— 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, tel que l'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.

— Il sera nécessaire de valider l'approche proposée par une évaluation comparative par rapport à l'approche actuelle. Elle consistera à monitorer le coût du système pour observer si l'approche proposée permet de réduire les calculs de manière globale. Plusieurs configurations de paramètres pourront aussi être utilisées pour déterminer l'impact respectif de chaque paramètre sur les résultats.

1.2.2 Détection et fusion manuelle de versions distantes

— À l'issue de cette thèse, nous constatons plusieurs limites des mécanismes de résolution de conflits automatiques dans les systèmes P2P à large échelle. La première d'entre elles est l'utilisation d'un contexte causal. Le contexte causal est utilisé par les mécanismes de résolution de conflits pour :

(i) Satisfaire le modèle de cohérence causale, c.-à-d. assurer que si nous avons deux modifications m_1 et m_2 telles que $m1 \rightarrow m_2$, alors l'effet de m_2 supplantera celui de m_1. Ceci permet d'éviter des anomalies de comportement de la part de la structure de données du point de vue des utilisateur-rices, par exemple la résurgence d'un élément supprimé au préalable.

(ii) Permettre de préserver l'intention d'une modification malgré l'intégration préalable de modifications concurrentes.

— Le contexte causal est utilisé de manière différente en fonction du mécanisme de résolution de conflit. Dans l'approche Operational Transformation (OT), le contexte causal est utilisé par l'algorithme de contrôle pour déterminer les modifications concurrentes à une modification lors de son intégration, afin de prendre en compte leurs effets. Dans l'approche CRDT, le contexte causal est utilisé par la structure de données répliquée à la génération de la modification pour en faire une modification indépendante de l'état, c.-à-d. un élément du sup-demi-treillis représentant la structure de données répliquée.

— Le contexte causal peut être représenté de différentes manières. Par exemple, le contexte causal peut prendre la forme d'un vecteur de version [12, 13] ou d'un Directed Acyclic Graph (DAG) des modifications [14]. Cependant, de manière intrinsèque, le contexte causal ne fait que de croître au fur et à mesure que des modifications sont effectuées ou que des noeuds rejoignent le système, incrémentant son surcoût en métadonnées, calculs et bande-passante.

— La stabilité causale permet cependant de réduire le surcoût lié au contexte causal. En effet, la stabilité causale permet d'établir le contexte commun à l'ensemble des noeuds, c.-à-d. l'ensemble des modifications que l'ensemble des noeuds ont intégré. Ces modifications font alors partie de l'histoire commune et n'ont plus besoin d'être considérées par les mécanismes de résolution de conflits. La stabilité causale permet donc de déterminer et de tronquer la partie commune du contexte causal pour éviter que ce dernier ne pénalise les performances du système à terme.

— La stabilité causale est cependant une contrainte forte dans les systèmes P2P dynamiques à large échelle dans lesquels nous n'avons aucun contrôle sur les noeuds. Il ne suffit en effet que d'un noeud déconnecté pour empêcher la stabilité causale de progresser. Pour répondre à ce problème, nous nous trouvons dès lors devant un spectre d'approches possibles dont les extrémités sont les suivantes :

  (i) Considérer tout noeud déconnecté comme déconnecté de manière définitive, et donc les exclure du système. Cette première approche permet à la stabilité causale de progresser, et ainsi aux noeuds connectés de travailler dans des conditions optimales. Mais elle implique cependant que les modifications potentielles du noeud déconnecté soient perdues, c.-à-d. de ne plus pouvoir les intégrer en l'absence d'un lien entre leur contexte causal de génération et le contexte causal actuel de chaque autre noeud.

  (ii) Assurer en toutes circonstances la capacité d'intégration des modifications des noeuds, même ceux déconnectés. Cette seconde approche permet de garantir que les modifications potentielles d'un noeud déconnecté pourront être intégrées automatiquement, dans l'éventualité où ce dernier se reconnecte à terme. Mais elle implique de bloquer potentiellement de manière définitive la stabilité causale et donc le mécanisme de GC du contexte causal.

— La seconde limite que nous constatons est la limite des mécanismes actuels de résolution de conflits automatiques pour préserver l'intention des utilisateur-rices. Par exemple, les mécanismes de résolution de conflits automatiques pour le type Séquence présentés dans ce manuscrit (cf. **??**, page **??**) définissent l'intention de la manière suivante : *l'intégration de la modification par les noeuds distants doit reproduire l'effet de la modification sur la copie d'origine.* Cette définition assure que chaque modification est porteuse d'une intention, mais limite voire ignore toute la dimension sémantique de la dite intention. Nous conjecturons que l'absence de dimension sémantique réduit les cas d'utilisation de ces mécanismes.

— Considérons par exemple une édition collaborative d'un même texte par un ensemble de noeuds. Lors de la présence d'une faute de frappe dans le texte, e.g. le mot "HLLO", plusieurs utilisateur-rices peuvent la corriger en concurrence, c.-à-d.

insérer l'élément "E" entre "H" et "L". Les mécanismes de résolution de conflits automatiques permettent aux noeuds d'obtenir des résultats qui convergent mais à notre sens insatisfaisant, e.g. "HEEEEEELLO". Nous considérons ce type de résultats comme des anomalies, au même titre que l'entrelacement [15]. Dans le cadre de collaborations temps réel à échelle limitée, nous conjecturons cependant qu'une granularité fine des modifications permet de pallier ce problème. En effet, les utilisateur-rices peuvent observer une anomalie produite par le mécanisme de résolution de conflits, et la résoudre par le biais d'actions supplémentaires de compensation.

— Cependant, dans le cadre de collaborations asynchrones ou à large échelle, nous conjecturons que ces anomalies de résolution de conflits s'accumulent au point d'entraver la collaboration. Pour reprendre l'exemple de l'édition collaborative de texte, nous pouvons constater dans de tels cas de la duplication de contenu et/ou l'entrelacement de mots, phrases voire paragraphes nuisant à la clarté et correction du texte.

— Il convient alors de s'interroger sur le bien-fondé de l'utilisation de mécanismes de résolutions de conflits automatiques pour intégrer un ensemble de modifications provenant d'une version distante de la donnée répliquée par rapport à la version courante.

— Pour répondre à ces deux limites, nous souhaitons proposer une approche combinant un ou des mécanismes de résolution de conflits automatiques avec un ou des mécanismes de résolution de conflits manuels. L'idée derrière cette approche est de faire varier le mécanisme de résolution de conflits utilisé pour intégrer des modifications en fonction de la distance entre la version courante de la donnée répliquée et celle de leur génération :

  (i) Si cette distance est faible, utiliser un mécanisme de résolution de conflits automatique.

  (ii) Si cette distance dépasse une distance seuil, faire intervenir les utilisateur-rices par le biais d'un mécanisme de résolution de conflits manuel. L'utilisation d'un mécanisme manuel n'exclut cependant pas tout pré-travail de notre part pour réduire la charge de travail des utilisateur-rices dans le processus de fusion.

  Dans un premier temps, cette approche pourra se focaliser sur un type d'application spécifique, e.g. l'édition collaborative de texte.

— Pour mener à bien ce travail, il conviendra tout d'abord de définir la notion de distance entre versions de la donnée répliquée. Dans le cadre de l'édition collaborative, nous pourrons pour cela nous baser sur les travaux existants pour évaluer la distance entre deux textes. *Matthieu: TODO : Insérer refs distance de Hamming, Levenstein, String-to-string correction problem (Tichy et al)*

— Il conviendra ensuite de déterminer comment établir la valeur seuil à partir de laquelle la distance entre textes est jugée trop importante. Les approches d'évaluation de la qualité du résultat pourront être utilisées pour déterminer un couple $\langle$méthode de calcul de la distance, valeur de distance$\rangle$ spécifiant les cas pour lesquels les méthodes de résolution de conflits automatiques ne produisent plus un résultat

satisfaisant. *Matthieu: TODO : Insérer refs travaux Claudia et Vinh* Le couple obtenu pourra ensuite être confirmé par le biais d'expériences utilisateurs inspirées de [16, 17].

— Finalement, il conviendra de proposer un mécanisme de résolution de conflits adapté pour gérer les éventuelles fusions manuelles concurrentes, ou à défaut un mécanisme de conscience de groupe invitant les utilisateur-rices à effectuer des actions de compensation.

— Le gain attendu de cette approche est la progression de la stabilité causale et la troncature d'une partie du contexte causal déterminée comme étant au-delà de la valeur seuil de distance, e.g. en retirant des entrées du vecteur de version correspondant à des noeuds inactifs ou les premières modifications du DAG.

1.2.3 Étude comparative des différents modèles de synchronisation pour CRDTs

— La spécification récente des Delta-based CRDTs . Ce nouveau type de CRDTs se base sur celui des State-based CRDTs. Partage donc les mêmes pré-requis :
  — États du type de données répliqué forment un sup-demi-treillis
  — Modifications locales entraînent une inflation de l'état
  — Possède une fonction de `merge`, permettant de fusionner deux états S et S', et qui
    — Est associative, commutative et idempotente
    — Retourne S", la Least Upper Bound (LUB) de S et S' (c.-à-d. $\nexists S''' \cdot merge(S, S') < S''' < S''$)

  Et bénéficie de son principal avantage : synchronisation possible entre deux pairs en fusionnant leur états, peu importe le nombre de modifications les séparant.

— Spécificité des Delta-based CRDTs est de proposer une synchronisation par différence d'états. Plutôt que de diffuser l'entièreté de l'état pour permettre aux autres pairs de se mettre à jour, idée est de seulement transmettre la partie de l'état ayant été mise à jour. Correspond à un élément irréductible du sup-demi-treillis. Permet ainsi de mettre en place une synchronisation en temps réel de manière efficace. Et d'utiliser la synchronisation par fusion d'états complets pour compenser les défaillances du réseau

— Ainsi, ce nouveau type de CRDTs semble allier le meilleur des deux mondes :
  — Absence de contrainte sur le réseau autre que la livraison à terme
  — Propagation possible en temps réel des modifications

  Semble donc être une solution universelle :
  — Utilisable peu importe la fiabilité réseau à disposition
  — Empreinte réseau du même ordre de grandeur qu'un Op-based CRDT
  — Utilisable peu importe la fréquence de synchronisation désirée