello	§ 3.1.2	–	✓	–	✓	–	●	●
Pinterest	§ 3.1.2	●	✓	●	✓	●	●	●
Dropbox	§ 3.1.3	✓	–	–	●	✓	–	✓
Git + GitHub	§ 3.1.4	✓	–	✓	–	✓	–	✓
Technologies employed by application developers								
Thin client (web apps)	§ 3.2.1	●	✓	●	✓	●	●	●
Thick client (mobile apps)	§ 3.2.2	✓	–	✓	●	–	●	●
Backend-as-a-service	§ 3.2.3	–	✓	✓	–	●	●	●
CouchDB	§ 3.2.4	–	–	✓	●	–	–	–

FIGURE 1.3 – Évaluation d'applications et de technologies vis-à-vis des 7 propriétés visées par les applications Local-First Softwares [12]. ✓, – et ● indiquent respectivement que l'application ou la technologie satisfait pleinement, partiellement ou aucunement le critère évalué.

tionnement en mode hors-ligne de l'application, c.-à-d. la consultation et la modification de contenu, les applications LFS doivent obligatoirement relaxer la propriété de cohérence des données.

Définition 9 (Cohérence)**.** La cohérence d'un système indique sa capacité à présenter une vue uniforme de son état à chacun de ses utilisateur-rices à un moment donné.

Ainsi, les applications LFS doivent autoriser les noeuds possédant une copie de la donnée à diverger temporairement, c.-à-d. à posséder des copies dans des états différents à un moment donné. Pour permettre cela, les applications LFS doivent adopter des méthodes de réplication dites optimistes [13], c.-à-d. qui permettent la consultation et la modification de la donnée sans coordination au préalable avec les autres noeuds [1]. Un mécanisme de synchronisation permet ensuite aux noeuds de partager les modifications effectuées et de les intégrer de façon à converger à terme [14], c.-à-d. obtenir de nouveau des états équivalents.

Il convient de noter que les méthodes de réplication optimistes autorisent la génération en concurrence de modifications provoquant un conflit, e.g. la modification et la

1. Les méthodes de réplication optimistes s'opposent aux méthodes de réplication dites pessimistes qui nécessitent une coordination préalable entre les noeuds avant toute modification de la donnée et interdisent ainsi toute divergence

suppression d'une même page dans un wiki. Un mécanisme de résolution de conflits est alors nécessaire pour assurer la convergence à terme des noeuds.

De nouveau, le modèle du système des applications que nous visons, c.-à-d. des applications LFS à large échelle, limitent les choix possibles concernant les mécanismes de résolution de conflits. Notamment, ces applications ne disposent d'aucun contrôle sur le nombre de noeuds qui compose le système, c.-à-d. le nombre d'appareils utilisés par l'ensemble de leurs utilisateur-rices. Ce nombre de noeuds peut croître de manière non-bornée. Les mécanismes de résolution de conflits choisis devraient donc rester efficaces, indépendamment de l'évolution de ce paramètre.

De plus, les noeuds composant le système n'offrent aucune garantie sur leur stabilité. Des noeuds peuvent donc rejoindre et participer au système, mais uniquement de manière éphèmère. Ce phénonème est connu sous le nom de *churn* [15]. Ainsi, de part l'absence de garantie sur le nombre de noeuds connectés de manière stable, les applications LFS à large échelle ne peuvent pas utiliser des mécanismes de résolution de conflits reposant sur une coordination synchrone d'une proportion des noeuds du système, c.-à-d. des mécanismes nécessitant une communication ininterrompue entre un ensemble de noeuds du système pour prendre une décision, e.g. des algorithmes de consensus [16, 17].

Ainsi, pour permettre la conception d'applications LFS à large échelle, il convient de disposer de mécanismes de résolution de conflits pour l'ensemble des types de données avec une complexité algorithmique efficace peu importe le nombre de noeuds et ne nécessitant pas de coordination synchrone entre une proportion des noeuds du système.

1.2 Questions de recherche et contributions

1.2.1 Ré-identification sans coordination synchrone pour les CRDTs pour le type Séquence

Les Conflict-free Replicated Data Types (CRDTs) [2] [18, 19] sont des nouvelles spécifications des types de données usuels, e.g. l'Ensemble ou la Séquence. Ils sont conçus pour permettre à un ensemble de noeuds d'un système de répliquer une donnée et pour leur permettre de la consulter, de la modifier sans aucune coordination préalable et d'assurer à terme la convergence des copies. Dans ce but, les CRDTs incorporent des mécanismes de résolution de conflits automatiques directement au sein leur spécification.

Cependant, ces mécanimes induisent un surcoût, aussi bien en termes de métadonnées et de calculs que de bande-passante. Ces surcoûts sont néanmoins jugés acceptables par la communauté pour une variété de types de données, e.g. le Registre ou l'Ensemble. Cependant, le surcoût des CRDTs pour le type Séquence constitue toujours une problématique de recherche.

En effet, la particuliarité des CRDTs pour le type Séquence est que leur surcoût croît de manière monotone au cours de la durée de vie de la donnée, c.-à-d. au fur et à mesure des modifications effectuées. Le surcoût introduit par les CRDTs pour ce type de données se révèle donc handicapant dans le contexte de collaborations sur de longues durées ou à large échelle.

2. Conflict-free Replicated Data Type (CRDT) : Type de données répliquées sans conflits

De manière plus précise, le surcoût des CRDTs pour le type Séquence provient de la croissance des métadonnées utilisées par leur mécanisme de résolution de conflits automatique. Ces métadonnées correspondent à des identifiants qui sont associés aux éléments de la Séquence. Ces identifiants permettent d'intégrer les modifications, e.g. en précisant quel est l'élement à supprimer ou en spécifiant la position d'un nouvel élément à insérer par rapport aux autres.

Plusieurs approches ont été proposées pour réduire le coût induit par ces identifiants. Notamment, [20, 21] proposent un mécanisme de ré-assignation des identifiants pour réduire leur coût a posteriori. Ce mécanisme génère toutefois des conflits en cas de modifications concurrentes de la séquence, c.-à-d. l'insertion ou la suppression d'un élément. Les auteurs résolvent ce problème en proposant un mécanisme de transformation des modifications concurrentes par rapport à l'effet du mécanisme de ré-assignation des identifiants.

Cependant, l'exécution en concurrence du mécanisme de ré-assignation des identifiants par plusieurs noeuds provoque elle-même un conflit. Pour éviter ce dernier type de conflit, les auteurs choisissent de subordonner à un algorithme de consensus l'exécution du mécanisme de ré-assignation des identifiants. Ainsi, le mécanisme de ré-assignation des identifiants ne peut être déclenché en concurrence par plusieurs noeuds du système.

Comme nous l'avons évoqué précédemment, reposer sur un algorithme de consensus qui requiert une coordination synchrone entre une proportion de noeuds du système est une contrainte incompatible avec les systèmes P2P à large échelle sujets au churn.

Notre problématique de recherche est donc la suivante : *pouvons-nous proposer un mécanisme sans coordination synchrone de réduction du surcoût des CRDTs pour Séquence, c.-à-d. adapté aux applications LFS ?*

Pour répondre à cette problématique, nous proposons dans cette thèse RenamableLogootSplit, un nouveau CRDT pour le type Séquence. Ce CRDT intègre un mécanisme de ré-assignation des identifiants, dit de renommage, directement au sein de sa spécification. Nous associons au mécanisme de renommage un mécanisme de résolution de conflits automatique additionnel pour gérer ses exécutions concurrentes. Finalement, nous définissons un mécanisme de Garbage Collection (GC) [3] des métadonnées du mécanisme de renommage pour supprimer à terme son propre surcoût. De cette manière, nous proposons un CRDT pour le type Séquence dont le surcoût est périodiquement réduit, tout en n'introduisant aucune contrainte de coordination synchrone entre les noeuds du système.

1.2.2 Éditeur de texte collaboratif P2P temps réel chiffré de bout en bout

Comme évoqué précédemment, la conception d'applications LFS à large échelle présente un ensemble de problématiques issues de domaines variés, e.g.

(i) Comment permettre aux utilisateur-rices de collaborer en l'absence d'autorités centrales pour résoudre les conflits de modifications ?

(ii) Comment authentifier les utilisateur-rices en l'absence d'autorités centrales ?

3. Garbage Collection (GC) : Récupération de la mémoire

(iii) Comment structurer le réseau de manière efficace, c.-à-d. en limitant le nombre de connexions par pair ?

Cet ensemble de questions peut être résumé en la problématique suivante : *pouvons-nous concevoir une application LFS à large échelle, sûre et sans autorités centrales ?*

Pour étudier cette problématique, l'équipe Coast développe l'application Multi User Text Editor (MUTE) [4] [22]. Il s'agit d'un Proof of Concept (PoC) [5] d'éditeur de texte web collaboratif P2P temps réel chiffré de bout en bout. Ce projet permet à l'équipe de présenter ses travaux de recherche portant sur les mécanismes de résolutions de conflits automatiques pour le type Séquence [23, 24, 25] et les mécanismes d'authentification des pairs dans les systèmes sans autorités centrales [26, 27].

De plus, en inscrivant ses travaux dans le cadre d'un système complet, ce projet permet à l'équipe d'identifier de nouvelles problématiques en relation avec les nombreux domaines de recherche nécessaires à la conception d'un tel système, e.g. le domaine des protocoles d'appartenance aux groupes [28, 29], des topologies réseaux P2P [30] ou encore des protocoles d'établissement de clés de chiffrement de groupe [31].

Dans le cadre de cette thèse, nous avons contribué au développement de ce projet. Nous avons notamment implémenté plusieurs CRDTs pour le type Séquence [23, 25] et le protocole d'appartenance au réseau SWIM [28].

1.3 Plan du manuscrit

Ce manuscrit de thèse est organisé de la manière suivante :

Dans le **??**, nous introduisons le modèle du système que nous considérons, c.-à-d. les systèmes P2P à large échelle sujets au churn et sans autorités centrales. Puis nous présentons dans ce chapitre l'état de l'art des CRDTs et plus particulièrement celui des CRDTs pour le type Séquence. À partir de cet état de l'art, nous identifions et motivons notre problématique de recherche, c.-à-d. l'absence de mécanisme adapté aux systèmes P2P à large échelle sujets au churn permettant de réduire le surcoût induit par les mécanismes de résolution de conflits automatiques pour le type Séquence.

Dans le **??**, nous présentons notre approche pour réaliser un tel mécanisme, c.-à-d. un mécanisme de résolution de conflits automatiques pour le type Séquence auquel nous associons un mécanisme de Garbage Collection (GC) de son surcoût ne nécessitant pas de coordination synchrone entre les noeuds du système. Nous détaillons le fonctionnement de notre approche, sa validation par le biais d'une évaluation empirique puis comparons notre approche par rapport aux approches existantes Finalement, nous concluons la présentation de notre approche en identifiant et en détaillant plusieurs de ses limites.

Dans le **??**, nous présentons MUTE, l'éditeur de texte collaboratif temps réel P2P chiffré de bout en bout que notre équipe de recherche développe dans le cadre de ses travaux de recherche. Nous présentons les différentes couches logicielles formant un pair

4. Disponible à l'adresse : `https://mutehost.loria.fr`
5. Proof of Concept (PoC) : Preuve de concept

et les services tiers avec lesquels les pairs interagissent, et détaillons nos travaux dans le cadre de ce projet, c.-à-d. l'intégration de notre mécanisme de résolution de conflits automatiques pour le type Séquence et le développement de la couche de livraison des messages associée. Pour chaque couche logicielle, nous identifions ses limites et présentons de potentielles pistes d'améliorations.

Finalement, nous récapitulons dans le chapitre 2 les contributions réalisées dans le cadre de cette thèse. Puis nous clotûrons ce manuscrit en introduisant plusieurs des pistes de recherches que nous souhaiterons explorer dans le cadre de nos travaux futurs.

Chapitre 2

Conclusions et perspectives

Sommaire

Dans ce chapitre, nous revenons sur les contributions présentées dans cette thèse. Nous rappelons le contexte dans lequel elles s'inscrivent, récapitulons leurs spécificités et apports, et finalement précisons plusieurs de leurs limites. Puis, nous concluons ce manuscrit en présentant plusieurs pistes de recherche qui nous restent à explorer à l'issue de cette thèse. La première s'inscrit dans la continuité directe de nos travaux sur un mécanisme de ré-identification pour CRDTs pour le type Séquence pour les applications Local-First Software (LFS). Les dernières traduisent quant à elles notre volonté de recentrer nos travaux sur le domaine plus général des CRDTs.

2.1 Résumés des contributions

2.1.1 Réflexions sur l'état de l'art des CRDTs

Les Conflict-free Replicated Data Types (CRDTs) [19] sont de nouvelles spécifications des types de données. Ils sont conçus pour permettre à un ensemble de noeuds d'un système de répliquer une même donnée et pour leur permettre de la consulter et de la modifier sans aucune coordination préalable. Les copies des noeuds divergent alors temporairement. Les noeuds se synchronisent ensuite pour s'échanger leurs modifications

respectives. Les CRDTs garantissent alors la convergence forte à terme [19], c.-à-d. que les noeuds obtiennent de nouveau des copies équivalentes de la donnée.

L'absence de coordination entre les noeuds avant modifications implique que des noeuds peuvent modifier la donnée en concurrence. De telles modifications peuvent donner lieu à des conflits, e.g. l'ajout et la suppression en concurrence d'un même élément dans un ensemble. Pour pallier ce problème, les CRDTs incorporent un mécanisme de résolution de conflits automatiques directement au sein de leur spécification.

Il convient de noter qu'il existe plusieurs solutions possibles pour résoudre un conflit. Pour reprendre l'exemple de l'élément ajouté et supprimé en concurrence d'un ensemble, nous pouvons par exemple soit le conserver l'élément, soit le supprimer. Nous parlons alors de sémantique du mécanisme de résolution de conflits automatique.

De la même manière, il existe plusieurs approches possibles pour synchroniser les noeuds, e.g. diffuser chaque modification de manière atomique ou diffuser l'entièreté de l'état périodiquement. Ainsi, lors de la définition d'un CRDT, il convient de préciser les sémantiques de résolution de conflits qu'il adopte et le modèle de synchronisation qu'il utilise [32].

Depuis leur formalisation, de nombreux travaux ont abouti à la conception de nouveaux CRDTs, soit en spécifiant de nouvelles sémantiques de résolution de conflits pour un type de données [33], soit en spécifiant de nouveaux modèles de synchronisation [34] ou en enrichissant les spécifications des modèles existants [35, 36].

Dans notre présentation des CRDTs (cf. **??**, page **??**), nous présentons chacun de ces aspects. Cependant, nous nous ne limitons pas à retranscrire l'état de l'art de la littérature. Notamment au sujet du modèle de synchronisation par opérations, nous précisons que le modèle de livraison causal n'est pas nécessaire pour l'ensemble des CRDTs synchronisés par opérations, c.-à-d. que certains peuvent adopter des modèles de livraison moins contraints et donc moins coûteux. Cette précision nous permet de proposer une étude comparative des différents modèles de synchronisation qui est, à notre connaissance, l'une des plus précises de la littérature (cf. **??**, page **??**).

Nous présentons ensuite les différents CRDTs pour le type Séquence de la littérature (cf. **??**, page **??**). Nous mettons alors en exergue les deux approches proposées pour concevoir le mécanisme de résolution de conflits automatiques pour le type Séquence, c.-à-d. l'approche à pierres tombales et l'approche à identifiants densément ordonnés. De nouveau, cette rétrospective nous permet d'expliciter des angles morts des articles d'origine, notamment vis-à-vis des modèle de livraison des opérations des CRDTs proposés. Puis, nous mettons en lumière les limites des évaluations comparant les deux approches, c.-à-d. le couplage entretenu entre approche du mécanisme de résolution de conflits et choix d'implémentations. Cette limite empêche d'établir la supériorité d'une des approches par rapport à l'autre. Finalement, nous conjecturons que le surcoût de ces deux approches est le même, c.-à-d. le coût nécessaire à la représentation d'un espace dense. Nous précisons dès lors par le biais de notre propre étude comparative comment ce surcoût s'exprime dans chacune des approches , c.-à-d. le compromis entre surcoût en métadonnées, calculs et bande-passante proposé par les deux approches (cf. **??**, page **??**).

Ces réflexions que nous présentons sur l'état des CRDTs définissent plusieurs pistes de recherches. Une première d'entre elles concerne notre étude comparative des modèles de synchronisation. D'après les critères que nous utilisons, une conclusion possible de cette comparaison est que le modèle de synchronisation par différences d'états rend obsolètes les modèles de synchronisation par états et par opérations. En effet, le modèle de synchronisation par différences d'états apparaît comme adapté à l'ensemble des contextes d'utilisation qui étaient jusqu'alors exclusifs à ces derniers, de par les multiples stratégies qu'il permet, e.g. synchronisation par états complets, synchronisation par états irréductibles...

Cette conclusion nous paraît cependant hâtive. Il convient d'étendre notre étude comparative pour prendre en compte des critères de comparaison additionnels pour confirmer cette conjecture, ou l'invalider et définir plus précisément les spécificités de chacun des modèles de synchronisation. Nous détaillons cette piste de recherche dans la sous-section 2.2.2.

Une seconde piste de recherche possible concerne les deux approches utilisées pour concevoir le mécanisme de résolution de conflits des CRDTs pour le type Séquence. Comme dit précédemment, nous conjecturons que ces deux approches ne sont finalement que deux manières différentes de représenter une même information : la position d'un élément dans un espace dense. La différence entre ces approches résiderait uniquement dans la manière que chaque représentation influe sur les performances du CRDT. Une piste de travail serait donc de confirmer cette conjecture, en proposant une formalisation unique des CRDTs pour le type Séquence.

2.1.2 Ré-identification sans coordination pour les CRDTs pour le type Séquence

Pour privilégier leur disponibilité, latence et tolérance aux pannes, les systèmes distribués peuvent adopter le paradigme de la réplication optimiste [13]. Ce paradigme consiste à relaxer la cohérence de données entre les noeuds du système pour leur permettre de consulter et modifier leur copie locale sans se coordonner. Leur copies peuvent alors temporairement diverger avant de converger de nouveau une fois les modifications de chacun propagées. Cependant, cette approche nécessite l'emploi d'un mécanisme de résolution de conflits pour assurer la convergence même en cas de modifications concurrentes. Pour cela, l'approche des CRDTs [18, 19] propose d'utiliser des types de données dont les modifications commutent nativement.

Depuis la spécification des CRDTs, la littérature a proposé plusieurs de ces mécanismes résolution de conflits automatiques pour le type de données Séquence [37, 38, 39, 40]. Cependant, ces approches souffrent toutes d'un surcoût croissant de manière monotone de leurs métadonnées et calculs. Ce problème a été identifié par la communauté, et celle-ci a proposé pour y répondre des mécanismes permettant soit de réduire la croissance de leur surcoût [41, 42], soit d'effectuer une Garbage Collection (GC) de leur surcoût [38, 20, 21]. Nous avons cependant déterminé que ces mécanismes ne sont pas adaptés aux systèmes P2P à large échelle souffrant de churn et utilisant des CRDTs pour le type Séquence à granularité variable.

Dans le cadre de cette thèse, nous avons donc souhaité proposer un nouveau mécanisme

adapté à ce type de systèmes. Pour cela, nous avons suivi l'approche proposée par [20, 21] : l'utilisation d'un mécanisme pour ré-assigner de nouveaux identifiants aux élements stockées dans la séquence. Nous avons donc proposé un nouveau mécanisme appartenant à cette approche pour le CRDT LogootSplit [23].

Notre proposition prend la forme d'un nouveau CRDT pour le type Séquence à granularité variable : RenamableLogootSplit. Ce nouveau CRDT associe à LogootSplit un nouveau type de modification, *ren*, permettant de produire un nouvel état de la séquence équivalent à son état précédent. Cette nouvelle modification tire profit de la granularité variable de la séquence pour produire un état de taille minimale : elle assigne à tous les éléments des identifiants de position issus d'un même intervalle. Ceci nous permet de minimiser les métadonnées que la séquence doit stocker de manière effective, c.-à-d. le premier et le dernier des identifiants composant l'intervalle.

Afin de gérer les opérations concurrentes aux opérations *ren*, nous définissons pour ces dernières un algorithme de transformation. Pour cela, nous définissons un mécanisme d'époques nous permettant d'identifier la concurrence entre opérations. Nous introduisons une relation d'ordre strict total, *priority*, pour résoudre de manière déterministe le conflit provoqué par deux opérations *ren*, c.-à-d. pour déterminer quelle opération *ren* privilégier. Finalement, nous définissons deux algorithmes, `renameId` et `revertRenameId`, qui permettent de transformer les opérations concurrentes à une opération *ren* pour prendre en compte l'effet de cette dernière. Ainsi, notre algorithme permet de détecter et de transformer les opérations concurrentes aux opérations *ren*, sans nécessiter une coordination synchrone entre les noeuds.

Pour valider notre approche, nous proposons une évaluation expérimentale de cette dernière. Cette évaluation se base sur des traces de sessions d'édition collaborative que nous avons générées par simulations.

Notre évaluation nous permet de valider de manière empirique les résultats attendus. Le premier d'entre eux concerne la convergence des noeuds. En effet, nos simulations nous ont permis de valider que l'ensemble des noeuds obtenaient des états finaux équivalents, même en cas d'opérations *ren* concurrentes.

Notre évaluation nous a aussi permis de valider que le mécanisme de renommage réduit à une taille minimale le surcoût du mécanisme de résolution de conflits incorporé dans le CRDT pour le type Séquence.

L'évaluation expérimentale nous a aussi permis de prendre conscience d'effets additionnels du mécanisme de renommage que nous n'avions pas anticipé. Notamment, elle montre que le surcoût éventuel du mécanisme de renommage, notamment en termes de calculs, est toutefois contrebalancé par l'amélioration précisée précédemment, c.-à-d. la réduction de la taille de la séquence.

Finalement, notons que le mécanisme que nous proposons est partiellement générique : il peut être adapté à d'autres CRDTs pour le type Séquence à granularité variable, e.g. un CRDT le type pour Séquence appartenant à l'approche à pierres tombales. Dans le cadre d'une telle démarche, nous pourrions réutiliser le système d'époques, la relation *priority* et l'algorithme de contrôle qui identifie les transformations à effectuer. Pour compléter une

telle adaptation, nous devrions cependant concevoir de nouveaux algorithmes `renameId` et `revertRenameId` spécifiques et adaptés au CRDT choisi.

Le mécanisme de renommage que nous présentons souffre néanmoins de plusieurs limites. Une d'entre elles concerne ses performances. En effet, notre évaluation expérimentale a mis en lumière le coût important en l'état de la modification *ren* par rapport aux autres modifications en termes de calculs (cf. **??**, page **??**). De plus, chaque opération *ren* comporte une représentation de l'ancien état qui doit être maintenue par les noeuds jusqu'à leur stabilité causale. Le surcoût en métadonnées introduit par un ensemble d'opérations *ren* concurrentes peut donc s'avérer important, voire pénalisant (cf. **??**, page **??**). Pour répondre à ces problèmes, nous identifions trois axes d'amélioration :

(i) La définition de stratégies de déclenchement du renommage efficaces. Le but de ces stratégies serait de déclencher le mécanisme de renommage de manière fréquente, de façon à garder son temps d'exécution acceptable, mais tout visant à minimiser la probabilité que les noeuds produisent des opérations *ren* concurrentes, de façon à minimiser le surcoût en métadonnées.

(ii) La définition de relations *priority* efficaces. Nous développons ce point dans nos perspectives, c.-à-d. dans la sous-section 2.2.1.

(iii) La proposition d'algorithmes de renommage efficaces. Cette amélioration peut prendre la forme de nouveaux algorithmes pour `renameId` et `revertRenameId` offrant une meilleure complexité en temps. Il peut aussi s'agir de la conception d'une nouvelle approche pour renommer l'état et gérer les modifications concurrentes, e.g. un mécanisme de renommage basé sur le journal des opérations (cf. **??**, page **??**).

Une seconde limite de RenamableLogootSplit que nous identifions concerne son mécanisme de GC des métadonnées introduites par le mécanisme de renommage. En effet, pour fonctionner, ce dernier repose sur la stabilité causale des opérations *ren*. Pour rappel, la stabilité causale représente le contexte causal commun à l'ensemble des noeuds du système. Pour le déterminer, chaque noeud doit récupérer le contexte causal de l'ensemble des noeuds du système. Ainsi, l'utilisation de la stabilité causale comme pré-requis pour la GC de métadonnées constitue une contrainte forte, voire prohibitive, dans les systèmes P2P à large échelle sujet au churn. En effet, un noeud du système déconnecté de manière définitive suffit pour empêcher la stabilité causale de progresser, son contexte causal étant alors indéterminé du point de vue des autres noeuds. Il s'agit toutefois d'une limite récurrente des mécanismes de GC distribués et asynchrones [38, 35, 43].

2.1.3 Éditeur de texte collaboratif P2P chiffré de bout en bout

Les applications collaboratives permettent à des utilisateur-rices de réaliser collaborativement une tâche. Elles permettent à plusieurs utilisateur-rices de consulter la version actuelle du document, de la modifier et de partager leurs modifications avec les autres. Ceci permet de mettre en place une réflexion de groupe, ce qui améliore la qualité du résultat produit [44, 45].

Cependant, les applications collaboratives sont historiquement des applications centralisées, e.g. Google Docs [46]. Ce type d'architecture induit des défauts d'un point de

vue technique, e.g. faible capacité de passage à l'échelle et faible tolérance aux pannes, mais aussi d'un point de vue utilisateur, e.g. perte de la souveraineté des données et absence de garantie de pérennité.

Les travaux de l'équipe Coast s'inscrivent dans une mouvance souhaitant résoudre ces problèmes et qui a conduit à la définition d'un nouveau paradigme d'applications : les applications Local-First Software (LFS) [12]. Le but de ce paradigme est la conception d'applications collaboratives, P2P, pérennes et rendant la souveraineté de leurs données aux utilisateur-rices.

Dans le cadre de cette démarche, l'équipe Coast développe depuis plusieurs années l'application Multi User Text Editor (MUTE), un éditeur de texte web collaboratif P2P temps réel chiffré de bout en bout. Cette application sert à la fois de plateforme de démonstration et de recherche pour les travaux de l'équipe, mais aussi de Proof of Concept (PoC) pour les LFS. Ainsi, MUTE propose au moment où nous écrivons ces lignes un aperçu des travaux de recherche existants concernant :

(i) Les mécanismes de résolution de résolutions de conflits automatiques pour l'édition collaborative de documents textes [23, 24, 25].

(ii) Les protocoles distribués d'appartenance au groupe [28].

(iii) Les mécanismes d'anti-entropie [47].

(iv) Les protocoles distribués d'authentification d'utilisateur-rices [26, 27].

(v) Les protocoles distribués d'établissement de clés de chiffrement de groupe [31].

(vi) Les mécanismes de conscience de groupe.

Dans cette liste, nous avons personnellement contribué à l'implémentation des CRDTs LogootSplit [23] et RenamableLogootSplit [25], et du protocole d'appartenance au groupe SWIM [28].

En son état actuel, MUTE présente cependant plusieurs limites. Tout d'abord, l'environnement web implique un certain nombre de contraintes, notamment au niveau des technologies et protocoles disponibles. Notamment, le protocole Web Real-Time Communication (WebRTC) repose sur l'utilisation de serveurs de signalisation, c.-à-d. de points de rendez-vous des pairs, et de serveurs de relais, c.-à-d. d'intermédiaires pour communiquer entre pairs lorsque les configurations de leur réseaux respectifs interdisent l'établissement d'une connexion directe. Ainsi, les applications P2P web doivent soit déployer et maintenir leur propre infrastructure de serveurs, soit reposer sur une infrastructure existante, e.g. celle proposée par OpenRelay [48].

Dans le cadre de MUTE, nous avons opté pour cette seconde solution. Cependant, ce choix introduit un Single Point Of Failure (SPOF) [6] dans MUTE : OpenRelay elle-même. Afin de garantir la pérennité de MUTE, nous devrions reposer non pas sur une unique infrastructure de serveurs de signalisation et de relais mais sur une multitude. Malheureusement, l'écosystème actuel brille par la rareté d'infrastructures publiques offrant ces

6. Single Point Of Failure (SPOF) : Point de défaillance unique

services. Nous devons donc encourager et supporter la mise en place de telles infrastructures par une pluralité d'organisations.

Une autre limite de ce système que nous identifions concerne l'utilisabilité des systèmes P2P de manière générale. L'expérience vécue suivante constitue à notre avis un exemple éloquent des limites actuelles de l'application MUTE dans ce domaine. Après avoir rédigé une version initiale d'un document, nous avons envoyé le lien du document à notre collaborateur pour relecture et validation. Lorsque notre collaborateur a souhaité accéder au document, celui-ci s'est retrouvé devant une page blanche : comme nous nous étions déconnecté du système entretemps, plus aucun pair possédant une copie n'était disponible pour se synchroniser.

Notre collaborateur était donc dans l'incapacité de récupérer le document et d'effectuer sa tâche. Afin de pallier ce problème, une solution possible est de faire reposer MUTE sur un réseau P2P global, e.g. le réseau de InterPlanetary File System (IPFS) [49], et d'utiliser les pairs de ce dernier, potentiellement des pairs étrangers à l'application, comme pairs de stockage pour permettre une synchronisation future. Cette solution limiterait ainsi le risque qu'un pair ne puisse récupérer l'état du document faute de pairs disponibles. Pour garantir l'utilisabilité du système P2P, une telle solution devrait donc permettre à un pair de récupérer l'état du document à sa reconnexion, malgré la potentielle évolution du groupe des collaborateur-rices et des pairs de stockage, e.g. l'ajout, l'éviction ou la déconnexion d'un des pairs. Cependant, la solution devrait en parallèle garantir qu'elle n'introduit aucune vulnérabilité, e.g. la possibilité pour les pairs de stockage selectionnés de reconstruire et consulter le document.

2.2 Perspectives

2.2.1 Définition de relations de priorité pour minimiser les traitements

Dans la **??**, nous avons spécifié la relation *priority* (cf. **??**, page **??**). Pour rappel, cette relation doit établir un ordre strict total sur les époques de notre mécanisme de renommage.

Cette relation nous permet ainsi de résoudre le conflit provoqué par la génération de modifications *ren* concurrentes en les ordonnant. Grâce à cette relation relation d'ordre, les noeuds peuvent déterminer vers quelle époque de l'ensemble des époques connues progresser. Cette relation permet ainsi aux noeuds de converger à une époque commune à terme.

La convergence à terme à une époque commune présente plusieurs avantages :

(i) Réduire la distance entre les époques courantes des noeuds, et ainsi minimiser le surcoût en calculs par opération du mécanisme de renommage. En effet, il n'est pas nécessaire de transformer une opérations livrée avant de l'intégrer si celle-ci provient de la même époque que le noeud courant.

(ii) Définir un nouveau Plus Petit Ancêtre Commun (PPAC) entre les époques courantes des noeuds. Cela permet aux noeuds d'appliquer le mécanisme de GC pour supprimer les époques devenues obsolètes et leur anciens états associés, pour ainsi minimiser le surcoût en métadonnées du mécanisme de renommage.

Il existe plusieurs manières pour définir la relation *priority* tout en satisfaisant les propriétés indiquées. Dans le cadre de ce manuscrit, nous avons utilisé l'ordre lexicographique sur les chemins des époques dans l'*arbre des époques* pour définir *priority*. Cette approche se démarque par :

(i) Sa simplicité.

(ii) Son surcoût limité, c.-à-d. cette approche n'introduit pas de métadonnées supplémentaires à stocker et diffuser, et l'algorithme de comparaison utilisé est simple.

(iii) Sa propriété arrangeante sur les déplacements des noeuds dans l'arbre des époques. De manière plus précise, cette définition de *priority* impose aux noeuds de se déplacer que vers l'enfant le plus à droite de l'arbre des époques. Ceci empêche les noeuds de faire un aller-retour entre deux époques données. Cette propriété permet de passer outre une contrainte concernant le couple de fonctions `renameId` et `revertRenameId` : leur reciprocité.

Cette définition présente cependant plusieurs limites. La limite que nous identifions est sa décorrélation avec le coût et le bénéfice de progresser vers l'époque cible désignée. En effet, l'époque cible est désignée de manière arbitraire par rapport à sa position dans l'arbre des époques. Il est ainsi possible que progresser vers cette époque détériore l'état de la séquence, c.-à-d. augmente la taille des identifiants et augmente le nombre de blocs, e.g. si l'état courant comporte majoritairement des éléments ayant été insérés en concurrence de la génération de l'époque cible. De plus, la transition de l'ensemble des noeuds depuis leur époque courante respective vers cette nouvelle époque cible induit un coût en calculs, potentiellement important (cf. **??**, page **??**).

Pour pallier ce problème, il est nécessaire de proposer une définition de *priority* prenant l'aspect efficacité en compte. L'approche considérée consisterait à inclure dans les opérations *ren* une ou plusieurs métriques qui représente le travail accumulé sur la branche courante de l'arbre des époques, e.g. le nombre d'opérations intégrées, les noeuds actuellement sur cette branche... L'ordre strict total entre les époques serait ainsi construit à partir de la comparaison entre les valeurs de ces métriques de leur opération *ren* respective.

Il conviendra d'adjoindre à cette nouvelle définition de *priority* un nouveau couple de fonctions `renameId` et `revertRenameId` respectant la contrainte de réciprocité de ces fonctions, ou de mettre en place une autre implémentation du mécanisme de renommage ne nécessitant pas cette contrainte, telle qu'une implémentation basée sur le journal des opérations (cf. **??**, page **??**).

Il conviendra aussi d'étudier la possibilité de combiner l'utilisation de plusieurs relations *priority* pour minimiser le surcoût global du mécanisme de renommage, e.g. en fonction de la distance entre deux époques.

Finalement, il sera nécessaire de valider l'approche proposée par une évaluation comparative par rapport à l'approche actuelle. Cette évaluation pourrait consister à mesurer

le coût du système pour observer si l'approche proposée permet de réduire les calculs de manière globale. Plusieurs configurations de paramètres pourraient aussi être utilisées pour déterminer l'impact respectif de chaque paramètre sur les résultats.

2.2.2 Étude comparative des différents modèles de synchronisation pour CRDTs

Comme évoqué dans l'état de l'art (cf. **??**, page **??**), un nouveau modèle de synchronisation pour CRDT fut proposé récemment [50]. Ce dernier propose une synchronisation des noeuds par le biais de différences d'états. Dans notre étude comparative des différents modèles de synchronisation (cf. **??**, page **??**), nous avons justifié que ce modèle de synchronisation est adapté à l'ensemble des contextes d'utilisation qui étaient jusqu'alors exclusifs soit au modèle de synchronisation par états, soit par opérations. Dans cette piste de recherche, nous souhaitons approfondir notre étude comparative pour déterminer si le modèle de synchronisation par différences d'états rend obsolètes les modèles de synchronisation précédents.

Pour rappel, ce nouveau modèle de synchronisation se base sur le modèle de synchronisation par états. Il partage les mêmes pré-requis, à savoir la nécessité d'une fonction `merge` associative, commutative et idempotente. Cette dernière doit permettre la fusion de toute paire d'états possible en calculant leur borne supérieure, c.-à-d. leur Least Upper Bound (LUB) [51].

La spécificité de ce nouveau modèle de synchronisation est de calculer pour chaque modification la différence d'état correspondante. Cette différence correspond à un élément irréductible du sup-demi-treillis du CRDT [36], c.-à-d. un état particulier de ce dernier. Cet élément irréductible peut donc être diffusé et intégré par les autres noeuds, toujours à l'aide de la fonction `merge`.

Ce modèle de synchronisation permet alors d'adopter une variété de stratégies de synchronisation, e.g. diffusion des différences de manière atomique, fusion de plusieurs différences puis diffusion du résultat..., et donc de répondre à une grande variété de cas d'utilisation.

Ainsi, un CRDT synchronisé par différences d'états correspond à un CRDT synchronisé par états dont nous avons identifié les éléments irréductibles. La différence entre ces deux modèles de synchronisation semble reposer seulement sur la possibilité d'utiliser ces éléments irréductibles pour propager les modifications, en place et lieu des états complets. Nous conjecturons donc que le modèle de synchronisation par états est rendu obsolète par celui par différences d'états. Il serait intéressant de confirmer cette supposition.

En revanche, l'utilisation du modèle de synchronisation par opérations conduit généralement à une spécification différente du CRDT, les opérations permettant d'encoder plus librement les modifications. Notamment, l'utilisation d'opérations peut mener à des algorithmes d'intégration des modifications différents que ceux de la fonction `merge`. Il convient de comparer ces algorithmes pour déterminer si le modèle de synchronisation par

opérations peut présenter un intérêt en termes de surcoût.

Au-delà de ce premier aspect, il convient d'explorer d'autres pistes pouvant induire des avantages et inconvénients pour chacun de ces modèles de synchronisation. À l'issue de cette thèse, nous identifions les pistes suivantes :

(i) La composition de CRDTs, c.-à-d. la capacité de combiner et de mettre en relation plusieurs CRDTs au sein d'un même système, afin d'offrir des fonctionnalités plus complexes. Par exemple, une composition de CRDTs peut se traduire par l'ajout de dépendances entre les modifications des différents CRDTs composés. Le modèle de synchronisation par opérations nous apparaît plus adapté pour cette utilisation, de par le découplage qu'il induit entre les CRDTs et la couche de livraison de messages.

(ii) L'utilisation de CRDTs au sein de systèmes non-sûrs, c.-à-d. pouvant compter un ou plusieurs adversaires byzantins [52]. Dans de tels systèmes, les adversaires byzantins peuvent par exemple générer des modifications différentes mais qui sont perçues comme identiques par les mécanismes de résolution de conflits. Cette attaque, nommée *équivoque*, peut provoquer la divergence définitive des copies. [43] propose une solution adaptée aux systèmes P2P à large échelle. Celle-ci se base notamment sur l'utilisation de journaux infalsifiables. Il convient alors d'étudier si l'utilisation de ces journaux infalsifiables ne limite pas le potentiel du modèle de synchronisation par différences d'états, e.g. en interdisant la diffusion des modifications par états complets.

Un premier objectif de notre travail serait de proposer des directives sur le modèle de synchronisation à privilégier en fonction du contexte d'utilisation du CRDT.

Ce travail permettrait aussi d'étudier la combinaison des modèles de synchronisation par opérations et par différences d'états au sein d'un même CRDT. Le but serait notamment d'identifier les paramètres conduisant à privilégier un modèle de synchronisation par rapport à l'autre, de façon à permettre aux noeuds de basculer dynamiquement entre les deux.

2.2.3 Proposition d'un framework pour la conception de CRDTs synchronisés par opérations

Plusieurs approches ont été proposées dans la littérature pour guider la conception de CRDTs :

(i) L'utilisation de la théorie des treillis pour la conception de CRDTs synchronisés par états et par différences d'états [19, 36].

(ii) L'utilisation d'un framework [35] pour la conception de CRDTs purs synchronisés par opérations.

Malgré ses améliorations [53, 54], le framework présenté dans [35] souffre de plusieurs limitations, ce qui entrave la conception de nouveaux CRDTs synchronisés par opérations.

Tout d'abord, il ne permet que la conception de CRDTs purs synchronisés par opérations, c.-à-d. des CRDTs dont les modifications enrichies de leurs arguments et d'une estampille fournie par la couche de livraison des messages sont commutatives. Cette contrainte limite à des types de données simples, e.g. le Compteur ou l'Ensemble, les

CRDTs pouvant être spécifiés et exclut des types de données plus complexes, e.g. la Séquence ou le Graphe.

Une seconde limite de ce framework est qu'il repose sur le modèle de livraison causal des opérations. Ce modèle induit l'ajout de données de causalité précises à chaque opération, sous la forme d'un vecteur de version [55, 56] ou d'une barrière causale [57]. Nous jugeons ce modèle trop coûteux pour les applications LFS à large échelle.

Nous souhaitons donc proposer un nouveau framework pour la conception de CRDTs synchronisés par opérations répondant à ces limites. Nos objectifs sont multiples.

Notre framework devrait permettre la conception de CRDTs non-purs. Ce framework devrait aussi mettre en lumière la présence et le rôle de deux modèles de livraison dans les CRDTs synchronisés par opérations, ainsi que leur relation :

(i) Le modèle de livraison minimal requis par le CRDT pour assurer la convergence forte à terme [19].

(ii) Le modèle de livraison employé par le système qui utilise le CRDT, qui est une stratégie offrant un compromis entre modèle de cohérence et performances. Ce second modèle de livraison doit être égal ou plus contraint que le modèle de livraison minimal du CRDT.

Notamment, nous souhaitons expliciter que pour un CRDT synchronisé par opérations, le premier modèle de livraison est immuable tandis que le second peut être amené à évoluer en fonction de l'état du système et de ses besoins.

Par exemple, un système peut initialement garantir le modèle de cohérence causale [7] à ses utilisateur-rices, et pour cela utiliser le modèle de livraison causal. Cependant, ce modèle de livraison implique un surcoût qui dépend du nombre de noeuds du système. Ce modèle de livraison peut donc devenir trop coûteux si le nombre de noeuds devient important. Ainsi, le système peut décider de temporairement relaxer le modèle de cohérence garanti, e.g. garantir seulement le modèle de cohérence PRAM [8] [58], dans le but d'utiliser un modèle de livraison moins coûteux, e.g. le modèle de livraison FIFO [9].

7. Le modèle de cohérence causale est le modèle de cohérence auquel les utilisateur-rices sont généralement habitué-es : il garantit que l'ensemble des modifications seront intégrées dans leur ordre de génération (cf. **??**, page **??**).

8. Le modèle de cohérence PRAM garantit seulement que les modifications d'un noeud seront intégrés par les autres noeuds dans leur ordre de génération.

9. Le modèle de livraison FIFO garantit que les messages d'un noeud seront livrés aux autres noeuds dans le même ordre qu'ils ont été envoyés.

Bibliographie

[1] Ina Fried. *Scoop : Google's G Suite cracks 2 billion users*. Last Accessed : 2022-10-19. URL : https://www.axios.com/2020/03/12/google-g-suite-total-users.

[2] Wikimedia. *Wikimedia Statistics - English Wikipedia*. Last Accessed : 2022-10-06. URL : https://stats.wikimedia.org/#/en.wikipedia.org.

[3] Statista. *Biggest social media platforms 2022*. Last Accessed : 2022-10-06. URL : https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/.

[4] GitHub. *Search · type :user*. Last Accessed : 2022-10-19. URL : https://github.com/search?q=type:user&type=Users.

[5] Taylor Lorenz. *Internet communities are battling over pixels*. Last Accessed : 2022-10-18. URL : https://www.washingtonpost.com/technology/2022/04/04/reddit-place-internet-communities/.

[6] Matthew O'Mara. *Twitch Plays Pokémon a wild experiment in crowd sourced gameplay*. Last Accessed : 2022-10-18. URL : https://financialpost.com/technology/gaming/twitch-plays-pokemon-a-wild-experiment-in-crowd-sourced-gameplay.

[7] Paul Baran. « On distributed communications networks ». In : *IEEE transactions on Communications Systems* 12.1 (1964), p. 1–9.

[8] Wikipedia. *Censorship of Wikipedia*. Last Accessed : 2022-10-18. URL : https://en.wikipedia.org/wiki/Censorship_of_Wikipedia.

[9] Cody Odgen. *Google Graveyard*. Last Accessed : 2022-10-11. URL : https://killedbygoogle.com/.

[10] Glen Greenwald et Ewen MacAskill. *NSA Prism program taps in to user data of Apple, Google and others*. Last Accessed : 2022-10-07. URL : https://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data.

[11] Barton Gellman et Laura Poitras. *U.S., British intelligence mining data from nine U.S. Internet companies in broad secret program*. Last Accessed : 2022-10-07. URL : https://www.washingtonpost.com/investigations/us-intelligence-mining-data-from-nine-us-internet-companies-in-broad-secret-program/2013/06/06/3a0c0da8-cebf-11e2-8845-d970ccb04497_story.html.

[12] Martin KLEPPMANN, Adam WIGGINS, Peter van HARDENBERG et Mark MCGRANAGHAN. « Local-First Software : You Own Your Data, in Spite of the Cloud ». In : *Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*. Onward ! 2019. Athens, Greece : Association for Computing Machinery, 2019, p. 154–178. ISBN : 9781450369954. DOI : 10.1145/3359591.3359737. URL : https://doi.org/10.1145/3359591.3359737.

[13] Yasushi SAITO et Marc SHAPIRO. « Optimistic Replication ». In : *ACM Comput. Surv.* 37.1 (mar. 2005), p. 42–81. ISSN : 0360-0300. DOI : 10.1145/1057977.1057980. URL : https://doi.org/10.1145/1057977.1057980.

[14] Douglas B TERRY, Marvin M THEIMER, Karin PETERSEN, Alan J DEMERS, Mike J SPREITZER et Carl H HAUSER. « 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.

[15] Daniel STUTZBACH et Reza REJAIE. « Understanding Churn in Peer-to-Peer Networks ». In : *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*. IMC '06. Rio de Janeriro, Brazil : Association for Computing Machinery, 2006, p. 189–202. ISBN : 1595935614. DOI : 10.1145/1177080.1177105. URL : https://doi.org/10.1145/1177080.1177105.

[16] Leslie LAMPORT. « The part-time parliament ». In : *Concurrency : the Works of Leslie Lamport*. 2019, p. 277–317.

[17] Diego ONGARO et John OUSTERHOUT. « In search of an understandable consensus algorithm ». In : *2014 USENIX Annual Technical Conference (Usenix ATC 14)*. 2014, p. 305–319.

[18] Marc SHAPIRO et Nuno PREGUIÇA. *Designing a commutative replicated data type*. Research Report RR-6320. INRIA, 2007. URL : https://hal.inria.fr/inria-00177693.

[19] Marc SHAPIRO, Nuno M. PREGUIÇA, Carlos BAQUERO et Marek ZAWIRSKI. « 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.

[20] Mihai LETIA, Nuno PREGUIÇA et Marc SHAPIRO. « Consistency without concurrency control in large, dynamic systems ». In : *LADIS 2009 - 3rd ACM SIGOPS International Workshop on Large Scale Distributed Systems and Middleware*. T. 44. Operating Systems Review 2. Big Sky, MT, United States : Assoc. for Computing Machinery, oct. 2009, p. 29–34. DOI : 10.1145/1773912.1773921. URL : https://hal.inria.fr/hal-01248270.

[21] Marek ZAWIRSKI, Marc SHAPIRO et Nuno PREGUIÇA. « Asynchronous rebalancing of a replicated tree ». In : *Conférence Française en Systèmes d'Exploitation (CFSE)*. Saint-Malo, France, mai 2011, p. 12. URL : https://hal.inria.fr/hal-01248197.

[22] Matthieu NICOLAS, Victorien ELVINGER, Gérald OSTER, Claudia-Lavinia IGNAT et François CHAROY. « MUTE : A Peer-to-Peer Web-based Real-time Collaborative Editor ». In : *ECSCW 2017 - 15th European Conference on Computer-Supported Cooperative Work*. T. 1. Proceedings of 15th European Conference on Computer-Supported Cooperative Work - Panels, Posters and Demos 3. Sheffield, United Kingdom : EUSSET, août 2017, p. 1–4. DOI : 10.18420/ecscw2017_p5. URL : https://hal.inria.fr/hal-01655438.

[23] Luc ANDRÉ, Stéphane MARTIN, Gérald OSTER et Claudia-Lavinia IGNAT. « 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.

[24] Victorien ELVINGER. « Réplication sécurisée dans les infrastructures pair-à-pair de collaboration ». Theses. Université de Lorraine, juin 2021. URL : https://hal.univ-lorraine.fr/tel-03284806.

[25] Matthieu NICOLAS, Gerald OSTER et Olivier PERRIN. « Efficient Renaming in Sequence CRDTs ». In : *IEEE Transactions on Parallel and Distributed Systems* 33.12 (déc. 2022), p. 3870–3885. DOI : 10.1109/TPDS.2022.3172570. URL : https://hal.inria.fr/hal-03772633.

[26] Hoang-Long NGUYEN, Claudia-Lavinia IGNAT et Olivier PERRIN. « Trusternity : Auditing Transparent Log Server with Blockchain ». In : *Companion of the The Web Conference 2018*. Lyon, France, avr. 2018. DOI : 10.1145/3184558.3186938. URL : https://hal.inria.fr/hal-01883589.

[27] Hoang-Long NGUYEN, Jean-Philippe EISENBARTH, Claudia-Lavinia IGNAT et Olivier PERRIN. « Blockchain-Based Auditing of Transparent Log Servers ». In : *32th IFIP Annual Conference on Data and Applications Security and Privacy (DBSec)*. Sous la dir. de Florian KERSCHBAUM et Stefano PARABOSCHI. T. LNCS-10980. Data and Applications Security and Privacy XXXII. Part 1 : Administration. Bergamo, Italy : Springer International Publishing, juil. 2018, p. 21–37. DOI : 10.1007/978-3-319-95729-6_2. URL : https://hal.archives-ouvertes.fr/hal-01917636.

[28] Abhinandan DAS, Indranil GUPTA et Ashish MOTIVALA. « SWIM : scalable weakly-consistent infection-style process group membership protocol ». In : *Proceedings International Conference on Dependable Systems and Networks*. 2002, p. 303–312. DOI : 10.1109/DSN.2002.1028914.

[29] Armon DADGAR, James PHILLIPS et Jon CURREY. « Lifeguard : Local health awareness for more accurate failure detection ». In : *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE. 2018, p. 22–25.

[30] Brice Nédelec, Julian Tanke, Davide Frey, Pascal Molli et Achour Mostéfaoui. « An adaptive peer-sampling protocol for building networks of browsers ». In : *World Wide Web* 21.3 (2018), p. 629–661.

[31] Mike Burmester et Yvo Desmedt. « A secure and efficient conference key distribution system ». In : *Advances in Cryptology — EUROCRYPT'94*. Sous la dir. d'Alfredo De Santis. Berlin, Heidelberg : Springer Berlin Heidelberg, 1995, p. 275–286. isbn : 978-3-540-44717-7.

[32] Nuno M. Preguiça. « Conflict-free Replicated Data Types : An Overview ». In : *CoRR* abs/1806.10254 (2018). arXiv : 1806.10254. url : http://arxiv.org/abs/1806.10254.

[33] Weihai Yu et Sigbjørn Rostad. « A Low-Cost Set CRDT Based on Causal Lengths ». In : *Proceedings of the 7th Workshop on Principles and Practice of Consistency for Distributed Data*. New York, NY, USA : Association for Computing Machinery, 2020. isbn : 9781450375245. url : https://doi.org/10.1145/3380787.3393678.

[34] Paulo Sérgio Almeida, Ali Shoker et Carlos Baquero. « Delta state replicated data types ». In : *Journal of Parallel and Distributed Computing* 111 (jan. 2018), p. 162–173. issn : 0743-7315. doi : 10.1016/j.jpdc.2017.08.003. url : http://dx.doi.org/10.1016/j.jpdc.2017.08.003.

[35] Carlos Baquero, Paulo Sergio Almeida et Ali Shoker. *Pure Operation-Based Replicated Data Types*. 2017. arXiv : 1710.04469 [cs.DC].

[36] Vitor Enes, Paulo Sérgio Almeida, Carlos Baquero et João Leitão. « Efficient Synchronization of State-Based CRDTs ». In : *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 2019, p. 148–159. doi : 10.1109/ICDE.2019.00022.

[37] Gérald Oster, Pascal Urso, Pascal Molli et Abdessamad Imine. « Data Consistency for P2P Collaborative Editing ». In : *ACM Conference on Computer-Supported Cooperative Work - CSCW 2006*. Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work. Banff, Alberta, Canada : ACM Press, nov. 2006, p. 259–268. url : https://hal.inria.fr/inria-00108523.

[38] Hyun-Gul Roh, Myeongjae Jeon, Jin-Soo Kim et Joonwon Lee. « Replicated abstract data types : Building blocks for collaborative applications ». In : *Journal of Parallel and Distributed Computing* 71.3 (2011), p. 354–368. issn : 0743-7315. doi : https://doi.org/10.1016/j.jpdc.2010.12.006. url : http://www.sciencedirect.com/science/article/pii/S0743731510002716.

[39] Nuno Preguica, Joan Manuel Marques, Marc Shapiro et Mihai Letia. « 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.

[40] Stéphane Weiss, Pascal Urso et Pascal Molli. « Logoot : A Scalable Optimistic Replication Algorithm for Collaborative Editing on P2P Networks ». In : *Proceedings of the 29th International Conference on Distributed Computing Systems - ICDCS 2009*. Montreal, QC, Canada : IEEE Computer Society, juin 2009, p. 404–412. doi : 10.1109/ICDCS.2009.75. url : http://doi.ieeecomputersociety.org/10.1109/ICDCS.2009.75.

[41] Brice Nédelec, Pascal Molli, Achour Mostéfaoui et Emmanuel Desmontils. « LSEQ : an adaptive structure for sequences in distributed collaborative editing ». In : *Proceedings of the 2013 ACM Symposium on Document Engineering*. DocEng 2013. Sept. 2013, p. 37–46. doi : 10.1145/2494266.2494278.

[42] Brice Nédelec, Pascal Molli et Achour Mostéfaoui. « A scalable sequence encoding for collaborative editing ». In : *Concurrency and Computation : Practice and Experience* (), e4108. doi : 10.1002/cpe.4108. eprint : https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.4108. url : https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4108.

[43] Victorien Elvinger, Gérald Oster et Francois Charoy. « Prunable Authenticated Log and Authenticable Snapshot in Distributed Collaborative Systems ». In : *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*. 2018, p. 156–165. doi : 10.1109/CIC.2018.00031.

[44] Sylvie Noël et Jean-Marc Robert. « Empirical study on collaborative writing : What do co-authors do, use, and like ? » In : *Computer Supported Cooperative Work (CSCW)* 13.1 (2004), p. 63–89.

[45] Jim Giles. « Special Report Internet encyclopaedias go head to head ». In : *nature* 438.15 (2005), p. 900–901.

[46] Google. *Google Docs*. Last Accessed : 2022-10-07. url : https://docs.google.com/.

[47] D. S. Parker, G. J. Popek, G. Rudisin, A. Stoughton, B. J. Walker, E. Walton, J. M. Chow, D. Edwards, S. Kiser et C. Kline. « Detection of Mutual Inconsistency in Distributed Systems ». In : *IEEE Trans. Softw. Eng.* 9.3 (mai 1983), p. 240–247. issn : 0098-5589. doi : 10.1109/TSE.1983.236733. url : https://doi.org/10.1109/TSE.1983.236733.

[48] OpenRelay. *OpenRelay*. Last Accessed : 2022-10-07. url : https://openrelay.xyz/.

[49] Protocol Labs. *IPFS*. Last Accessed : 2022-10-07. url : https://ipfs.io/.

[50] Paulo Sérgio Almeida, Ali Shoker et Carlos Baquero. « Efficient State-Based CRDTs by Delta-Mutation ». In : *Networked Systems*. Sous la dir. d'Ahmed Bouajjani et Hugues Fauconnier. Cham : Springer International Publishing, 2015, p. 62–76. isbn : 978-3-319-26850-7.

[51] Nuno M. Preguiça, Carlos Baquero et Marc Shapiro. « Conflict-free Replicated Data Types (CRDTs) ». In : *CoRR* abs/1805.06358 (2018). arXiv : 1805.06358. url : http://arxiv.org/abs/1805.06358.

[52] Leslie LAMPORT, Robert SHOSTAK et Marshall PEASE. « The Byzantine Generals Problem ». In : *Concurrency : The Works of Leslie Lamport*. New York, NY, USA : Association for Computing Machinery, 2019, p. 203–226. ISBN : 9781450372701. URL : https://doi.org/10.1145/3335772.3335936.

[53] Jim BAUWENS et Elisa Gonzalez BOIX. « Flec : A Versatile Programming Framework for Eventually Consistent Systems ». In : *Proceedings of the 7th Workshop on Principles and Practice of Consistency for Distributed Data*. PaPoC '20. Heraklion, Greece : Association for Computing Machinery, 2020. ISBN : 9781450375245. DOI : 10.1145/3380787.3393685. URL : https://doi.org/10.1145/3380787.3393685.

[54] Jim BAUWENS et Elisa Gonzalez BOIX. « Improving the Reactivity of Pure Operation-Based CRDTs ». In : *Proceedings of the 8th Workshop on Principles and Practice of Consistency for Distributed Data*. PaPoC '21. Online, United Kingdom : Association for Computing Machinery, 2021. ISBN : 9781450383387. DOI : 10.1145/3447865.3457968. URL : https://doi.org/10.1145/3447865.3457968.

[55] Friedemann MATTERN et al. *Virtual time and global states of distributed systems*. Univ., Department of Computer Science, 1988.

[56] Colin FIDGE. « Logical Time in Distributed Computing Systems ». In : *Computer* 24.8 (août 1991), p. 28–33. ISSN : 0018-9162. DOI : 10.1109/2.84874. URL : https://doi.org/10.1109/2.84874.

[57] Ravi PRAKASH, Michel RAYNAL et Mukesh SINGHAL. « An Adaptive Causal Ordering Algorithm Suited to Mobile Computing Environments ». In : *Journal of Parallel and Distributed Computing* 41.2 (1997), p. 190–204. ISSN : 0743-7315. DOI : https://doi.org/10.1006/jpdc.1996.1300. URL : https://www.sciencedirect.com/science/article/pii/S0743731596913003.

[58] Richard J LIPTON et Jonathan S SANDBERG. *PRAM : A scalable shared memory*. Princeton University, Department of Computer Science, 1988.

Résumé

Un système collaboratif permet à plusieurs utilisateur-rices de travailler ensemble pour créer un contenu. Afin de supporter des collaborations impliquant des millions d'utilisateurs, ces systèmes adoptent une architecture décentralisée pour garantir leur haute disponibilité, tolérance aux pannes et capacité de passage à l'échelle. Cependant, ces sytèmes échouent à garantir un autre ensemble de propriétés : confidentialité des données, souveraineté des données, pérennité et résistance à la censure. Pour répondre à ce problème, la littérature propose la conception d'applications Local-First Software (LFS) : des applications collaboratives pair-à-pair (P2P).

Une pierre angulaire des applications LFS sont les Conflict-free Replicated Data Types (CRDTs). Il s'agit de nouvelles spécifications des types de données, tels que l'Ensemble ou la Séquence, permettant à un ensemble de noeuds de répliquer une donnée. Les CRDTs permettent aux noeuds de consulter et de modifier la donnée sans coordination préalable, et incorporent un mécanisme de résolution de conflits pour intégrer les modifications concurrentes. Cependant, les CRDTs pour le type Séquence souffrent d'une croissance monotone du surcoût de leur mécanisme de résolution de conflits. Pouvons-nous proposer un mécanisme de réduction du surcoût des CRDTs pour le type Séquence qui soit compatible avec les applications LFS ? Dans cette thèse, nous proposons un nouveau CRDT pour le type Séquence, RenamableLogootSplit. Ce CRDT intègre un mécanisme de renommage qui minimise périodiquement le surcoût de son mécanisme de résolution de conflits ainsi qu'un mécanisme de résolution de conflits pour intégrer les modifications concurrentes à un renommage. Finalement, nous proposons un mécanisme de Garbage Collection (GC) qui supprime à terme le propre surcoût du mécanisme de renommage.

Abstract

A collaborative system enables multiple users to work together to create content. To support collaborations involving millions of users, these systems adopt a decentralised architecture to ensure high availability, fault tolerance and scalability. However, these systems fail to guarantee another set of properties : data confidentiality, data sovereignty, durability and resistance to censorship. To address this problem, the literature proposes the design of Local-First Software (LFS) applications : collaborative peer-to-peer applications.

A cornerstone of LFS applications are Conflict-free Replicated Data Types (CRDTs). CRDTs are new specifications of data types, e.g. Set or Sequence, enabling a set of nodes to replicate a data. CRDTs enable nodes to access and modify the data without prior coordination, and incorporate a conflict resolution mechanism to integrate concurrent modifications. However, Sequence CRDTs suffer from a monotonous growth in the overhead of their conflict resolution mechanism. Can we propose a mechanism for reducing the overhead of Sequence-type CRDTs that is compatible with LFS applications ? In this thesis, we propose a novel CRDT for the Sequence type, RenamableLogootSplit. This CRDT embeds a renaming mechanism that periodically minimizes the overhead of its conflict resolution mechanism as well as a conflict resolution mechanism to integrate concurrent modifications to a rename. Finally, we propose a mechanism of Garbage Collection (GC) that eventually removes the own overhead of the renaming mechanism.