# Dédicaces

Je dédie ce travail à :

Mes chers parents, qui ont œuvrés durement pour ma réussite et qui ont cru en moi et m'ont soutenu quotidiennement durant l'élaboration de ce projet ainsi qu'à mon encadreur Mr *Mechaoui* qui a su me conseiller et m'encourager jusqu'à ce que je puisse le mener à terme.

Sans oublier ma famille, mes amis, mes proches ainsi que toutes les personnes qui ont contribué à la mise en œuvre de ce travail par leurs idées, leurs conseils ou par leur simple présence.

# Remerciements

*Avant tout, je remercie **DIEU**, le tout puissant, pour la force, la volonté, la santé et la patience qu'il m'a donnée pour accomplir ce travail.*

*Je tiens à exprimer mon grand respect et ma gratitude à mon promoteur le professeur **Mechaoui**, pour m'avoir honoré en acceptant de diriger mon mémoire, pour son encadrement de qualité, ses précieuses suggestions scientifiques, sa présence encourageante, et sa patience tout au long de ce travail.*

*mes reconnaissances vont aussi à tous mes enseignants pour leurs apports inestimables en informations indispensables pour ma formation.*

*Enfin, je remercie chaleureusement toutes personnes ayant contribués de près ou de loin à l'élaboration de ce travail.*

*J'adresse mes remerciements infinis à mes parents pour leurs soutiens et leurs encouragements.*

*Une pensée particulière est adressée à tous nos collègues et amis,*

*Pour terminer, Merci pour tous ceux qui, par leurs remarques et leurs conseils, ont contribué à la réalisation de ce travail.*

# Liste des figures

# Liste des tableaux

# Liste des abréviations

| Abréviation | Expression Complète | Page |
|---|---|---|
| P2P | Peer To Peer | 1 |
| CVS | Concurrent Versions System | 1 |
| XML | Extensible Markup Language | 1 |
| CCI | Coherence, Causality, Intention | 14 |
| CRDT | Commutative Replicated Data Type | 16 |
| TO | Transformée Opérationnelle | 17 |
| IT | Inclusive Transformation | 21 |
| UML | Unified Modeling Language | 32 |
| HTML | HyperText Markup Language | 63 |
| CSS | Cascading Style Sheets | 63 |
| DOM | Document Object Model | 63 |
| IDE | Integrated Development Environment | 63 |
| API | Application Programming Interface | 63 |
| WHATWG | Web Hypertext Application Technology Working Group | 64 |
| LAN | Local Area Network | 64 |
| WAN | Wide Area Network | 64 |

# Table des matières

# Introduction

Aujourd'hui, avec le développement de la technologie mobile et l'apparition du web social, les gens commençaient d'utiliser plus en plus l'environnement en ligne, ils partagent en permanence des informations, qu'il s'agisse de leurs projets ou de leurs dernières nouvelles.
Ce partage a obligé les gens de travailler ensemble ce qui a fait apparaitre le travail collaboratif. Ce développement a effectivement fait apparaitre une technologie sans fil améliorée et des appareils mobiles plus étendus dans la capacité de mener des activités de collaboration à de nouvelles situations et applications. Ce qui a permis aux gens de partager et de collaborer lors les évènements entre eux sans avoir besoin d'un serveur pour gérer leurs communications. Cela a fait l'objet de recherches approfondies de beaucoup de chercheurs dont le but de développer des systèmes performants et cohérents pour l'édition collaborative pair à pair.
Ce travail présente plusieurs systèmes d'édition collaborative existants, tel que le Wiki pour l'édition des documents, Google DOC et Google Wave introduits par Google, So6 pour l'édition collaborative des fichiers XML, CVS (Concurrent Versions System) pour la gestion de versions, la plateforme d'édition collaborative « IceCube » qui se base sur une approche de convergence et d'autres systèmes qui sont basés généralement sur des architectures centralisées. Cependant, après une étude et une analyse faites sur l'architecture centralisé ou bien dite client-serveur appliquée dans les systèmes d'édition collaborative, on trouve qu'elle rend ces systèmes plus faible et vulnérable aux pannes.

L'architecture P2P (peer-to-peer) permet de décentraliser les utilisateurs et donne une dynamique ce qui garantit le passage à l'échelle contrairement aux limites des systèmes centralisés, mais cette dynamique des paires pose un problème de mises à jour de tous les répliques distribuées.
Notre objectif est de proposer un nouveau type de données capable d'assurer la convergence des données dans un environnement d'édition collaborative P2P indépendamment de l'ordre

d’exécution des opérations d’édition sur les différents sites. Afin de gérer ce nouveau type de données, des nouvelles fonctions de transformation conformes à cette structure de données doivent être développées et optimisées de sorte que la transformation s'effectue le plus rapidement possible. Puisqu’un très grand nombre d’opérations peut être traité dans un intervalle de temps rétréci, la solution proposée ne permet pas seulement d’assurer que le système passe facilement à l’échelle mais aussi d’éviter le regroupement de plusieurs opérations simultanées et par conséquent d’améliorer les performances et d’assurer la convergence des données.

# Chapitre 1

# Etat de l’art

## 1.1 Introduction :

Grace au développement rapide des technologies que connait le monde actuel, l’Internet est devenu plus interactive et plus collaborative ce qui a permet aux applications d'édition collaborative d’avoir un grand progrès informatique. Mais ces applications classiques conçues pour soutenir une charge de données et d’utilisateurs limités ne permettent pas de supporter ce passage à l’échelle.

Dans ce chapitre, nous allons tout d’abord détailler la notion de l’édition collaborative, les systèmes P2P et ses architectures, le travail collaboratif, les systèmes élaborés dans ce contexte ainsi que les problèmes et les solutions proposées pour élargir les possibilités actuelles de la collaboration sur les différents types de réseaux.

## 1.2 Les systèmes d’édition collaborative :

Le développement des réseaux et du matériel informatique fait apparaitre un nouveau paradigme d’interaction homme-homme via l’ordinateur. Cela permet la collaboration entre des utilisateurs distribués dans l’espace ou dans le temps. Les formes de cette collaboration semblent variées mais s’appuient toujours sur le même principe, celui de faire collaborer un groupe de personnes situées dans des lieux différents et leurs permettre de partager et de travailler simultanément sur un ensemble de documents de différents types : texte, graphe, programme etc.

Les personnes sont connectées grâce aux systèmes d'édition collaborative. Ces systèmes fournissent un environnement simple, réactif et de haute disponibilité pour faciliter la coopération du travail et rendre les objets partagés visible et accessibles à tout moment. **[3]** Actuellement, les systèmes d'édition collaborative sont partout dans les forums interactifs de discussion, le développement des logiciels « open source », la rédaction d'articles scientifiques par plusieurs chercheurs et dans beaucoup d'autre domaine visant à éditer des documents numériques (texte, image, son …) par plusieurs utilisateurs en temps réel. **[4]**

## 1.3 Les modèles d'édition collaborative :

Il existe plusieurs modèles pour la mise en œuvre d'un système d'édition collaborative. On distingue généralement deux modèles élémentaires pour ces systèmes, le premier appelé centralisé qui est l'origine des réseaux informatiques, le deuxième appelé décentralisé ou Peer-to-Peer qui est apparu pour pallier les insuffisances du premier. On s'intéresse plus au modèle basé sur une architecture Peer-to-Peer.

## 1.4 Modèles pair-a-pair

### 1.4.1 Définition :

Le pair-à-pair (en anglais, peer-to-peer, abrégé P2P) est un modèle de réseau informatique où, les participants mettent une partie de leurs ressources, telles que la puissance de traitement, le stockage sur disque ou la bande passante réseau, directement à la disposition des autres participants du réseau, sans avoir besoin d'une coordination centrale par des serveurs ou des hôtes stables **[5].**

Les utilisateurs sont à la fois des clients et des serveurs des ressources, contrairement au modèle client-serveur traditionnel dans lequel la consommation et l'offre de ressources sont divisées. Les systèmes P2P collaboratifs émergents permet aux participants de faire des choses similaires tout en partageant des ressources, telle que l'édition collaborative.

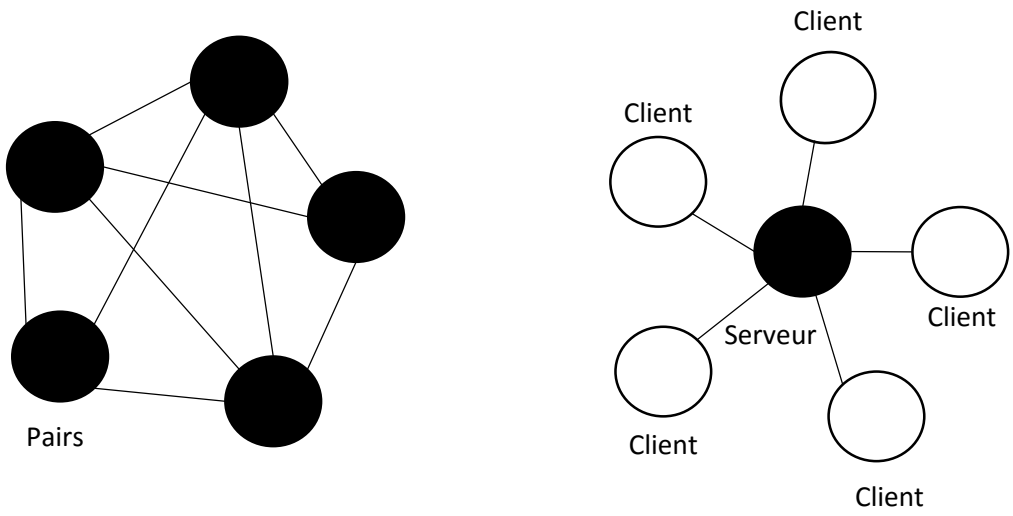



Figure 1 – Architecture P2P vs architecture Client-Serveur

### 1.4.2 Caractéristiques

Les systèmes P2P sont en plein essor depuis maintenant plusieurs années. Ce paradigme permet de concevoir des systèmes de très grande taille à forte disponibilité et à faible coût sans recourir à des serveurs centraux. En effet, les réseaux P2P sont des réseaux overlay, décentralisés, où tous les nœuds, appelés pairs, jouent à la fois le rôle de serveur et de client (voir **Figure 1**). L'organisation dans un tel réseau repose sur l'ensemble des pairs, donc il n'y a pas d'entité chargée d'administrer le réseau. En distribuant les données et les traitements sur tous les pairs du réseau, les systèmes P2P peuvent passer à très grande échelle sans recourir à des serveurs très puissants. Toutefois, ils induisent certains problèmes, liés par exemple à la sécurité des ressources. Dans la suite de cette section, nous présentons l'ensemble des caractéristiques que présente le réseau P2P **[7]**.

#### 1.4.2.1 Le passage à l'échelle

L'absence de coordination globale entre les pairs se répercute directement sur le passage à l'échelle des applications P2P. Les architectures P2P se présentent actuellement comme une solution viable pour permettre le passage à l'échelle de ses applications. **[8]**

#### 1.4.2.2 La tolérance aux fautes

La disponibilité d'un service dans l'architecture client-serveur repose intégralement sur celle du serveur. S'il s'écoule, le service qu'il fournit sera indisponible. Contrairement au contexte P2P, là où aucun point central de faute n'existe. Si un participant disparaît, le service continuera d'être fourni par le reste et sa disponibilité n'est plus liée à tel ou tel participant mais à la communauté des participants qui le fournissent.

#### 1.4.2.3 La dynamicité

Le comportement dynamique des pairs est l'une des caractéristiques les plus importantes des réseaux P2P, les participants peuvent se connecter ou se déconnecter du système de manière libre et à n'importe quel moment.

#### 1.4.2.4 L'organisation automatique

L'absence d'un élément central dans les systèmes P2P les oblige d'être organisés automatiquement en mettant en place des mécanismes d'auto-organisation permettant aux participants de réaliser une fonction globale de manière décentralisée quels que soient les connexions et déconnexions des pairs et la disponibilité des ressources dans le réseau.

#### 1.4.2.5 Un réseau virtuel

Un système P2P offre souvent la propriété de transparence qui peut être appliquée à plusieurs points. Un des points les plus importants est de faire l'abstraction des natures des participants qui ont souvent différentes caractéristiques :

- Matérielles, tel que : des stations de travail, des téléphones mobiles, etc.
- Logicielles, au niveau des systèmes d'exploitation par exemple.
- De communication, s'ils utilisent de différentes technologies

Un tel système est considéré comme un réseau virtuel.

### 1.4.3 Architecture des systèmes P2P :

Les architectures des systèmes pair-à-pair sont divisées en deux grandes classes : *architectures des réseaux structurés* et *architectures des réseaux non structurés*.

#### 1.4.3.1 Les systèmes P2P non structurés :

Les systèmes P2P non structurés sont créés de manière non-déterministe, il n'y a pas de répertoire centralisé ni de contrôle précis sur la topologie du réseau ou l'emplacement des fichiers. Un nœud peut connaître ses voisins dans le réseau sans connaître les données qu'ils stockent, le placement de données est effectué généralement sur le nœud demandeur et la recherche d'une donnée est réalisée par inondation.

Les approches développées sur ces systèmes simples donnent lieu à des systèmes tolérants aux fautes où les nœuds sont très autonomes. Par contre, la recherche d'une donnée dans un grand nombre de nœuds peut être longue et ne donne que des résultats partiels. **[8]**

##### 1.4.3.1.1 **Architecture centralisée :**

Un unique serveur est placé au centre et relie tous les utilisateurs. Il recense les fichiers proposés par les différents clients (voir **Figure 2**).

Contrairement au mode Client/serveur, le serveur ici possède uniquement des index sur les fichiers pour faire la recherche, il ne dispose pas de fichiers et le téléchargement se fait à partir du nœud possédant le fichier.

Comme exemple on trouve Napster, une application de partage de fichiers musicaux mp3. Il utilise un modèle P2P à répertoire centralisé pour les données administratives et les index des fichiers mp3 téléchargeables par les pairs. Un pair se connecte directement sur le serveur. Le

protocole basique est simple, avec des primitives de service : login request, login OK, login failed, user unknown, user not accepted, …

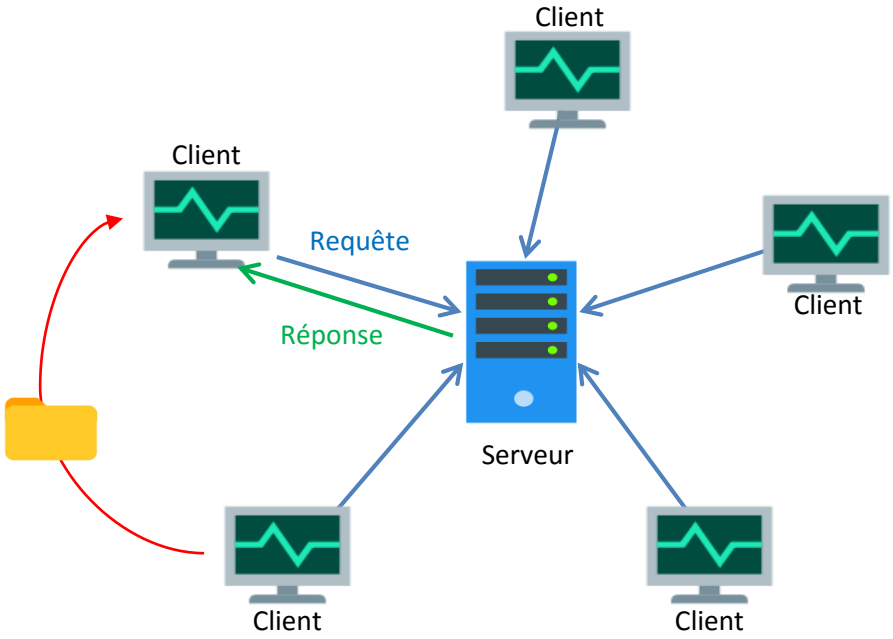



Figure 2 – Exemple d'architecture centralisée [6].

#### 1.4.3.1.2 **Architecture décentralisée**

Dans un système purement décentralisé personne ne joue le rôle d'administrateur du réseau. Chacun est connecté à un ensemble de voisins et joue le rôle de client et de serveur en même temps (voir **Figure 1**). La recherche se fait par un participant en interrogeant tous ses voisins et en leur envoyant un message de recherche, ses voisins vont faire la même avec leurs propres voisins à travers une méthode de propagation appelée inondation.

Cette méthode associe à chaque message un champ TTL (Time To Live) pour comptabiliser le nombre de retransmissions restantes, une fois le nombre atteint 0, le message ne sera plus retransmis. Les participants ayant le fichier demandé répondent au voisin (avec nom du fichier + leur adresse IP) qui leur a retransmis la requête et ainsi, la réponse remonte de proche en proche jusqu'au participant qui a initié la requête pour qu'il puisse choisir les fichiers à télécharger en envoyant directement une requête de téléchargement au participant qui possède

le fichier. Cependant, cette inondation est coûteuse en bande passante et les recherches sur ce sujet sont devenues plus lentes que dans les réseaux centralisés.

L'application Gnutella est un exemple du modèle P2P purement décentralisé. C'est un protocole de fichiers partagés qui propose aux usagers un service qui ne repose sur aucune infrastructure physique concrète mais sur un simple logiciel distribué au sein d'une communauté d'utilisateurs.

#### 1.4.3.1.3 **Architectures hybrides :**

Le modèle hybride est une combinaison des deux types d'architectures (centralisé et décentralisé) pour créer un réseau super-pair (voir **Figure 3**). Dans ce réseau, certains pairs dits super-pairs, jouent le rôle d'un serveur pour un ensemble de pairs et effectuent des fonctions complexes comme le traitement des requêtes, le contrôle d'accès et la gestion des métadonnées ce qui ressemble au réseau client-serveur.

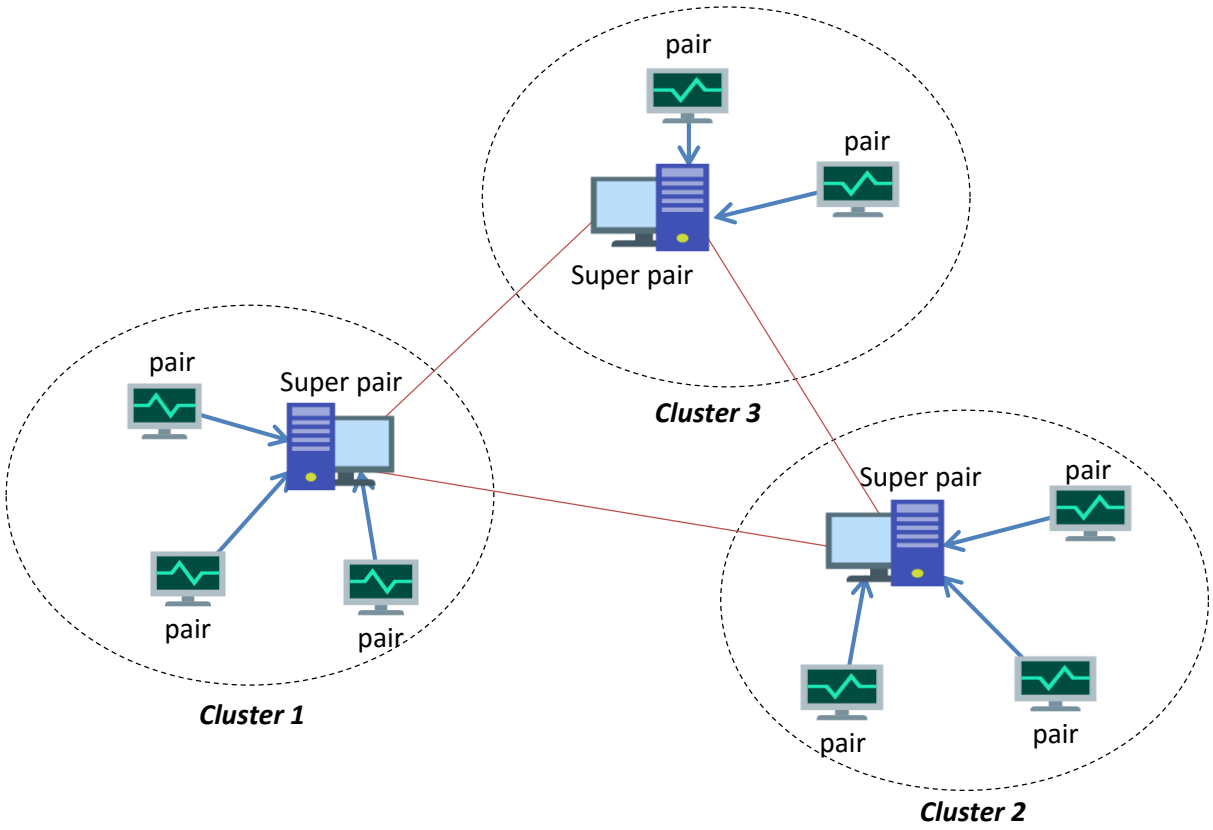



Figure 3 – Exemple d'architecture hybride [6].

Ces super-pairs sont organisés entre eux en mode pair-à-pair, choisis automatiquement en fonction de leurs caractéristiques (bande passante, vitesse de traitement, capacité mémoire) et remplacés en cas de présence d'une faute ce qui vient des architectures décentralisées.

#### 1.4.3.2 Les systèmes P2P structurés :

Contrairement aux réseaux non-structurés, Les réseaux structurés utilisent un protocole de contrôle de la topologie du réseau virtuel et pallient ainsi les limites des réseaux non-structurés. Chaque donnée est stockée dans un nœud avec une table de hachage distribuée (DHT - Distributed Hash Table) qui détermine la donnée et sa localisation par un tuple (ID_DONNEE, ID_NOEUD) qui indique qu'une donnée ID_DONNEE est stockée sur le nœud ID_NOEUD.

Il existe plusieurs grandes classes de DHTs qui se différencient principalement par le modèle topologique sur lequel elles reposent. Par exemple, Le protocole Pastry est un protocole de routage et de localisation P2P. Il repose sur un modèle P2P décentralisé et utilise une approche hybride pour le routage. Ce protocole est utilisé dans PAST ; une application distribuée de stockage de fichiers, et dans Scribe, une infrastructure de notification d'événements à grande échelle.

## 1.5 La réplication

Pour garantir qu'un système soit robuste et résistant aux pannes, les documents partagés auprès des nœuds du système peuvent être répliqués, ce qui facilite et accélère l'accès aux documents.

Comme les réseaux P2P peuvent prendre en charge un très grand nombre d'utilisateurs dans des zones géographiques éloignées, une opération (lecture ou écriture du document partagé) lancée par un utilisateur peut passer par un grand nombre de pairs pour atteindre le nœud souhaité. D'autre part, les informations que ce nœud contient peuvent être d'une taille importante. Cela peut impliquer une latence lors de réponse à la demande et ainsi des risques menaçant la continuité et la rapidité d'accès aux données.

Pour une meilleure amélioration des performances du réseau, le besoin a conduit à la conception d'une infrastructure distribuée et partageable permettant de partager et de dupliquer les données dans le système. Il s'agit donc d'un processus de réplication où tous ou bien certains objets partagés seront copiés et sauvegardés auprès de chaque site à travers le système. Ces copies sont dites répliques **[3]**.

La manipulation concurrentielle de répliques ne doit pas entraîner de perte d'informations, le système doit donc pouvoir assurer la continuité et la persistance des données partagées quelles que soient les conditions de communication. Ainsi, les procédures d'organisation et de gestion des répliques deviennent complexes car, pour chaque requête créée sur une copie (réplique), il est absolument indispensable de reproduire et de propager cette opération sur toutes les autres répliques. Les mises à jour de ces répliques peuvent être sous formes : synchrones ou asynchrones.

Les contraints sur les données partagées sont aussi différentes et suivent dans la plupart du temps le type de l'application collaborative c'est-à-dire le mode de collaboration et l'infrastructure de communication. Pour les applications qui exigent de voir rapidement les mises-à-jour comme les compagnies aériennes, les banques, les sociétés de bourse et la plupart des jeux multi-joueurs où les interactions entre les membres sont nombreuses, ce type d'applications nécessite une forte synchronisation pour les données qui sont modifiées fréquemment, on privilégiera donc un mode de réplication synchrone. Par contre, dans les applications où les données sont faiblement structurées comme dans les forums, les données échangées sont de natures différentes : date, auteur, sujet, corps d'un message, etc. ou comme dans les communautés professionnelles : base de données, agenda de groupe, édition collaborative d'un document quelconque, etc. on peut autoriser une collaboration entre des utilisateurs distants et potentiellement déconnectés c'est-à-dire asynchrone. **[9]**

### 1.5.1 La réplication synchrone vs asynchrone :

Dans un mode de réplication asynchrone, les transactions effectuées sont regroupées et sauvegardées suivant leurs ordres d'exécution sur un site, ensuite elles sont diffusées plus tard aux autres pairs du réseau à l'aide d'un processus de synchronisation. Dans ce mode de

réplication, les utilisateurs ont toujours accès au document partagé même s'ils possèdent d'une version plus ancienne. Cependant, les données modifiées ne sont pas assez sécurisées **[9]**.

A vrai dire, la séquence de transactions effectuée sur un site est stockée dans une file locale afin qu'elle soit diffusée ultérieurement aux autres nœuds distants. En cas de panne du site serveur, les transactions seront perdues et ne peuvent jamais être récupérées **[9]**.

Pour la réplication synchrone, les répliques sont traitées en temps réel. Le système effectue des mises–à-jour immédiates sur toutes les copies suite au lancement de la moindre transaction. Les requêtes sont déployées sur l'ensemble des bases de données avant leur validation sur le poste là où elles ont été générées **[3]**. Ces opérations sont aussitôt propagées qu'elles peuvent être exécutées dans un ordre quelconque sur les sites distants provoquant ainsi des défaillances de certaines répliques. On peut dire alors que les répliques ne se trouvent pas dans le même état à un moment donné, de même, les répliques ne gardent pas leurs cohérences par rapport à la dernière version du document **[9]**.

Pour couvrir ces problèmes de mises-à-jour, plusieurs solutions envisageables ont été proposées, certaines utilisent le principe de verrouillage des écritures tel que dans les transactions de base de données, une fois les mises-à-jour propagées (les données atteignent les sites distants) et enregistrées, le système autorise les écritures à nouveau sur les répliques. D'autres solutions utilisent des mécanismes d'ordonnancement sur les transactions distantes, à chaque opération est attribué un identifiant unique indiquant son ordre d'exécution sur les nœuds distants. Cette opération risque d'attendre indéfiniment si les opérations qui la précèdent n'ont pas été reçues **[9]**.

Cependant, les verrous sur un ensemble de nœuds dynamiques sont d'un coût très élevé et le maintien d'un ordre total nécessite la présence d'un serveur central ce qui est incompatible avec les caractéristiques pair-à-pair. Même si ces solutions permettent au système de résister aux pannes, la gestion d'un très grand nombre d'utilisateurs et d'opérations dans un intervalle du temps rétréci ne sera pas garantie dans toutes les situations et le système passe ainsi difficilement à l'échelle. Ces solutions sont étroitement liées aux deux mécanismes de réplication dites pessimiste et optimiste **[3]**.

### 1.5.2 Les mécanismes de réplication

Deux approches sont possibles pour l'édition collaborative : l'approche pessimiste et l'approche optimiste.

L'approche pessimiste consiste à autoriser la lecture à partir de n'importe quelle réplique ce qui entraine que les mises à jour doivent être synchronisées et coordonnées, généralement par un site central. Ces mises à jour nécessitent usuellement de verrouiller le document ou une partie du document pendant la mise à jour et la propagation de celle-ci. En opposition avec ce mécanisme,

L'approche optimiste, ne demande pas de mises à jour immédiates, mais accepte les différentes mises à jour au fur et à mesure de l'édition. Cette approche laisse diverger les copies locales, mais elles doivent finir par converger sur un document unique lorsque toutes les mises à jour ont été faites. Nous verrons qu'assurer cette convergence n'est pas trivial et le processus d'édition peut créer des conflits entre les opérations. Par exemple, deux mises à jour simultanées choisissant deux couleurs distinctes sur un même objet seront en conflit. Malgré ces problèmes, cette approche reste la plus adaptée à l'édition en pair-à-pair.

## 1.6 Le modelé de cohérence CCI :

### 1.6.1 Respect de la causalité :

Pour certaines opérations, il existe une relation de précédence causale qui doit être maintenue. On dit que l'opération $Op_1$ précède l'opération $Op_2$ (noté $Op_1$ -> $Op_2$) si et seulement si 0P2 a été générée et exécutée sur une réplique après l'exécution de 0P1 sur cette même réplique. Puisque $Op_2$ a été générée après $Op_1$, elle tient compte implicitement de ces effets, c'est-à-dire que l'on suppose que cette opération $Op_2$ est dépendante des effets produits par l'opération $Op_1$.

Le respect de la causalité garantit que les opérations pour lesquelles il existe une relation de causalité, celles-ci seront exécutées dans le même ordre sur toutes les répliques.

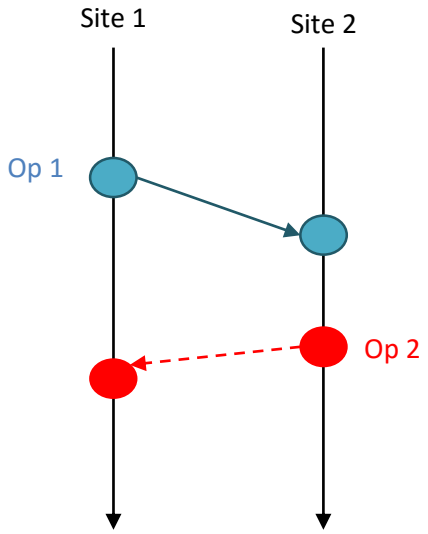



Figure 4 – Principe de la causalité

Par exemple, nous considérons le scénario présenté à la **figure 4**. Puisque l'opération $Op_2$ a été générée sur le site 2 après que l'opération $Op_1$ se soit exécutée, $Op_2$ précède $Op_1$. Si sur un troisième site, l'opération $Op_2$ est reçue avant l'opération 0P1 alors pour garantir l'exécution des opérations selon l'ordre de précédence, l'exécution de $Op_2$ sera différée jusqu'à la réception et l'exécution de $Op_1$.

Généralement, cette dépendance est maintenue en utilisant des vecteurs d'horloges.

### 1.6.2 Préservation de l'intention

Deux opérations $Op_1$ et $Op_2$ qui ne sont pas liées par une relation de précédence sont concurrentes. De ce fait, ces deux opérations peuvent être exécutées sur les différentes répliques dans un ordre quelconque. Cependant, si l'on exécute l'opération $Op_1$ avant l'opération $Op_2$ alors il faudra lors de l'exécution de $Op_2$ tenir compte des effets de $Op_1$ de façon à respecter les effets de $Op_2$.

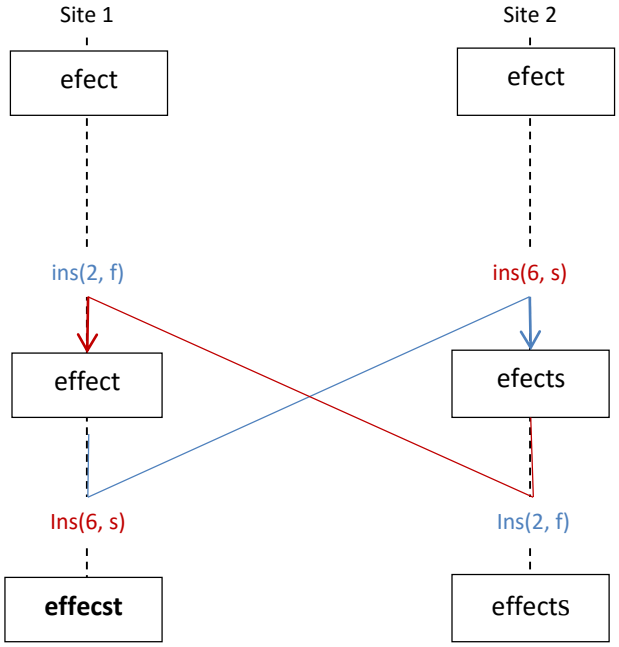



Figure 5 – Intégration incorrecte

La **figure 5** illustre un cas de violation de l'intention. Dans cet exemple, deux sites site l et site2 partagent une chaîne de caractères dont la valeur initiale est *effet*. Les deux répliques sont modifiées en parallèle : sur le site l, la chaîne est devenue *effect*, elle résulte de l'exécution de l'opération $Op_1$= Ins (2, f) qui a inséré le caractère f en deuxième position dans la chaîne. Sur le site2, la réplique a pour nouvelle valeur *efects* qui est le résultat de l'exécution de l'opération $Op_2$ = Ins (6, s).

Lorsque $Op_2$ est reçue et exécutée sur le site l, elle produit l'état *effecst* qui n'est pas l'état attendu. En effet, dans ce cas précis, l'effet de l'opération $Op_2$ n'a pas été respecté : il s'agissait d'insérer le caractère *s* à la fin de la chaîne.

### 1.6.3 La convergence :

Dans un système distribué avec réplication des données sur chaque site, un algorithme distribué est convergent si toutes les répliques sont identiques sur chaque site après exécution de toutes les opérations considérées. Dans le cas contraire, on parlera de divergence.

Cette propriété est difficile à assurer et fait toujours l'objet de recherches actuelles dans différents domaines.

#### 1.6.3.1 Types de la convergence :

La convergence se présente également sur différents types qui sont **[4]** :

- **La convergence forte :** dans ce type de convergence les répliques doivent être identiques sur tous les sites suite à la moindre modification. Le système doit donc garantir les mises-à-jour de l'ensemble des copies à chaque fois qu'un changement se produit.

- **Divergence limitée** : ce type de convergence n'est pas adaptée aux systèmes pair-à-pair car il tolère quelques défaillances des répliques. En effet, cette approche permet une divergence bornée en limitant généralement le nombre d'opérations à exécuter sur un site par un seuil.

- **Convergence à terme** : cette approche permet la divergence des répliques au cours des modifications à condition qu'elles finissent par converger. Les copies peuvent avoir des valeurs différentes tant que le système est actif, une fois les opérations de modification sur l'ensemble de nœuds sont propagées, le système doit conduire ces copies vers un état stable (les répliques sont identiques sur tous les sites). On peut dire alors que ce type de convergence est le plus convenable pour une édition collaborative.

## 1.7 Approches de gestion de la cohérence dans les éditeurs collaboratifs :

Plusieurs algorithmes de convergence et de détection de conflits issus des opérations concurrentes ont été élaborés. Ces algorithmes peuvent suivre différentes approches dont certaines détectent les conflits en imposent un verrouillage sur les ressources ou un ordre total sur les opérations **[4]**. Ces modèles sont très efficaces pour assurer la cohérence des répliques, malheureusement ils ne sont pas adaptés aux architectures P2P dans lesquelles aucune entité centrale n'est présente pour le maintien d'un ordre sur les opérations.
Une de ces approches est l'approche CRDT **[2]**, un type de donnes pour lequel l'exécution des opérations concurrentes est commutative. On peut facilement démontrer qu'un système base

sur un type de donnes CRDT respectant la causalité garantit la convergence **[2]**. En effet, d'après la condition de précédence, seules les opérations ayant une relation de causalité ne commutent pas. Or, ces opérations sont nécessairement exécutées dans le même ordre sur toutes les répliques. Par conséquent, si le système assure la causalité, le type de données CRDT assure la convergence.

Une autre approche basée sur la transformation des opérations propose une transformation préalable des opérations distantes reçues par rapport aux opérations concurrentes au niveau d'un site à l'aide d'un algorithme de transformation **[3]**.

Dans sa thèse, M. Ressel a démontré que cet algorithme de transformation peut assurer la convergence à condition que la transformée satisfasse deux propriétés TP1 et TP2. **[9]**

Cette approche dite Transformée Opérationnelle permet de garantir la convergence des documents sans imposer de contraintes de verrouillage ni d'ordre total, elle assure également que toutes les mises à jour soient faites sur l'ensemble des répliques même si ces opérations ne sont pas initialement commutatives, c'est pourquoi, dans la partie qui suit nous allons nous pencher plus vers cette approche qui semble beaucoup plus convenable pour une édition collaborative sur un réseau P2P. **[9]**

## 1.8 Conclusion :

Dans un premier temps, nous avons présenté la notion d'édition collaborative et des systèmes p2p. Ensuite, nous avons parlé du modèle de cohérence des répliques ainsi que les approches de cohérence dans les éditeurs collaboratif tel que CRDT et TO.

Dans le chapitre suivant, nous allons détailler l'approche TO et étudier ses algorithmes de transformation et d'intégration.

# Chapitre 2

# Edition collaborative des documents

## 2.1 Introduction :

Dans ce chapitre, nous allons introduire les objets les plus communs (texte, arbres, etc.) utilisés dans l'édition collaborative. Ensuite, nous donnons une présentation générale de l'approche des transformées opérationnelles (TO) et les éléments de base du modèle des TO. Puis, nous présentons les conditions qui doivent être satisfaites pour garantir la convergence des copies. Enfin, nous terminons le chapitre par une conclusion.

## 2.2 Les documents

### 2.2.1 Textes non formatés

Le document texte non formaté est simplement une suite de caractères. Ce type de document a été longtemps favorisé par les éditeurs collaboratifs. Tout autre type de document peut se ramener à ce type primitif sachant que la structure du document sera perdue rapidement lors de l'édition de ce type de fichier.

Les opérations d'édition sur ce format texte sont l'insertion et la suppression de caractères. La façon la plus naturelle d'identifier un élément dans un texte non formaté est de prendre sa position dans le texte. La difficulté qui survient est que les opérations d'édition changent cette identification (décalage de +1 en cas d'insertion avant le caractère considéré par exemple).**[10]**

### 2.2.2 Documents structurés

Dans le monde de l'édition, nous remarquons qu'il existe beaucoup de documents structurés. Ils sont beaucoup plus lisibles par l'humain que les documents non-structurés. Le type le plus commun est le document organisé de manière hiérarchique, que ce soit un texte structuré en chapitres, sections et paragraphes, que ce soit un code source avec les fonctions et les objets, voir même un fichier, Même une carte peut se voir comme un dessin où des nœuds relient des arêtes. **[10]**

### 2.2.3 Documents arborescents

Les structures arborescentes sont parmi les plus fondamentales en informatique. Un arbre est constitué de sommets et d'arêtes orientées reliant les sommets de manière suivante :
– Tout sommet a un et un seul père (une arête entrante) sauf la racine qui n'en a pas.
– Il n'y a pas de cycle.
– Les nœuds ou les arêtes peuvent avoir des étiquettes
– Les nœuds ou les arêtes peuvent être ordonnés ou pas.
Une forêt d'arbres usuels se représente par un seul arbre avec étiquetage sur les arêtes ce qui est pratique pour modéliser les documents XML. **[10]**

## 2.3 L'approche transformée opérationnelle pour l'édition collaborative

### 2.3.1 Généralités

L'approche des transformées opérationnelles a été utilisée dans le domaine des éditeurs collaboratifs synchrones pour sérialiser les opérations concurrentes. Elle se présente sous forme d'algorithmes qui consistent à transformer les opérations reçues pour tenir compte de celles qui ont pu entre-temps modifier l'état de l'objet. Elle utilise les propriétés sémantiques des opérations pour garantir la convergence des copies. Plus précisément, elle exploite la transformation des opérations pour construire un historique de chaque copie de l'objet.

Les historiques des différentes copies ne sont pas identiques mais elles sont équivalentes (elles aboutissent au même état final) puisque l'ordre d'exécution des opérations concurrentes peut être différent d'un historique à une autre. Comparée à d'autres techniques optimistes, elle possède les avantages suivants :

(i) Elle supporte un travail collaboratif sans contraintes. En effet, elle n'impose pas un ordre total sur les opérations concurrentes. Aussi, les sites peuvent échanger leurs modifications dans n'importe quel ordre.

(ii) Elle permet la transformation des opérations pour les exécuter dans un ordre arbitraire même si à l'origine ces opérations ne commutent pas.

(iii) Elle produit un état de convergence sans perte de mises-à-jour.

Chaque objet partagé doit avoir son propre algorithme de transformation dont l'écriture incombe aux développeurs de l'éditeur collaboratif. Ainsi, pour chaque paire d'opérations, le développeur doit préciser comment transformer ces opérations l'une par rapport à l'autre.

### 2.3.2 Modèle

Un éditeur collaboratif est considéré comme un groupe de sites (ou utilisateurs) qui manipulent un objet partagé, tel qu'un document texte ou graphique.

#### 2.3.2.1 Définition (Objet collaboratif)

Un objet est dit collaboratif s'il est une ressource de données partagée en lecture/écriture par plusieurs sites.

L'objet collaboratif est répliqué sur plusieurs sites de telle façon que chaque site pourra lire et/ou écrire sur sa propre copie. Chaque objet collaboratif possède :

- Un type (i.e. un texte, un arbre XML, un système de fichiers, etc.) qui définit un ensemble d'états possibles, noté par S ;
- Un ensemble d'opérations primitives, noté par O, où chaque opération possède une précondition de telle façon que l'opération ne peut s'exécuter sur un état de l'objet que si sa précondition est vérifiée.
- Une fonction de transition Do : $S \times O \rightarrow S$. Cette fonction permet d'appliquer l'effet de l'opération O sur l'état S et retourne un nouvel état.

#### 2.3.2.2 Définition (Opération locale/distante)

Une opération est dite locale si elle est générée localement sur site. Une opération est dite distante si elle provient d'un autre site.

Chaque site génère séquentiellement des opérations et les stocks dans une liste appelée un historique (ou journal des opérations).

#### 2.3.2.3 Définition (Algorithme de transformation)

Un algorithme de transformation est une fonction TI : $O \times O \rightarrow O$ définie pour tout $(op_1, op_2) \in O \times O$ tel que :

1. $op_1$ et $op_2$ sont concurrentes et définies sur le même état, et ;
2. la précondition de TI ($op_1$, $op_2$) est satisfaite sur l'état résultant de l'exécution d'$op_2$.

L'algorithme de transformation est utilisé comme suit :

Soient op_i et op_j deux opérations concurrentes définies sur le même état.

Supposons que op_i et op_j sont générées respectivement sur les sites i et j (avec i != j).

Soient op_i' = TI(op_i ,op_j ) et op_j' = TI(op_j ,op_i ).

Ainsi, le site i exécute l'historique [op_i ;op_j' ] et le site j exécute l'historique [op_j ;op_i' ].

D'une manière générale, lorsqu'une opération distante arrive sur un site donné, elle est transformée pour inclure les effets des autres opérations (qu'elle n'a pas vu sur son site originel) pour être correctement intégrée dans l'historique locale.

**Exemple 1** : Deux sites (ou utilisateurs) modifient en parallèle un document partagé, dont les copies contiennent initialement la chaîne de caractères 'efect' (voir la **figure 6**).

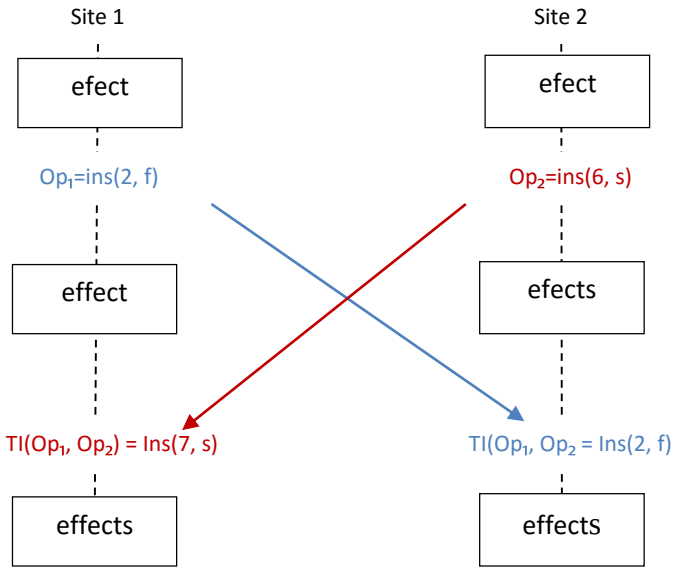



Figure 6 – Transformation des opérations concurrentes [1]

Le site 1 exécute l'opération $Op_1$ = Ins(2, f) pour insérer le caractère f à la position 2. Au même moment, le site 2 ajoute à la fin de la chaîne le caractère 's' en exécutant l'opération $Op_2$ = Ins(6,s).

Les opérations op 1 et $Op_2$ sont donc concurrentes puisqu'aucune n'a vu l'autre au moment de la génération. Lorsque $Op_2$ arrives sur le site 1, elle doit tenir compte que op 1 a été exécutée avant son arrivée.

Ainsi, $Op_2$ est transformée par rapport à op 1 pour devenir $Op_2'$ = TI($Op_2$, $Op_1$) = Ins(7,s) (la position d'insertion a été incrémentée car op 1 ) a inséré un caractère avant celui de op 2 ). Par contre, lorsque $Op_2$ arrive sur le site 2, elle reste inchangée car sa position d'insertion se trouve avant celle de $Op_2$. Les deux sites convergent donc vers la même chaîne de caractères " effets " qui contiennent les effets de $Op_1$ et $Op_2$.

De manière intuitive, on peut écrire la transformation TI pour les opérations Ins comme suit :

```
TI(Ins(p1, c1), Ins(p2, c2)) =
        si (P1 <= p2) alors retourner Ins (P1, c1)
        sinon retourner Ins (p1+1 , c1)
        finsi;
TI(Ins(p1, c1), Del(p2, c2)) =
        si (p1 <= p2) alors retourner Ins (p1, c1)
        sinon retourner Ins(pl-l , cl)
        finsi;
TI(Del (p2,c2), Ins(p1, c1)) =
        si (p2>= p1) alors retourner Del(p2+1, c2)
        sinon retourner Del(p2, c2)
        finsi;
```

### 2.3.3 Conditions de Convergence

La préservation des relations de causalité entre les opérations est nécessaire mais pas suffisante pour obtenir dans toutes les situations des exécutions qui garantissent la convergence des copies sur tous les sites. L'exemple suivant montre un cas de divergence entre deux opérations concurrentes.

**Exemple 2 :** Considérons deux sites partageant une chaîne de caractères dont la valeur initiale est" mis"(voir la **figure 7)**.

Le site 1 insère le caractère ‘a’ à la position 2 en exécutant l’opération op 1 = Ins(2,'a'). Au même moment, le site 2 exécute l’opération op 2 = Ins(2,'o') dont l’effet consiste à ajouter le caractère ‘o’ à la position 2.

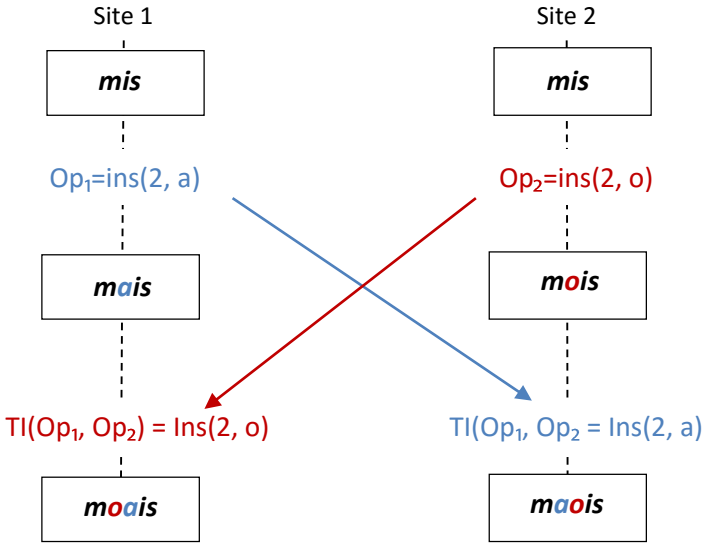



Figure 7 – Divergence de copies due à la violation de la condition TP1[1]

Pour intégrer les opérations concurrentes nous utilisons la fonction de transformation donnée à l’exemple 1. Sur le site 1, $Op_2$ est transformée par rapport à $Op_1$ pour produire :
$Op_2'$ =TI ($Op_2$, $Op_1$) = Ins(2,'o') dont l’exécution donne la chaîne ‘moais’ qui contient les effets des deux opérations.
Quant à $Op_1$, elle est transformée sur le site 2 par rapport à $Op_2$ pour donner :

$Op_1'$ = TI ($Op_1$, $Op_2$) = Ins (2,'a'). Le résultat de l’exécution de $Op_1'$ est la chaîne ‘ maois ‘ qui contient aussi les effets des deux opérations. Nous sommes en présence d’une situation de divergence puisque les deux sites 1 et 2 ont des états différents.

Afin de garantir la convergence des copies après l’application de la fonction de transformation TI, cette fonction doit satisfaire la condition suivante ;

#### 2.3.3.1 Définition (Condition TP1)

La fonction TI satisfait la condition TP1 ssi pour toutes les opérations concurrentes définies sur le même état $Op_1$, op 2 ∈ O : $[Op_1 ; Op_2'] \equiv_{st} [Op_2 ; Op_1']$

où $Op_1' = TI (Op_1, Op_2)$ et $Op_2' = TI (Op_2, Op_1)$.

La condition TP1 définit une identité d'états. Elle stipule que l'état produit en exécutant $Op_1$ avant $Op_2$ est le même que celui résultant de l'exécution de $Op_2$ avant $Op_1$.

La fonction TI défini à l'exemple 1 satisfait TP1 sauf dans le cas de deux insertions concurrentes à la même position, comme le montré l'exemple 2. Dans ce cas, on est en présence d'une situation de conflit entre deux insertions concurrentes. Il faut faire un choix pour déterminer quel caractère ajouter avant l'autre. Cependant, il faut assurer que le même choix sera fait sur tous les sites.

Plusieurs solutions ont été proposées dans la littérature, comme l'utilisation d'un ordre de priorité sur les opérations, l'utilisation d'un ordre total sur les identifiants des sites, et l'utilisation de l'ordre alphabétique sur les caractères. Ainsi si l'on utilise l'identifiant du site où l'opération a été générée, on doit ajouter cette information comme un nouveau paramètre dans les opérations et la définition de TI pour les opérations Ins devient donc :

```
TI(Ins(p1, c1,id1), Ins(p2, c2,id2) =

     si ((p1<p2) ou (p1 = p2 et id1<id2)) alors retourner
     Ins(p1,c1)

     sinon
             retourner Ins (p1+1, c1)

     finsi;
```

D'après cette définition, dans le cas où les positions d'insertion sont égales le caractère qui a le plus grand identifiant du site sera inséré après l'autre. La **figure 8** illustre l'utilisation de cette nouvelle définition pour éviter la divergence de l'exemple 2.

La condition TP1 est nécessaire mais pas suffisante pour assurer la convergence des copies lorsque l'on a plus de deux opérations concurrentes. L'exemple suivant montre une telle situation :

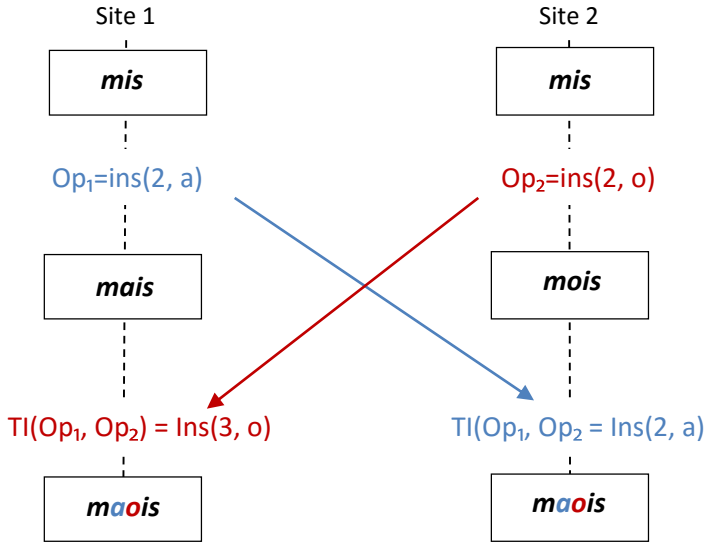



Figure 8 – Convergence de copies avec la satisfaction de la condition TP1[1]

**Exemple 3**

Considérons trois sites 1, 2 et 3 qui éditent simultanément le même texte (voir la **figure 9**).

Initialement, les utilisateurs démarrent à partir du même état “ciré”.

Ils génèrent concurremment les opérations $Op_1$ =Ins (3,'r'), $Op_2$ = Del(2) et $Op_3$ = Ins (2,'a') pour changer leurs états respectivement en “cirré”, “cré” et “cairé”.

Pour intégrer les trois opérations, nous utilisons la nouvelle définition de TI qui vérifie TP1.

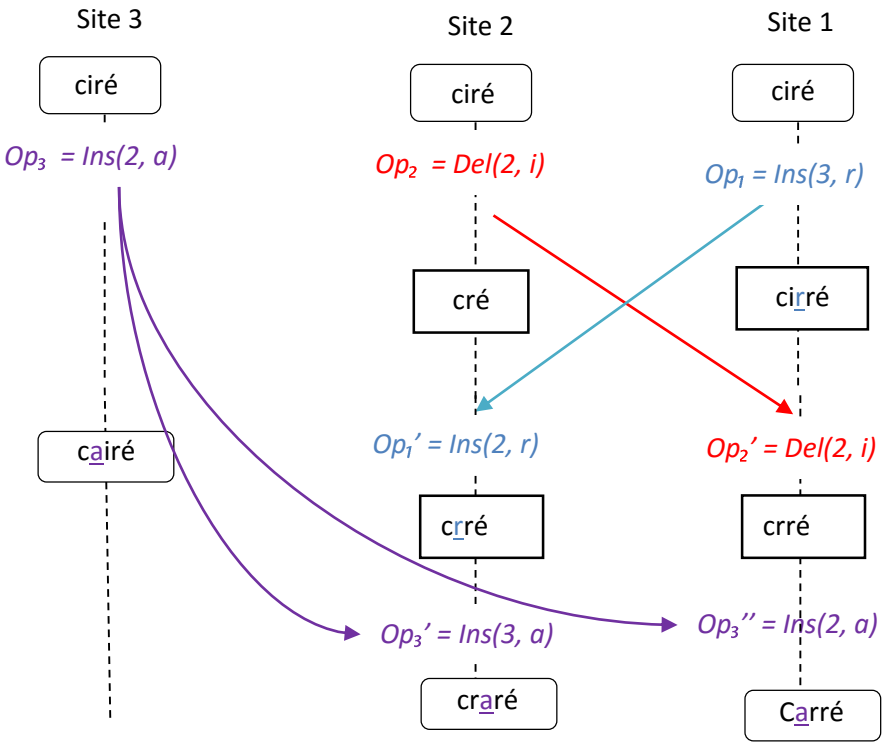



Figure 9 – Divergence de copies pour trois opérations concurrentes[1]

Au site 1, quand $Op_2$ arrive, elle est d'abords transformée par rapport à $Op_1$, i.e. $Op_2' = TI(Op_2, Op_1) = Del(2)$ et ensuite $Op_2'$ est exécuté donnant lieu à l'état "crré".
Sur le même site, l'intégration de $Op_3$ passe par la transformation par rapport à la séquence $[Op_1 ; Op_2']$ qui résulte en $Op_3'' = Ins(2,'a')$ dont l'exécution mène à l'état "carré".

Dans le site 2, $Op_1$ est en premier transformée par rapport à $Op_2$, i.e. $Op_1' = TI(Op_1, Op_2) = Ins(2,'r')$, et ensuite le résultat $Op_1'$ est exécutée produisant l'état "crré". Quand $Op_3$ arrive, elle est transformée par rapport à $[Op_2 ; Op_1']$ dont le résultat $Op_3' = Ins(3,'a')$ est ensuite exécuté produisant l'état "craré".

Nous constatons la transformation de $Op_3$ par rapport aux deux séquences équivalentes $[Op_1 ; Op_2']$ et $[Op_2 ; Op_1']$ donne deux opérations différentes, i.e $Op_3' = Ins(3,'a')$ et $Op_3'' = Ins(2,'a')$, ce qui induit à une divergence de données puisque les états des sites 2 et 3 sont différents.

En plus de TP1, la fonction TI doit satisfaire la condition suivante :

#### 2.3.3.2 Définition (Condition TP2)

Soient trois opérations concurrentes définies sur le même état Op, $Op_1$ et $Op_2$ .La fonction TI satisfait la condition TP2 ssi :

$$TI(TI(Op, Op_1), Op_2') = TI(TI(Op, Op_2), Op_1')$$

où $Op_1' = TI(Op_1, Op_2)$ et $Op_2' = TI(Op_2, Op_1)$.

La condition TP2 définit une identité d'opérations. Elle stipule que le résultat de la transformation d'une opération par rapport à une séquence d'opérations concurrentes ne dépend pas de l'ordre selon lequel les opérations de cette séquence ont été transformées.

## 2.4 Algorithmes d'intégration

Dans l'approche des transformées opérationnelles, chaque site est équipé de deux composants principaux : le composant d'intégration et le composant de transformation.

Le premier composant se présente comme un algorithme qui est responsable de recevoir, diffuser et exécuter les opérations. Il est indépendant de la sémantique des objets collaboratifs. Plusieurs algorithmes d'intégration ont été proposés dans le domaine des éditeurs collaboratifs, comme dOPT, adOPTed et SOCT2, et SOCT4. Le composant de transformation est un ensemble d'algorithmes de transformation qui est responsable de sérialiser deux opérations concurrentes définies sur le même état. Chaque algorithme de transformation est spécifique à la sémantique d'un objet collaboratif [1].

Chaque site génère séquentiellement les opérations et stocke ces opérations dans un journal d'opérations (ou historique). Lorsqu'un site reçoit une opération distante op, le composant d'intégration procède par les étapes suivantes :

1. Déterminer dans l'historique locale *h* les opérations qui sont concurrentes à *op*.
2. Utiliser le composant de transformation pour produire *op'* qui est la forme transformée de *op* par rapport aux opérations de *h* qui sont concurrentes à *op*.
3. Exécuter *op'* sur l'état courant du site.
4. Ajouter *op'* à l'historique locale.

Le composant d'intégration permet donc de construire l'historique des opérations exécutées sur le site, tout en préservant la relation causale entre les opérations. Lorsque le système est au repos, les historiques des différents sites ne sont pas forcément identiques mais elles sont équivalentes (elles aboutissent au même état final) malgré que l'ordre d'exécution des opérations concurrentes puisse être différent d'un historique à une autre. Cette équivalence ne peut être assurée que si l'algorithme de transformation utilisé satisfait les conditions TP1 et TP2 [1].

Plusieurs algorithmes, reposant sur le principe des transformées opérationnelles satisfaisant la convergence, ont été proposés tel que GOTO, Opt, ABT, mais malheureusement aucune des transformées proposées n'assure les propriétés TP1 et TP2. D'autres algorithmes proposés ont tout simplement des scénarios divergeant tels que SDT, Suleiman et Al.

## 2.4.1 Etude des Algorithmes Existants

### 2.4.1.1 AdOpted

Ressel en 1996 a donné un algorithme basé sur la notion de vecteur d'état. Un état correspond à un point dans un hypercube dont la dimension est le nombre de sites. Chaque site correspond à un axe et une opération effectuée par ce site est une translation de longueur 1 selon cet axe. Une suite d'opérations correspond à un chemin dans cet hyper cube.

La fonction d'intégration correspond à la prise en compte des opérations exécutées sur les autres axes et les propriétés permettent d'assurer que chaque site suit un chemin qui mène au même point. Les positions dans le cube expriment les dépendances entre les opérations.

Les contraintes imposées par cet algorithme telles que le maintien de l'histoire des opérations, le graphe multidimensionnel, le vecteur d'état renvoyé à chaque interaction et la recherche d'un chemin de transformation pour chaque requête posent des problèmes de gestion et d'occupation mémoire, de bande passante ainsi que des difficultés pour passer à l'échelle lorsque le nombre de collaborateurs augmente.

#### 2.4.1.2 COT

Cet algorithme associe à chaque opération un contexte qui est l'état du document lors de la création de l'opération. La condition TP2 est donnée par l'algorithme qui pour un même groupe de contexte va transformer les opérations dans un même ordre. Dans le pire des cas un ré-ordonnancement est requis. L'envoi du vecteur d'état, ici appelé contexte passe difficilement à l'échelle [1].

#### 2.4.1.3 Google Wave

Un projet de l'entreprise Google. Le principe est de pouvoir collaborer sur un document hiérarchique à l'image d'un forum où des personnes peuvent poster, répondre être éditer en temps réel. Il est basé sur l'algorithme de réconciliation nommé Jupiter, conçu pour fonctionner de façon centralisée. Le nombre de participants est directement lié à la taille du nuage et ce système n'est pas adéquat pour le vrai pair-à-pair.

#### 2.4.1.4 So6

Un projet développé par l'équipe ECOO de l'INRIA Lorraine, basé sur l'algorithme de SOCT4, il utilise un serveur qui estampille chaque message de façon unique avec sa date. Ainsi l'ordre d'intégration des messages est total. C'est une approche hybride qui repose sur une notion de temps partagé et correspond plus à une approche centralisée qu'à du vrai pair-à-pair, comme Google Wave.

## 2.5 Conclusion

Dans ce chapitre, nous avons donné une vue générale sur l'édition collaborative sur les documents. Nous avons aussi parlé du modèle des transformées opérationnelles, qui est utilisé pour sérialiser par transformation des opérations concurrentes en satisfaisant les conditions pour garantir la convergence des répliques dans les sites du système.

Cependant, l'écriture d'un algorithme de transformation est devenue une tâche fastidieuse puisqu'elle nécessite des vérifications qui sont très difficiles, et sans une approche formelle, la conception des algorithmes de transformation restera une activité ardue et sujette à erreurs,

étant donné le nombre important de cas à considérer pour garantir des propriétés aussi critiques que la convergence des copies.

Pour cette raison, Beaucoup d'algorithmes proposés tel que Google Wave, So6, ADOPTE etc., n'arrivent pas à satisfaire les propriétés pour permettre le passage à l'échelle.

# Chapitre 3

# Conception et Modélisation

## 3.1 Introduction

Après avoir détaillé la notion d'édition collaborative sur les documents, la phase de conception et de modélisation vient pour mieux éclaircir le travail présenté dans ce mémoire. Ce chapitre se divise en trois parties. La première partie consiste à présenter le modèle de coordination sur lequel nous exécuterons l'algorithme d'édition collaborative, la deuxième partie décrit la description des données du modèle du système collaboratif tel que les messages échangés et les types de collaboration. Ensuite, nous présenterons la modélisation du système par les diagrammes UML et enfin, nous clôturons ce chapitre par une conclusion.

### Motivation

Le contexte de notre travail porte sur les applications d'édition collaborative des documents à structure linéaire (texte) dans les environnements distribués (paire-à-paire). Afin d'améliorer la disponibilité des données dans ces environnements qui sont très dynamiques, ces applications suivent un mode de réplication optimiste. Les répliques d'un même document peuvent être modifiées par plusieurs utilisateurs à la fois. Toutefois, la cohérence à terme des répliques n'est pas toujours assurée après leurs mises à jour, puisque les modifications concurrentes effectuées sur les différentes répliques peuvent créer des divergences de copies (problème de concurrence partielle). De plus, l'ordre de réception arbitraire de ces modifications dans les environnements paire-a-paire cause un vrai problème de convergence.

Nous avons choisie d'utiliser l'algorithme d'intégration ADOPTED pour notre application d'édition collaborative dans les environnements paire-à-paire. Cet algorithme est parmi les premières solutions proposées dans la littérature pour palier aux problèmes de

divergences des documents partagés édités dans les environnements paire-à-paire. Cette algorithme fonctionne très bien avec deux sites, mais pose des énormes problèmes quand le nombre de site dépasse les deux. L'algorithme ADOPTED permet l'intégration des opérations de mise-à-jour mais ne permet pas de passer à l'échelle en termes de nombre d'utilisateurs vue aux données répliquées qui deviennent volumineuses quand le nombre d'utilisateurs augmente, ce qui nécessite plus de temps pour modifier et mettre-à-jour les répliques. En plus, cet algorithme pose aussi un problème de concurrence partielle. Ce problème se produit lorsque deux ou plusieurs utilisateurs génèrent des opérations de mise à jour simultanément sur deux états différents de document partagé. Pour une meilleure compréhension du fonctionnement de l'algorithme ADOPTED, vous pouvez consulter le travail de *HACHELAF* [11].

## Contribution

Notre travail consiste à proposer une application d'édition collaborative basée sur l'approche des transformée opérationnelle pour les environnements paire-à-paire. Cette application utilise les objets collaboratifs qui admettent une structure linéaire. Ces objets peuvent être assimilés à une liste finie d'éléments d'un type de données particulier où chaque élément peut être considéré comme un caractère, un paragraphe, une page, … etc.
Cette application utilise une ***version revisitée*** de l'algorithme d'intégration ADOPTED pour assurer la diffusion, la réception des mises à jour, ainsi que la détection des liens de causalités entres les mises a jours dépendantes.

Cette version revisitée de ADOPTED est basée sur une nouvelle structure de donnée afin d'éviter le problème de concurrence partielle qui fait défaut dans sa version originale.
Dans cette nouvelle version, nous avons modifié l'état du document ADOPTED (qui est un texte ordinaire) par une nouvelle structure de données. Cette structure permet d'associer pour chaque caractère du document un poids unique. Le document partagé est vus comme une séquence ordonnée de couple (caractère, poids) dont les caractères sont ordonnés suivant leurs poids uniques.

### 3.1.1 Modèle de Coordination :

Dans ce modèle, une requête *r* est définie en tant qu'un quadruplet (*u*, *k*, *v*, *op*) où :

- *u* est l'identifiant du site collaborateur émetteur de la requête ;
- *k* est le numéro séquentiel des requêtes générées localement ;
- *v* représente le vecteur d'état du site ;
- *op* est l'opération à être exécutée sur l'état partagé.

Deux types d'opérations sont permis :

1) *Ins (i, (c,ω))* : insérer le caractère *c* ayant le poids $\omega$ à la position *i*.

2) *Del(i)* : supprimer l'élément de la position *i*.

#### 3.1.1.1 Le poids de position unique :

Soit l'ensemble des identifiants des utilisateurs « UID ». On suppose que les identifiants des utilisateurs sont uniques, cela implique que, deux utilisateurs distincts n'aient pas les mêmes identifiants.

On définit le poids de position comme suit$\mathcal{W}$ = {(*p*, *u*) : *p* ∈ ℕ*, u ∈UID }

Où, *p* représente la valeur de la position et *u* représente l'identifiant de l'utilisateur qui a créé ce poids.

Pour un état de document S =$\omega_1\omega_2...\omega_L$on utilise les notations suivantes :

- |S| signifie la longueur de S.
- $\omega_i$ représente le poids de l'élément à la position « i » tel que : i ∈ ℕ (1 ≤ i ≤ L).

#### 3.1.1.2 Principe

Dans un ordonnancement des poids de position, la position *p* peut être dénotée par l($\omega$) et le l'identifiant du site peut être dénoté par r($\omega$). La relation d'ordre $\omega_1 < \omega_2$ est vérifiée dans l'une des situations suivantes :

- Si : $l(\omega_1) < l(\omega_2)$
- Si : $l(\omega_1) = l(\omega_2)$ et $r(\omega_1) < r(\omega_2)$

  Avec l = left : la partie gauche d'un poids = la position ;

  et r = right : la partie droite d'un poids = l'identifiant du site.

Soit $\omega_0^u$ = (1, u), le poids minimal que peut générer un utilisateur « u ». Comme pour deux poids $\omega_1$ et $\omega_2$ générés par un même utilisateur « u », le composant droit de la définition du poids minimal r($\omega_1$) = r($\omega_2$) = u est le même, de plus, pour n'importe quel poids $\omega$ est inférieur à son successeur immédiat $\omega'$= s($\omega$) car : s($\omega$) = ( l ($\omega$) +1 , r ($\omega_2$) ).

Donc pour un document S =$\omega_1\omega_2...\omega_L$ nous avons : $\omega_1 < \omega_2 <...<\omega_i <...<\omega_L$ et on peut dire alors, que chaque élément créé au niveau d'un site aura un poids unique.

Les utilisateurs peuvent modifier le document partagé en effectuant l'une des opérations permises :

## Opération d'insertion

Un nouveau poids doit être généré pour chaque opération d'insertion exécuté sur un document, afin que le document partagé résultant reste ordonné.

L'opération Ins (i, $\omega$ ) permet d'insérer un poids « $\omega$ » à la position « i » sachant que tous les poids à partir de cette position seront incrémentés et décalés d'une position vers la droite.

Pour un document S =$\omega_1\omega_2...\omega_L$deux cas d'insertion sont possibles :

- *Une insertion au début de « S »* : une insertion à la position « i = 1 » avec un poids minimal $\omega_0^u$= (1, u). Le reste des éléments se trouvant sur la droite de cette position seront automatiquement décalés d'une position pour obtenir un nouvel S = $\boldsymbol{\omega_0^u}\omega_1'\omega_2'...\omega_L'$ avec : $\omega_1'$= s ($\omega_1$), $\omega_2'$= s ($\omega_2$) … et $\omega_L'$= s ($\omega_L$).

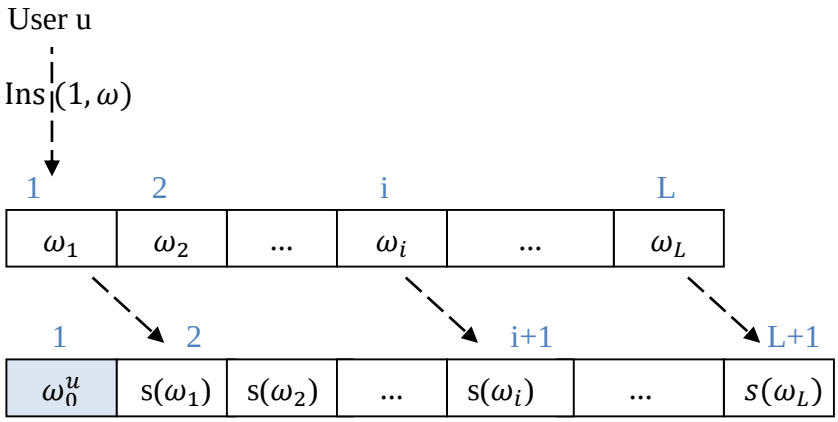



Figure 10 – Insertion d'un élément au début d'un document.

- *Une insertion ailleure :* une insertion à une position au milieu ou à la fin du document, un utilisateur « u » doit inserer son élément à la position « i » tel que $1 \leq i \leq L+1$. Le nouveau poids $\omega$ de la position « i » sera calculé par rapport à son prédécesseur immédiat $\omega_{i-1}$ tel que : $\omega = (l(\omega_{i-1}) + 1, u)$.

Les poids correspondants aux indices supérieurs à « i » seront décalés vers la droite comme le cas d'une insertion au début. On obtient ainsi un nouveau document :
$S = \omega_1 \dots \omega_{i-1} \boldsymbol{\omega} \; \omega'_i \dots \omega'_L$ avec $\omega'_i = s(\omega_i)$ et $\omega'_L = s(\omega_L)$.

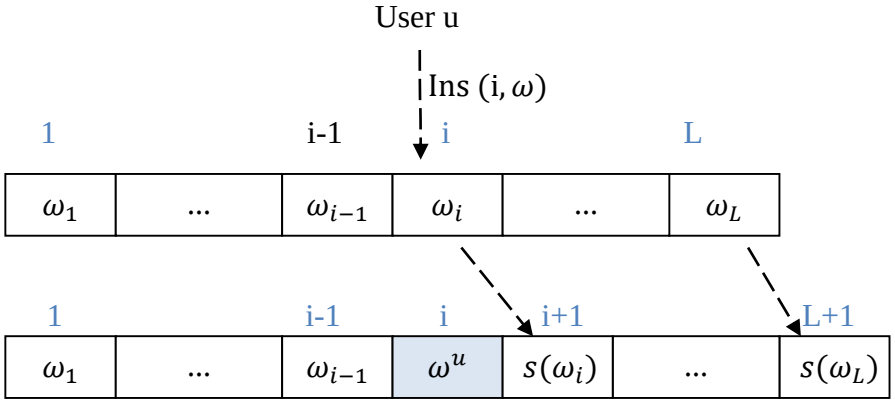



Figure 11 – Insertion d'un élément au milieu ou à la fin d'un document.

### 3.1.1.3 La suppression d'un élément :

Lors de la suppression d'un élément du document, l'utilisateur doit utiliser l'opération Del(i) qui supprime directement le poids qui se trouve à la position i.

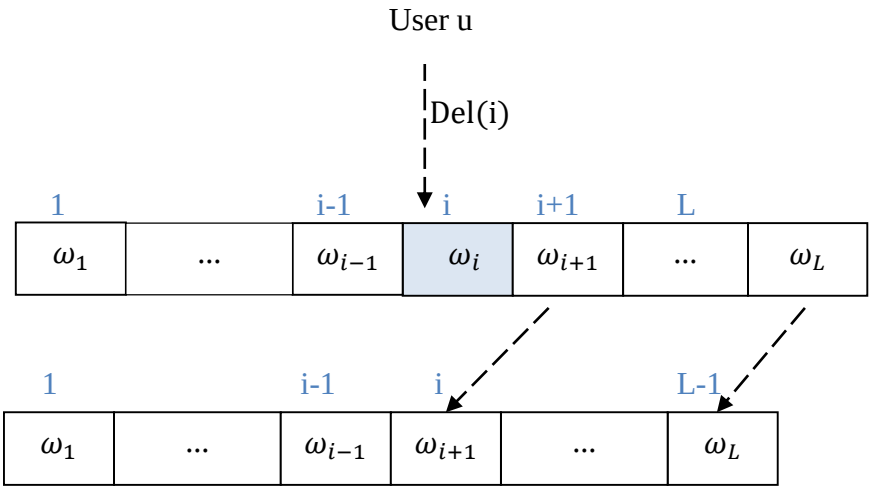



Figure 12 – Suppression d'un élément

### 3.1.2 Architecture du système de collaboration

Notre application d'édition collaborative est basée sur un modèle de réplication optimiste dans un environnement distribué, où chaque utilisateur dispose d'une réplique (copie) du document partagé. Cette application est destinée aux milieux collaboratifs synchrone où la communication entre les utilisateurs est favorisé d'être élaborée sans perte d'informations.
Chaque utilisateur est invité à modifier volontairement et à n'importe quel moment la réplique qu'il a et se communiquer avec les autres pairs directement sans passer par une entité intermédiaire ni par un site central.

Suite aux modifications générées par chaque site du réseau, chaque modification effectuée doit être traitée pour que la mise-à-jour des répliques distantes soit garantie.
L'état de chaque réplique est défini par l'ensemble des requêtes générées et reçues au niveau d'un site et l'ordre de l'éxecution de ces requêtes sur ce site.
La Figure 13 présente comment une requête locale est générée (étapes 1-4) et comment elle est reçue comme une requête distante (étapes 5-9) :

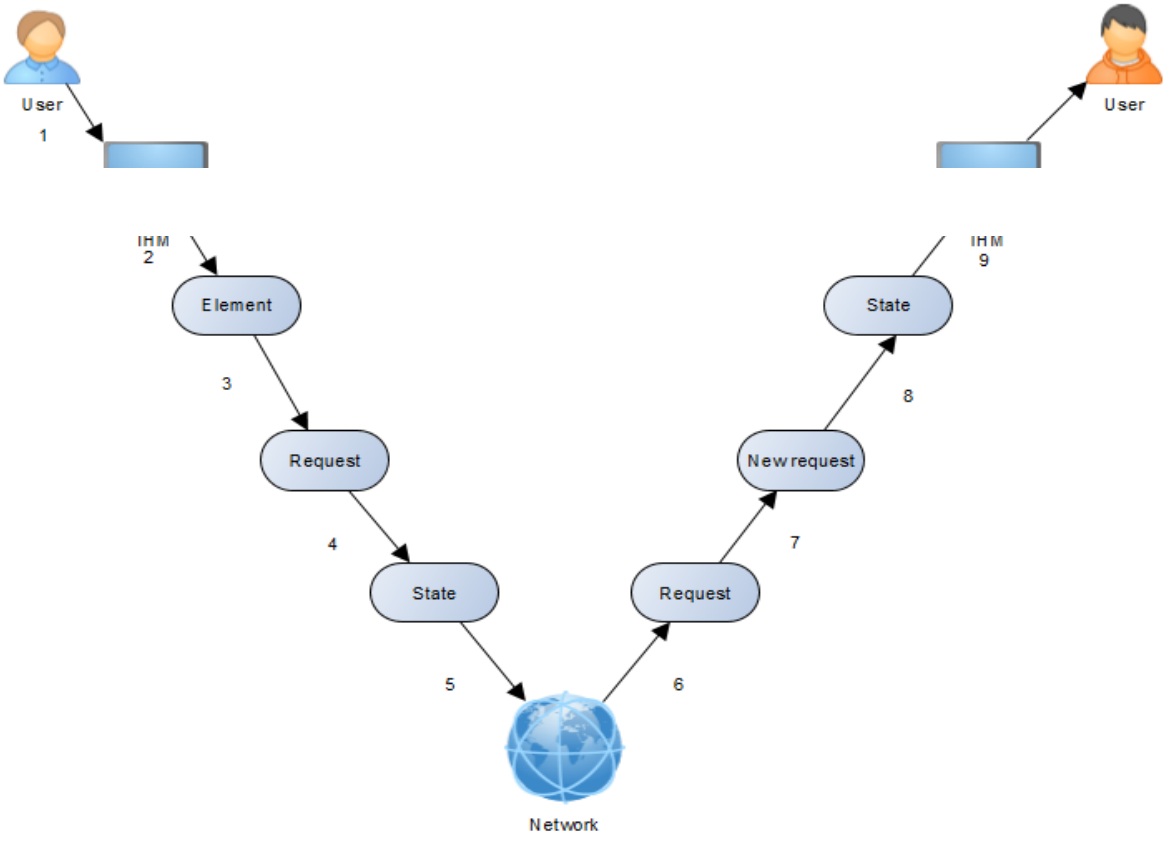



Figure 13 – Génération et intégration d'une requête.

1- **Génération d'une opération :** La réplique de chaque site est une séquence de caractères que l'utilisateur peut la modifier en générant des opérations d'insertion ou de suppression.

2- **Création d'une requête :** lors de la modification de la réplique à travers une opération générée par l'interface graphique de l'utilisateur, le système détecte s'il s'agit d'une opération d'insertion ou de suppression. Dans le cas d'une insertion, le système récupère le caractère qui vient d'être inséré ainsi que sa position d'insertion. Dans l'autre cas où il s'agit d'une opération de suppression, nous devons récupérer la position de l'effacement et retrouver le caractère qui doit être supprimé.

3- **Génération d'un nouveau poids** : lors d'une insertion, la requête entraine la création d'un nouveau poids $\omega$ pour l'élément à inserer. Ce poids est calculé à partir de la position d'insertion « i » récupérée dans l'étape précédante de la manière décrite sur les **Figures** (28 et 29) .

4- **Exécution d'une requête :** une fois le nouveau poids est généré, le système applique la requête localement. Chaque requête aura le même type que celui de l'opération utilisée pour sa création. Suite à chaque execution d'une requête, l'état d'une réplique change et le résultat « S(O) » obtenu après l'exécution de l'opération O sur l'état S = $\omega_1\omega_2...\omega_L$ de la réplique peut être résumé comme suit :

$$S(O) \begin{cases} \omega_1 ... \ \omega_{i-1}\boldsymbol{\omega^u}\ s(\omega_i) ... \ s\ (\omega_L) & \text{Si O = Ins (i, } \boldsymbol{\omega^u}\text{), } i \leq L+1 \\ & \text{et } \ \omega_{i-1} \leq \boldsymbol{\omega^u} \leq \text{succ}(\omega_i) \\ \omega_1 ... \omega_{i-1}\omega_{i+1} ... \omega_L & \text{Si O = Del(i) et } i \leq L \\ \text{N'est pas définie} & \text{Sinon} \end{cases}$$

Avec succ($\omega_i$) le successeur du poids $\omega_i$

5- **Envoi de la requête locale :** après l'étape de l'exécution de la requête, le système envoi la requête à tous les autres utilisateurs collaborateurs. Les sites distants seront chargé de trouver une position d'insertion qui permet de garder la cohérence du document partagé.

6- **Intégration d'une requête :** le mode de collaboration utilisé dans l'application est synchrone, de ce fait le système doit rester toujours en écoute pour collecter les opérations propagées sur le réseau. Dès qu'une requête est détectée, son intégration dans le système doit être garantie. Comme l'ordre de réception des opérations peut être différent de leur ordre de génération, le système doit s'en charger d'analyser les différentes requêtes reçues et de décider quelle requête à exécuter en respectant la causalité.

7- **Transformation d'une requête :** le but de cette phase est de préserver les intentions des utilisateurs et de garder l'ordonnacement des éléments de chaque réplique. Afin de réaliser ça, une transformation est nécéssaire d'être mise sur l'ensemble des opérations concurrentes.

Pour intégrer une opération de suppression Del(i), le caractère de la position « i » sera supprimé quel que soit le poids qu'il possède.

Pour une opération Ins (i,$\omega^u$), l'élément à insérer doit être mis à la position « i » qui se situe entre « i-1 »et « i+1 ». Les caractères se situant à ces positions peuvent être supprimés précédemment ou avoir d'autres caractères insérés en concurrence. La procédure de transformation doit réaliser un ordonnancement en se basant sur la position « i » ainsi que le paramètre de position unique $\boldsymbol{\omega^u}$ pour obtenir un nouveau poids $\boldsymbol{\omega^u}$ vérifiant la relation d'ordre : $\omega_{i-1} \leq \boldsymbol{\omega'} \leq \text{succ}\ (\omega_i)$.

8- **Exécution d'une requête :** après la transformation de la requête, son exécution se fera comme l'exécution d'une requête locale car le nouveau poids calculé permet de vérifier les conditions d'exécution.

9- **Synchronisation de l'état :** avec chaque requête exécutée, le système doit actualiser l'état de la réplique pour fournir à l'utilisateur la dernière version du document partagé.

### 3.1.2.1 Exemple illustratif :

Soit l'exemple suivant qui met en évidence ce système de poids unique. Chaque flèche indique l'intégration d'une opération au niveau du site.

Soient Site1, Site2 et Site3 trois utilisateurs rentrant en une session collaborative pour éditer un même article dont l'état initial est vide et soit le scénario suivant :

- Une Opération O1 = Ins (1, ('Y', (1,2))) est générée par le site2. Le caractère « Y » est inséré à la première position accompagné par son poids minimal $\omega_0^2$= (1,2) sur le document partagé. Cette opération sera envoyée vers les autres sites (1 et 3) pour garantir la mise à jour de leurs répliques.

  Une fois l'opération O1 est exécutée sur tous les sites, toutes les répliques sont convergentes.

- Les Opérations O2, O3 et O4 sont simultanément générés par les trois sites 1, 2 et 3 successivement,
  - O2 = Ins (1, (‘Z’, (1, 1))) ;
  - O3 = Del (1) ;
  - O4 = Ins (2, (‘U’, (2, 3))).

  Après la mise à jour tous les sites, on obtient des répliques convergentes vers la même valeur « ZU ».

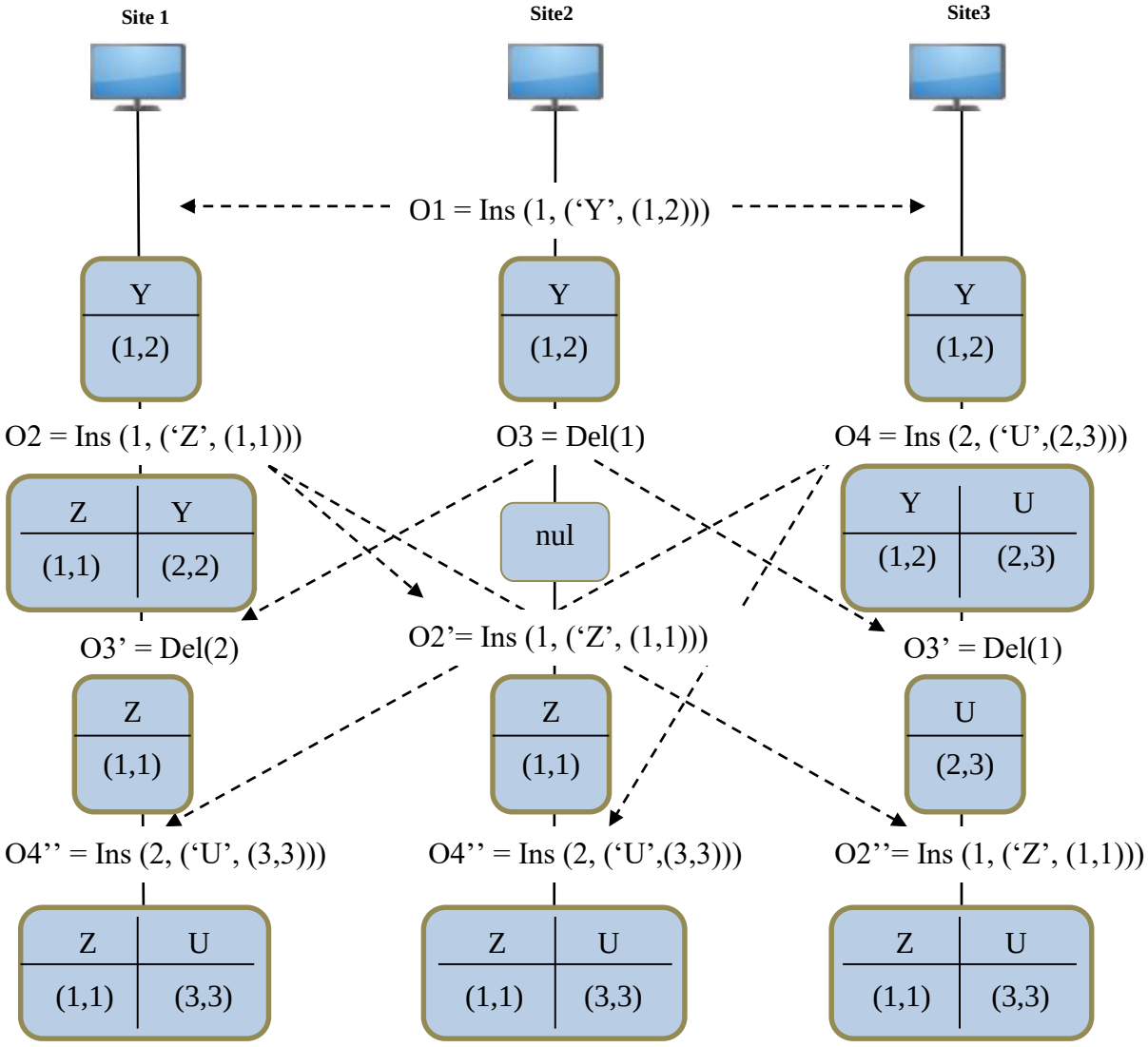



Figure 14 – édition concurrente basée sur l’ordre des poids des pairs

**Site1 :**

- L'opération O1 est intégrée et on obtient l'état « Y » ;
- L'opération O2 insère le caractère « Z » à la 1ère position. Le poids généré pour cette opération sera : $\omega_0^1 = (1,1)$.
- Ainsi, le poids du caractère « Y » sera décalé d'une position vers la droite et on obtient la chaine « ZY ».
- L'opération O3 sera transformée en O3' car elle est venue en concurrence par rapport à O2, telle que O3' = IT (O3, O2) = Del (2).
- L'opération O3' supprime le caractère « Y » sans modifier le poids. On obtient ainsi l'état « Z ».
- L'opération O4 est aussi venue en concurrence après O2 et O3', elle est donc transformée en O4" par rapport à la séquence [O2, O3'].
- L'opération O4' = Ins (3, ('U', (3, 3))) est le résultat de transformation de O4 par rapport à O2.

$$\text{IT (O4, O2) = IT (} \overbrace{\text{Ins (2,('U',(2,3)))}}^{\text{O4}} \text{ , } \overbrace{\text{Ins (1,('Z',(1,1)))}}^{\text{O2}} \text{ ) = } \overbrace{\text{Ins (3,('U',(3,3)))}}^{\text{O4'}}$$

- L'opération O4" = Ins (2, ('U', (3,3)) ) est le résultat de transformation de O4' par rapport à O3'.

$$\text{IT (O4', O3') = IT (} \overbrace{\text{Ins (3,('U',(3,3)))}}^{\text{O4'}} \text{ , } \overbrace{\text{Del(2)}}^{\text{O3'}} \text{ ) = } \overbrace{\text{Ins ( 2,('U',(3,3)) )}}^{\text{O4"}}$$

- L'opération O4" insère le caractère « U » à la 2ème position pour avoir enfin l'état « ZU ».

**Site2 :**

- L'opération O1 est intégrée et on obtient l'état « Y » ;
- L'opération O3 supprime le caractère « Y »et l'état devient vide.
- L'opération O2 sera transformée en O2' par rapport à O3 car elle est venue en concurrence, elle devient O2' = Ins (1, ('Z', (1, 1))).

$$IT\ (O2, O3) = IT\ (\overbrace{Ins\ (1, (‘Z’, (1, 1))}^{O2}), \overbrace{Del\ (1)}^{O3}) = \overbrace{Ins\ (1, (‘Z’, (1, 1)))}^{O2'}$$

- L’opération O2’ insère le caractère « Z » à la 1ère position et permet d’avoir l’état « Z ».
- L’opération O4 est transformée par rapport à la séquence [O3, O2'].
- L'opération O4' = Ins (3, (‘U’, (3, 3))) est le résultat de transformation de O4 par rapport à O3.

$$IT\ (O4, O3) = IT\ (\overbrace{Ins\ (2,(‘U’,(2,3)))}^{O4}, \overbrace{Del(1)}^{O3}) = \overbrace{Ins\ (1,(‘U’,(2,3)))}^{O4'}$$

- L'opération O4" = Ins (2, (‘U’, (3,3)) ) est le résultat de transformation de O4' par rapport à O2'.

$$IT\ (O4', O2') = IT\ (\overbrace{Ins\ (2,(‘U’,(2,3)))}^{O4'}, \overbrace{Ins\ (1,(‘Z’,(1,1)))}^{O2'}) = \overbrace{Ins\ (3,(‘U’,(3,3)))}^{O4''}$$

- L'opération O4" insère le caractère « U » à la 2ème position pour avoir enfin l’état « ZU ».

**Site3 :**

- L’opération O1 est intégrée et on obtient l’état « Y » ;
- L'opération O4=Ins (2, (‘U’, (2,3))) insère le caractère « U » et l’état devient « YU ».
- L'opération O3 est transformée par rapport à O4, on obtient O3' = IT(O3, O4) = Del(1);
- L'opération O3' supprime le caractère « Y » sans modifier le poids. On obtient ainsi l’état « U ».
- L’opération O2 est transformée par rapport à la séquence [O4 , O3’].
- L'opération O2' = Ins (3,(‘U’,(3,3))) est le résultat de transformation de O2 par rapport à O4.

$$IT\ (O2, O4) = IT\ (\overbrace{Ins\ (1,(‘Z’,(1,1)))}^{O2}, \overbrace{Ins\ (2,(‘U’,(2,3)))}^{O4}) = \overbrace{Ins\ (1,(‘Z’,(1,1)))}^{O2'}$$

- L'opération O2" = Ins (1,(‘Z’,(1,1))) est le résultat de transformation de O2' par rapport à O3'.

O2' O3' O2"

**IT (O2', O3') = IT ( Ins (1,('Z',(1,1))) , Del(1) ) = Ins (1,('Z',(1,1)))**

- L'opération O4" insère le caractère « U » à la 2ème position pour avoir enfin l'état « ZU ».
- L'opération O2" insère le caractère « Z » à la 1ère position pour avoir enfin l'état « ZU ».

## 3.2 Modèle du système

### 3.2.1 Description des données du système

Pour modéliser un système, on doit lui présenter ses paramètres d'entrées. Notre projet à besoin des données qui décrivent :

#### 3.2.1.1 Les nœuds du système

Il existe deux types de nœuds dans le système :

**Les utilisateurs :**

Chaque site du système est représenté par un pair ayant les capacités de communiquer, éditer, partager, créer de nouveaux articles et joindre une édition collaborative.

**Les collaborateurs :**

Un utilisateur *c* est dit collaborateur d'un utilisateur *u*, s'ils rentrent en édition collaborative du même article. Les collaborateurs sont détectés grâce aux échanges des messages indiquant le même article. Deux pairs qui n'éditant pas le même article ne peuvent être collaborateurs.

#### 3.2.1.2 Les messages

Il existe six types de messages qui circulent dans le réseau :

- Message de création d'un nouvel article à éditer.
- Message de chat.
- Un message pour mettre à jour un paragraphe d'un article.
- Message pour créer un nouveau paragraphe.
- Un message pour mettre à jour le titre des paragraphes.

Leurs structures sont les suivantes :

- ***Message de création d'un nouvel article à éditer* :** est de la forme suivante :

| *@Source* | *@ Récepteur* | *Obj_article* | *Type* |
|---|---|---|---|

  - ✓ ***@Source*** : représente l'adresse du nœud initiateur du message.
  - ✓ ***@Récepteur*** *:* l'adresse du nœud qui va recevoir le message.
  - ✓ ***Obj_article*** *:* Un objet de classe Articles qui porte toutes les propriétés de l'article tel que son nom, son identifiant, la date de sa création et la liste des éditeurs.
  - ✓ ***Type :*** détermine que c'est un message de création d'article.

- ***Message de chat*:** il contient les champs suivants :

| *@Source* | *@ Récepteur* | *Msg* | *Type* |
|---|---|---|---|

  - ✓ ***@Source*** : représente l'adresse du nœud initiateur du message.
  - ✓ ***@Récepteur*** *:* l'adresse du nœud qui va recevoir la requête.
  - ✓ ***Msg*** *:* Contenu textuel du message du chat.
  - ✓ ***Type :*** détermine que c'est un message de chat.

- ***Un message pour mettre à jour un paragraphe d'article*:** est de la forme suivante :

| *@Source* | *@ Récepteur* | *ID_article* | *ID_para* | *Obj_op* | *Type* |
|---|---|---|---|---|---|

  - ✓ ***@Source*** : représente l'adresse du nœud initiateur du message.
  - ✓ ***@Récepteur*** *:* l'adresse du nœud qui va recevoir la requête.
  - ✓ ***ID_article*** *:* L'ID de l'article contenant le paragraphe à modifier.
  - ✓ ***ID_para*** *:* L'ID du paragraphe à mettre à jour.
  - ✓ ***Obj_op*** *:* Un objet de classe Opération portant toutes les propriétés de l'opération tel que le type "ajout ou suppression", la position et le caractère.
  - ✓ ***Type :*** détermine que c'est un message de mise à jour d'un paragraphe.

- ***Message pour créer un nouveau paragraphe* :** est de la forme suivante :

| *@Source* | *@ Récepteur* | *ID_article* | *Type* |
|---|---|---|---|

- ✓ ***@Source*** : représente l'adresse du nœud initiateur du message.
- ✓ ***@Récepteur :*** l'adresse du nœud qui va recevoir le message.
- ✓ ***ID_article :*** L'ID de l'article auquel on souhaite ajouter un nouveau paragraphe.
- ✓ ***Type :*** détermine que c'est un message de création d'un paragraphe.

- ***Un message pour mettre à jour le titre d'un paragraphe* :** est de la forme suivante :

| *@Source* | *@ Récepteur* | *ID_article* | *ID_para* | *Titre_para* | *Type* |
|---|---|---|---|---|---|

- ✓ ***@Source*** : représente l'adresse du nœud initiateur du message.
- ✓ ***@Récepteur :*** l'adresse du nœud qui va recevoir le message
- ✓ ***ID_article :*** L'ID de l'article.
- ✓ ***ID_para :*** L'ID de paragraphe.
- ✓ ***Titre_para :*** Le contenu du nouveau titre du paragraphe.
- ✓ ***Type :*** détermine que c'est un message de mise à jour du titre du paragraphe.

### 3.2.2 Les outils du système

#### Un buffer article-contacts :

Chaque pair du réseau possède un buffer d'article-contacts qui contient les articles et les pairs qui ont participé à son édition. Ce buffer est une structure de données dynamique (tableau 1).

**Tableau 1 – Article-Contacts**

| Article | Pairs |
|---|---|
| A1 | User 1, User 2, User 3 |
| A2 | User 4, User 5 |
| A3 | User6 |
| ……. | |
| An | User n |

#### Un buffer Contact-Requêtes

Chaque pair du réseau possède un buffer de requêtes qui lui ont été envoyées. Ce buffer est une structure de données dynamique (tableau 2).

**Tableau 2 – Contact-Requêtes**

| Contacts | Requêtes |
|---|---|
| User 1 | R1, R2, R3 …, |
| User 2 | R2, R1, R3 …, Rm" |
| User 3 | R1, R3, R2 …, Rm' |
| ……. | …… |
| User n | R3, R2, R1, …, Rm"' |

## 3.3 Modélisation

Le Modèle conceptuel de données est une représentation statique du système d'information. Il a comme objectif de constituer une représentation claire et cohérente des données manipulées dans le système d'information. Cette section, sera présentée comme suit : nous commençons par le choix de la méthodologie de conception et justification. Ensuite nous identifions les acteurs et les diagrammes des cas d'utilisation, puis nous présentons le diagramme de classe, diagramme de collaboration et enfin les diagrammes d'état transition.

### 3.3.1 Présentation d'UML :

UML (Unified Modeling Language) est un langage formel et normalisé en termes de modélisation objet. Son indépendance par rapport aux langages de programmation, aux domaines de l'application et aux processus, son caractère polyvalent et sa souplesse ont fait lui un langage universel. En plus UML est essentiellement un support de communication, qui facilite la représentation et la compréhension de solution objet. Sa notation graphique permet d'exprimer visuellement une solution objet, ce qui facilite la comparaison et l'évaluation des solutions. L'aspect de sa notation limite l'ambigüité et les incompréhensions.

UML fournit un moyen astucieux permettant de représenter diverses projections d'une même représentation grâce aux vues. Une vue est constituée d'un ou plusieurs diagrammes. On distingue deux types de vues : La vue statique, permettant de représenter le système physiquement et la vue dynamique, montrant le fonctionnement du système.

## 3.3.2 Description de la vue statique

### 3.3.2.1 Diagrammes des cas d'utilisation

Les cas d'utilisation décrient un ensemble d'actions réalisées par le système, en réponse à une action d'un acteur.

#### a) Identification des acteurs

Les utilisateurs sont les seuls acteurs qui interagissent avec le système.
Utilisateur : édite des articles en collaboration avec les autres.

#### b) Identification des cas d'utilisation

Chaque acteur a les cas d'utilisation suivants :

| Acteur | Rôle |
|---|---|
| Editeur d'article<br>(Utilisateur) | - S'authentifier<br>- Créer un article<br>- Consulter la liste des articles existant<br>- Joindre un article<br>- Consulter la liste des contacts<br>- Ajouter un paragraphe<br>- Modifier un paragraphe<br>- Consulter les articles par un navigateur web<br>- Chat avec ses contactes |

#### c) Diagramme de cas d'utilisation globale

Ci-dessous, nous présentons le diagramme de cas d'utilisation pour la compréhension du fonctionnement du système.


Figure 15 – Diagramme de cas d'utilisation générale pour l'utilisateur

### c.1) Description du cas d'utilisation créer article

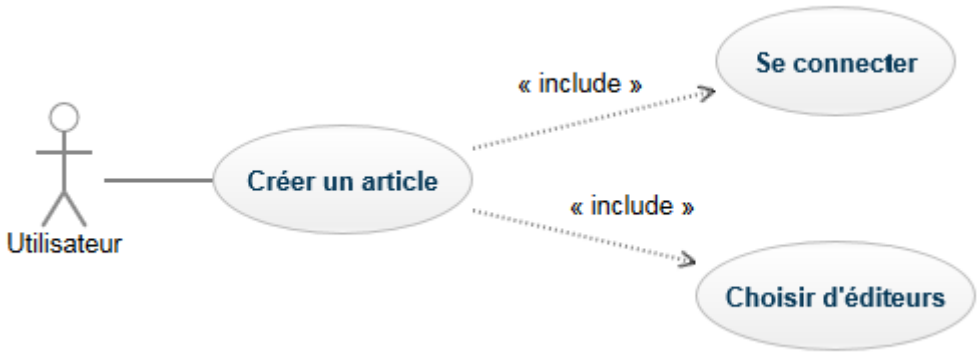



Figure 16 – Diagramme de cas d'utilisation « créer article »

**Tableau 3 – Fiche de description du cas d'utilisation : créer article**

| | |
|---|---|
| Titre | Cas d'utilisation « Créer Article » |
| But | Créer un article |
| Résumé | L'utilisateur demande la création d'un article en présentant son titre et la liste des éditeurs. |
| Acteur | L'Utilisateur |
| Pré Conditions | L'utilisateur est authentifié |
| Post conditions | Créer un article et inviter les éditeurs |
| Scénario nominal | DEBUT<br>• L'utilisateur se connecte à l'application.<br>• L'utilisateur demande la création d'un article via un bouton correspondant.<br>• Le système invite l'utilisateur à écrire le nom de l'article à créer.<br>• L'utilisateur écrire et valide le nom d'article<br>• Le système affiche une liste des éditeurs et invite l'utilisateur à choisir les participants.<br>• L'utilisateur choisit les participants et valide la création de l'article à créer.<br>• Le système crée l'article et invite les éditeurs à joindre l'article.<br>FIN |
| Scénarios Alternatifs | • Si l'utilisateur ne remplit pas le nom d'article, le système affiche une erreur et demande à l'utilisateur de remplir le champ.<br>• En cas où l'utilisateur ne choisit aucun éditeur et valide la création de l'article, le système affiche une erreur et demande à l'utilisateur de choisir au moins un éditeur avant de valider. |

### c.2) Description du cas d'utilisation ajouter Paragraphe

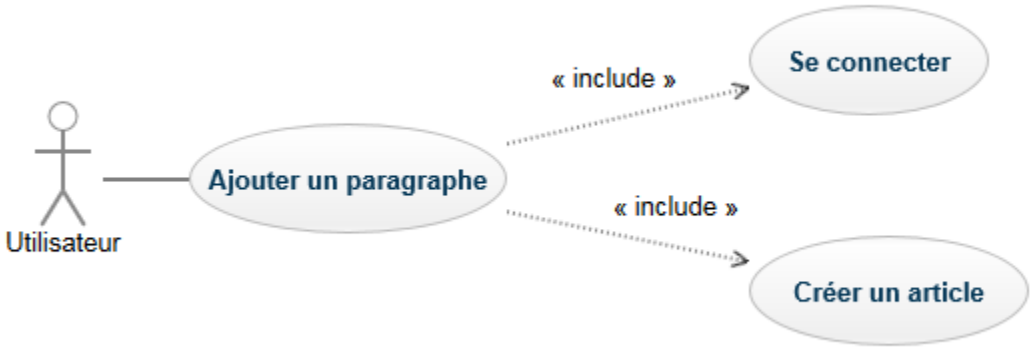



Figure 17 – Diagramme de cas d'utilisation « ajouter Paragraphe »

**Tableau 4 – Fiche de description du cas d'utilisation : ajouter paragraphe**

| | |
|---|---|
| Titre | Cas d'utilisation « ajouter Paragraphe » |
| But | Ajouter un Paragraphe et partage avec autre d'éditeurs |
| Résumé | L'utilisateur clique sur un bouton d'ajout de Paragraphe, Une fois la création validée, le paragraphe est créé et diffusé aux autres éditeurs. |
| Acteur | L'Utilisateur |
| Pré Conditions | L'utilisateur est authentifié et appartient à une édition collaborative |
| Post conditions | Créer un Paragraphe et le diffuser aux autres éditeurs |
| Scénario nominal | DEBUT<br>• L'utilisateur se connecte à l'application.<br>• L'utilisateur demande l'ajout d'un paragraphe via un bouton correspondant.<br>• Le système crée le paragraphe et le diffuse avec les autres éditeurs.<br>FIN |

### c.3) Description du cas d'utilisation « consulter liste des articles »


Figure 18 – Diagramme de cas d'utilisation « consulter liste articles »

**Tableau 5 – Fiche de description du cas d'utilisation : Consulter liste des articles**

| | |
|---|---|
| Titre | Cas d'utilisation « consulter liste articles » |
| But | Afficher la liste des articles |
| Résumé | Le client demande au système la liste des articles via le menu |
| Acteur | L'Utilisateur |
| Pré Conditions | L'utilisateur est authentifié |
| Post conditions | Afficher la liste des articles |
| Scénario nominal | DEBUT<br>• L'utilisateur se connecte à l'application.<br>• L'utilisateur demande la liste des articles via un élément correspondant à partir du menu.<br>• Le système charge tous les articles existants et les affiche sous forme d'une liste<br>• Une fois l'utilisateur choisit un article, il sera redirigé vers son éditeur d'article.<br><br>FIN |

### c.4) Description du cas d'utilisation « consulter l'article via navigateur web »


Figure 19 – Diagramme de cas d'utilisation « consulter l'article via navigateur web »

**Tableau 6 – Fiche de description du cas d'utilisation : consulter l'article via navigateur web**

| | |
|---|---|
| Titre | Cas d'utilisation « consulter l'article via navigateur web » |
| But | Afficher l'article via navigateur web |
| Résumé | Le client demande au système d'afficher un article détaillé via le web |
| Acteur | L'Utilisateur |
| Pré Conditions | L'utilisateur est authentifié et demande la liste des articles |
| Post conditions | Afficher l'article via navigateur web |
| Scénario nominal | DEBUT<br>• L'utilisateur se connecte à l'application.<br>• L'utilisateur demande la liste des articles via le menu.<br>• Le système affiche la liste d'article<br>• L'utilisateur sélectionne l'article qu'il souhaite consulter à travers un navigateur web.<br>• Le system charge les données de l'article et ouvre la page dans un navigateur web.<br>FIN |

### c.5) Description du cas d'utilisation « tchatcher avec d'autres éditeurs »

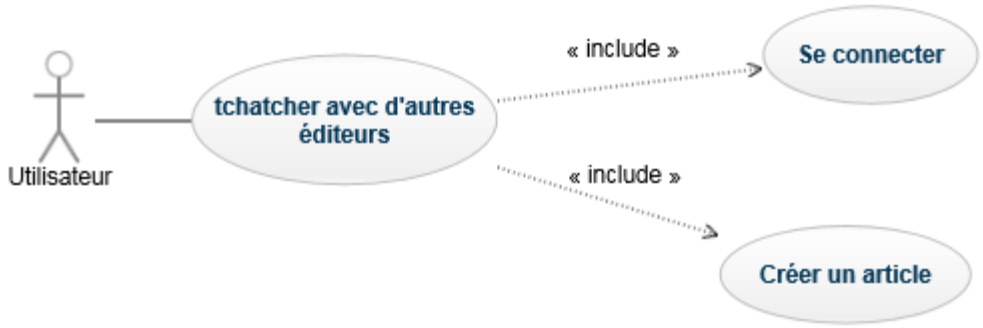



Figure 20 – Diagramme de cas d'utilisation « tchatcher avec d'autres éditeurs »

**Tableau 7 – Fiche de description du cas d'utilisation : tchatcher avec d'autres éditeurs**

| | |
|---|---|
| Titre | Cas d'utilisation « tchatcher avec d'autres éditeurs » |
| But | Chat avec les personnes éditant le même article |
| Résumé | L'utilisateur saisir un message puis click sur bouton envoyer. Le message est diffusé avec d'autres éditeurs |
| Acteur | L'Utilisateur |
| Pré Conditions | L'utilisateur est authentifié et fait partie d'une édition collaborative d'un article |
| Post conditions | Ouvrit une fenêtre de chat avec les autres éditeurs |
| Scénario nominal | DEBUT<br>• L'utilisateur se connecte à l'application.<br>• L'utilisateur rentre en mode d'édition d'un article<br>• L'utilisateur ouvre la fenêtre du chat et saisit le message dans le champ bloc chat et puis click sur le bouton envoyer<br>• Le système diffuse le message avec autre éditeurs<br>FIN |

### d) Diagramme de classes

La figure ci-dessous récapitule les tableaux précédents dans un diagramme de classes qui contient toutes les informations telles que les classes, les méthodes, les associations et les propriétés.

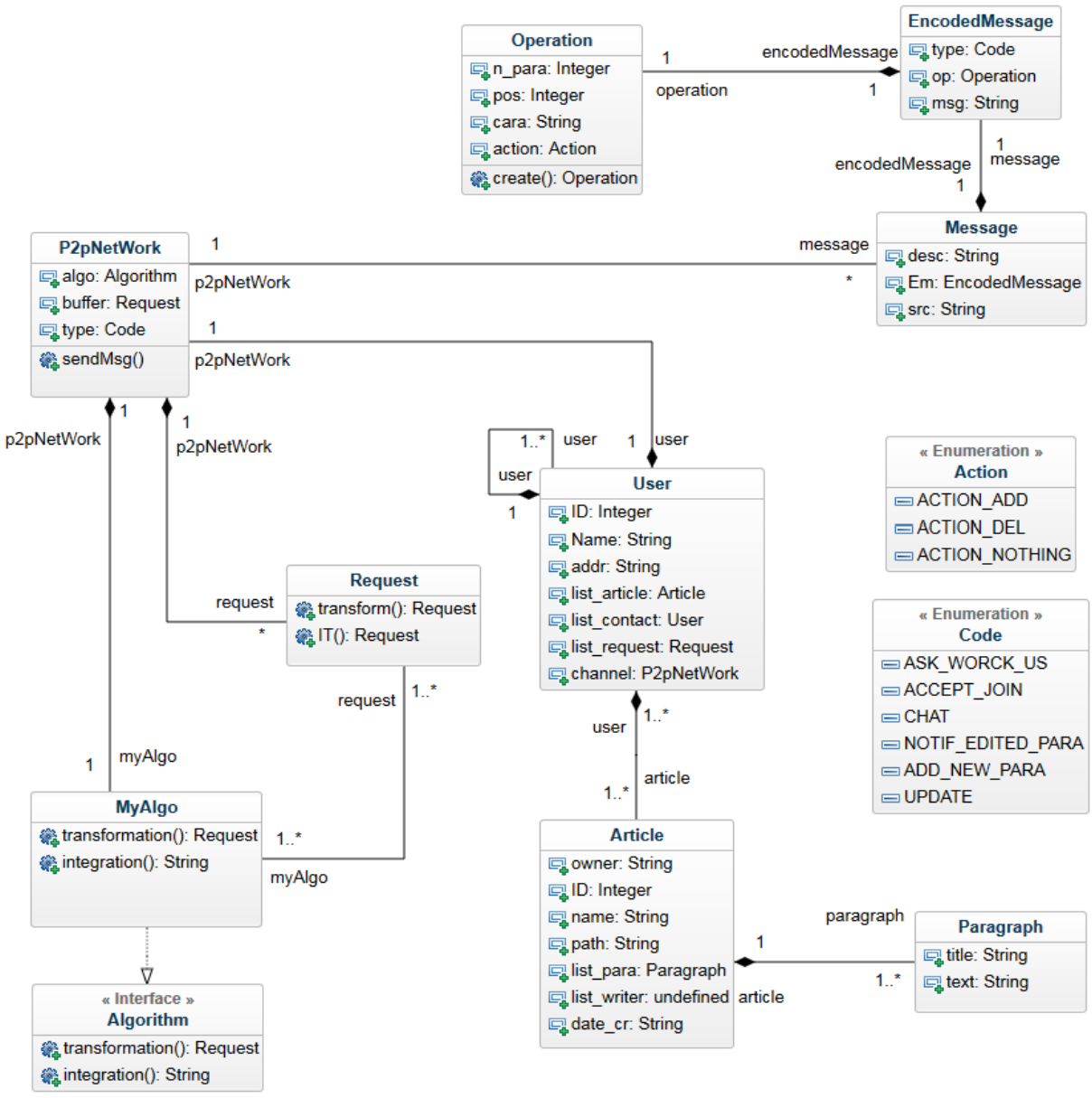



Figure 21 – Diagramme de classes associé à l'application

### 3.3.3 Description de la vue dynamique

#### 3.3.3.1 Diagrammes de séquences

Les diagrammes de séquences représentent les interactions entre les objets en indiquant la chronologie des séquences. Les diagrammes de séquences ajoutent une dimension temporelle aux diagrammes de collaborations.

**a) Diagramme de séquence : « S'authentifier »**

Le diagramme de séquence « S'authentifier » présente le séquencement des interactions entre utilisateur et l'interface d'authentification. L'utilisateur lance l'application et accède l'interface de connexion. In introduit son nom et valide via un bouton correspondant. Le système génère un ID utilisateur et cherche les fichiers locaux correspondants à cet ID, s'il trouve il les charges. Enfin l'utilisateur est redirigé vers l'interface d'accueil.

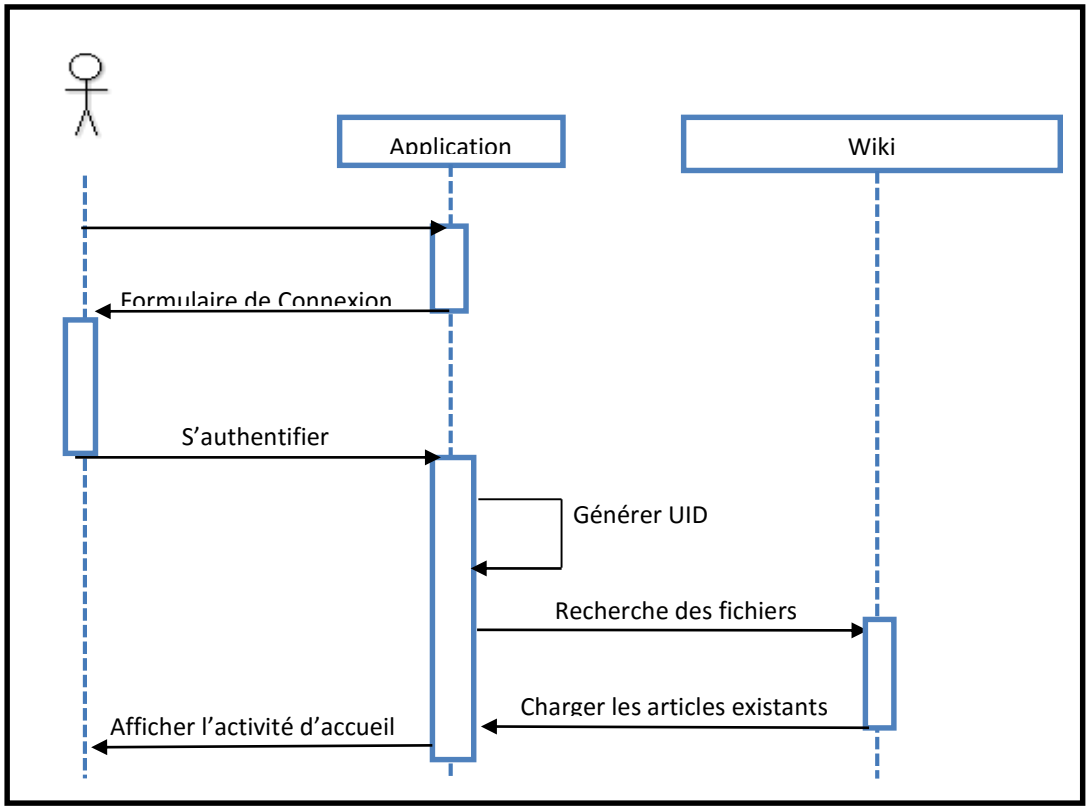



Figure 22 – Diagramme de séquence relatif à « S'authentifier »

### b) Digramme de séquence : « Ajouter Article »

Le diagramme de séquence « Ajouter article » présente le séquencement des interactions entre les utilisateurs créateurs, les utilisateurs collaborateurs, l'interface nouveau article, le Wiki et l'interface d'éditeur d'article.

Un utilisateur peut consulter, modifier ou ajouter un article via l'interface d'accueil. Pour l'ajout d'un article, l'utilisateur créateur demande la création d'un article, l'interface s'affiche en invitant l'utilisateur d'introduire le titre et la liste des éditeurs collaborateurs parmi la liste des contacts.

L'utilisateur valide une fois les champs saisis et le système vérifies ces données, crée un fichier article dans le Wiki local, lance l'éditeur des articles et notifie les collaborateurs de l'article sur lequel ils vont travailler.

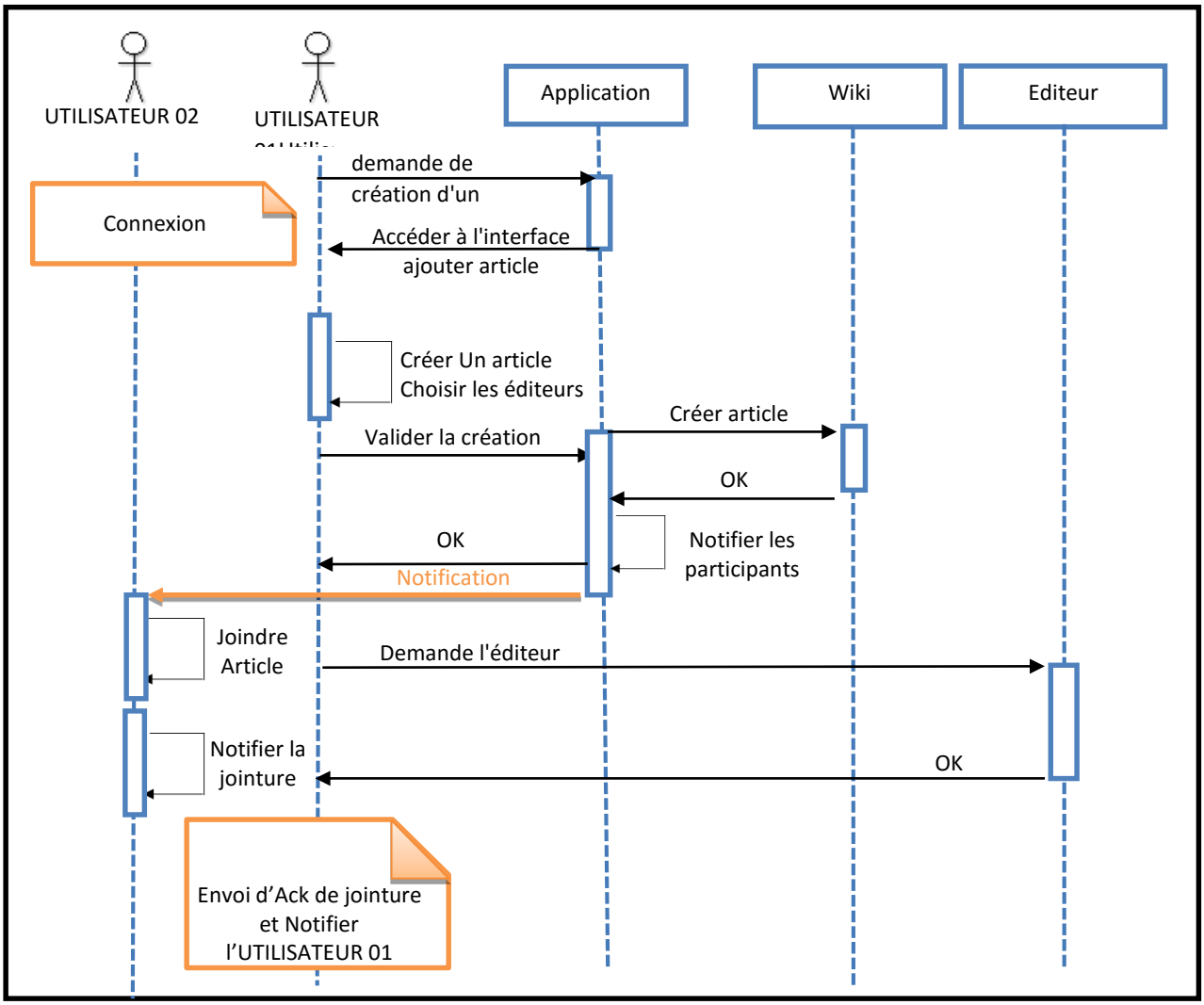



Figure 23 – Diagramme de séquence relatif à « Créer article »

### c) Digramme de séquence : « ajouter un paragraphe »

Le diagramme de séquence « ajouter un paragraphe » présente le séquencement des interactions entre l'utilisateur, les collaborateurs et l'interface d'édition où un utilisateur demande l'ajout d'un paragraphe, l'application crée un paragraphe demande un titre et les autres collaborateurs du paragraphe ajouté.

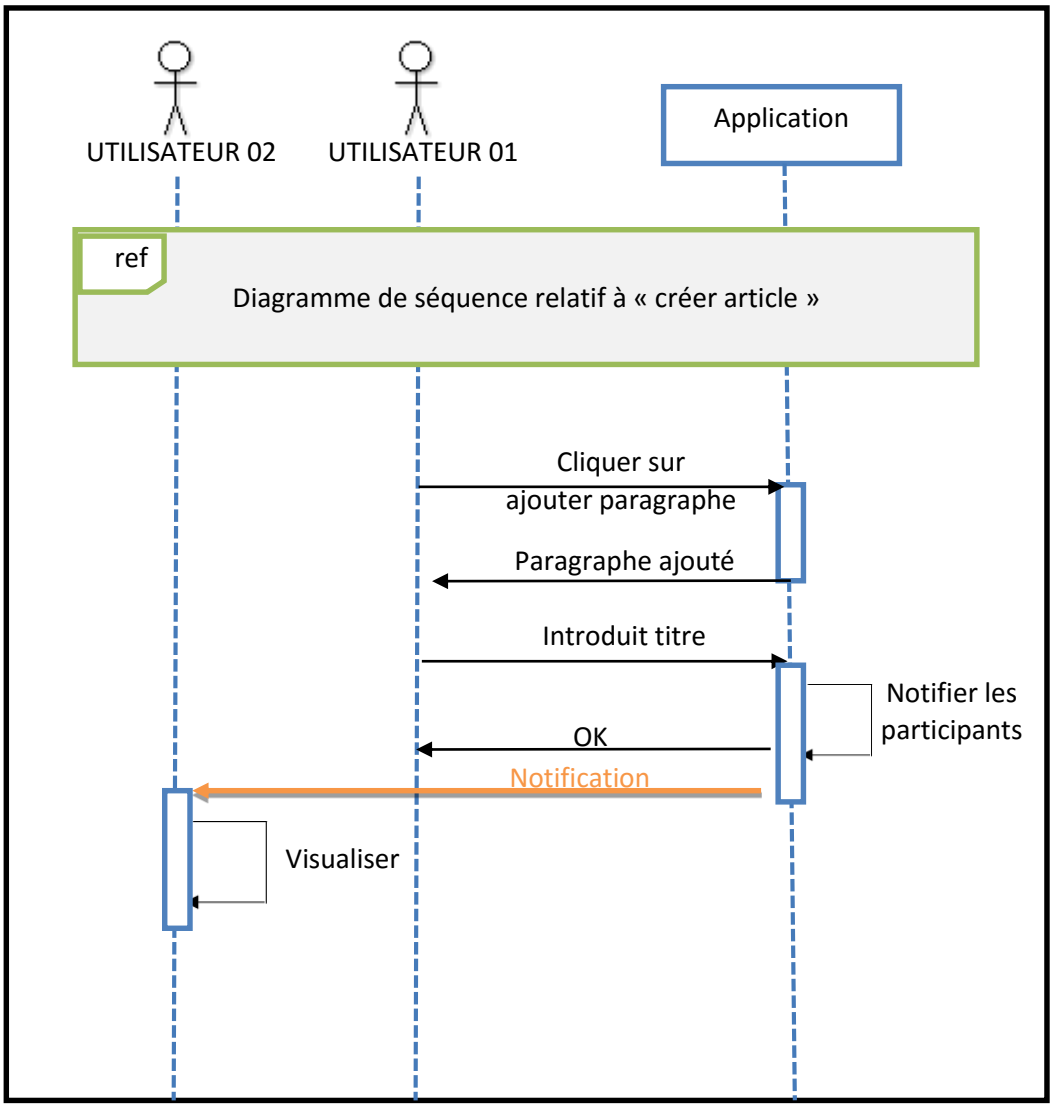



Figure 24 – Diagramme de séquence relatif à « ajouter un paragraphe »

### 3.3.3.2 Diagramme d'activités :

Dans ce qui suit, nous présentons notre diagramme d'activité pour créer et gérer le profil de l'utilisateur. La figure suivante illustre le déroulement séquentiel des traitements accomplis par un utilisateur.

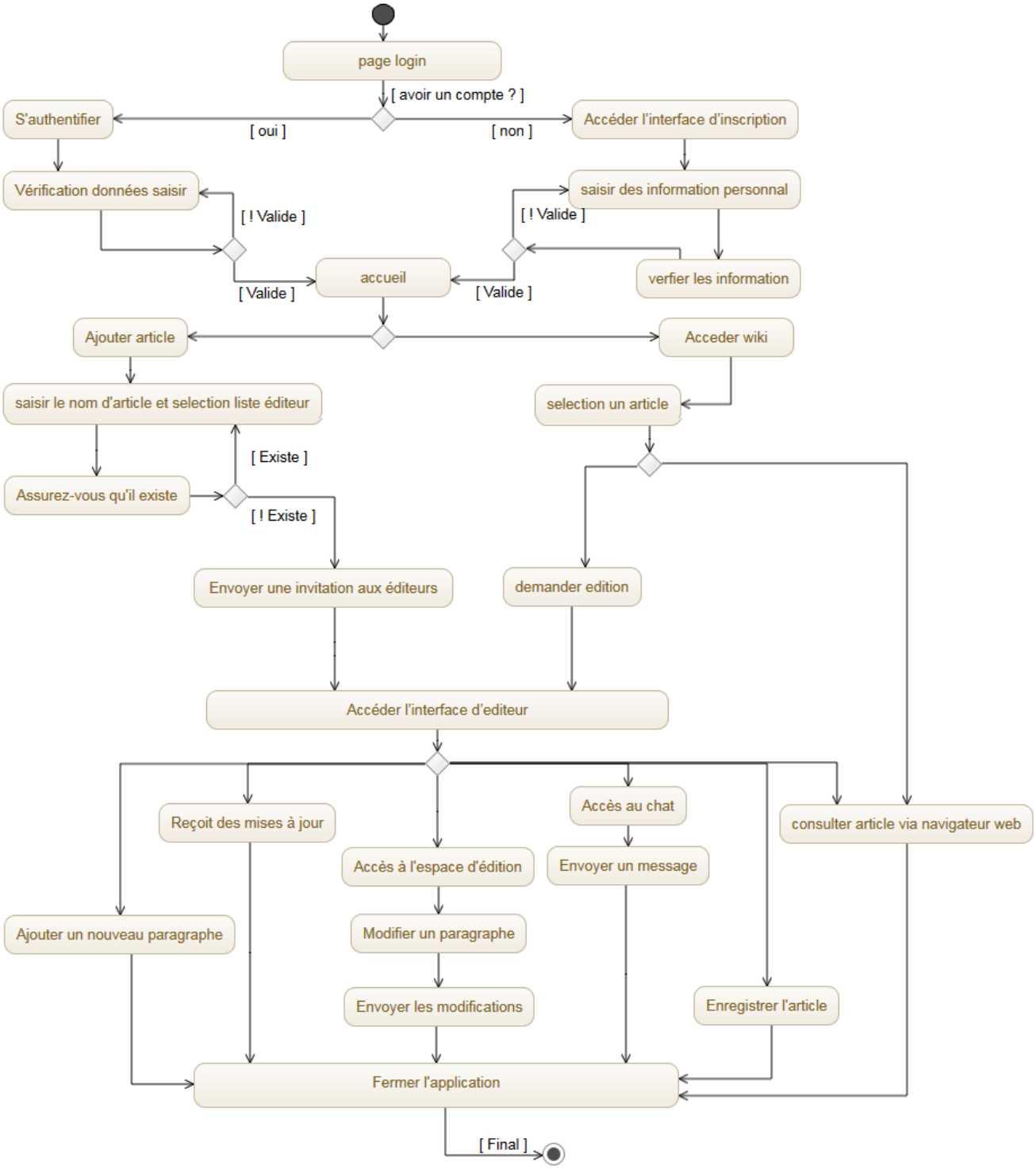



Figure 25 – Diagramme d'activité « Gérer profil l'utilisateur »

## 3.4 Conclusion

Cette phase de conception avait pour but d'exposer d'une manière globale et détaillée, le modèle du système et le fonctionnement de l'application afin de faciliter l'implémentation.
Nous avons présenté lors la première partie de ce chapitre l'algorithme d'édition collaborative ADOPTED, les problèmes qu'il pose et la solution proposée pour les résoudre. Ensuite nous avons introduit notre modèle à travers des diagrammes d'UML.

Le chapitre suivant fera le thème de la phase de réalisation, cette dernière se concentre sur l'élaboration de l'application ainsi que l'observation des résultats obtenus par l'algorithme utilisé.

# Chapitre 4

# Réalisation

## 4.1 Introduction

Après avoir détaillé la conception adaptée à notre application, nous allons consacrer le dernier chapitre de ce rapport à la partie réalisation. Dans ce chapitre, nous allons présenter les outils utilisés pour implémenter notre application collaborative baptisée ***WikiLight***, cette application est basée sur une version modifiée de l'algorithme ADOPTED. Dans cette nouvelle version, nous avons utilisé la structure de donnée et la nouvelle fonction de transformation proposé par IMINE (Univ Nancy & LORIA) et MECHAOUI (Univ Mostaganem) afin d'éviter le problème de divergence de l'algorithme ADOPTED causé par la concurrence partielle.

## 4.2 Environnement de travail

### 4.2.1 Environnement matériel

Pour la réalisation de notre projet, nous avons utilisé un ordinateur DELL (Voir annexe) caractérisé par :

• Système d'exploitation : Windows 10 Enterprise.

• Processeur : Intel(R) Core(TM) i5-52000U CPU @ 2.20GHz.

• Mémoire vive : 8 Go.

• Disque Dur : 1 To.

### 4.2.2 Environnement logiciel

**JAVA** : Java est à la fois un langage de programmation et un environnement d'exécution. Il a la particularité principale d'être portable sur plusieurs systèmes d'exploitation tels que Microsoft Windows, Mac OS ou Linux c'est la plateforme qui garantit la portabilité des applications.

Ce langage reprend en grande partie la syntaxe du langage C++, très utilisé par les informaticiens. Néanmoins, Java a épuré des concepts les plus subtils du C++ et à la fois les plus déroutants, tels que l'héritage multiple remplacé par l'implémentation des interfaces. Java permet de développer des applications autonomes mais aussi, et surtout, des applications client/serveur.

Il est doté en standard d'une riche bibliothèque de classes, comprenant la gestion des interfaces graphiques (fenêtres, boites de dialogue, contrôles, menus) [12]


**NetBeans** IDE vous permet de développer rapidement et facilement des applications bureautiques, mobiles et Web Java, ainsi que des applications HTML5 avec HTML, JavaScript et CSS. L'EDI fournit également un grand ensemble d'outils pour les développeurs PHP et C / C ++. Il est gratuit et open source et a une grande communauté d'utilisateurs et de développeurs dans le monde entier. » [13]


**JavaFX** est un ensemble de progiciels graphiques et multimédias qui permet aux développeurs de concevoir, créer, tester, déboguer et déployer des applications client enrichi qui fonctionnent de manière cohérente sur diverses plates-formes.[14]

**Jsoup** est une bibliothèque Java pour travailler avec du HTML réel. Il fournit une API très pratique pour extraire et manipuler des données, en utilisant les meilleures méthodes DOM, CSS et jquery.

Jsoup implémente la spécification WHATWG HTML5 et analyse le code HTML du même DOM que les navigateurs modernes. Jsoup est conçu pour traiter toutes les variétés de HTML trouvées dans la nature ; de primitif et validant, à tag-soupe invalide ; jsoup créera un arbre d'analyse sensible.[15]

**JGroups** est une boîte à outils pour une messagerie fiable. Il peut être utilisé pour créer des clusters dont les nœuds peuvent envoyer des messages les uns aux autres. Les principales caractéristiques comprennent

- Création et suppression de cluster. Les nœuds de cluster peuvent être répartis sur des réseaux LAN ou WAN
- Rejoindre et quitter des clusters,
- Détection d'appartenance et notification sur les nœuds de cluster joints / à gauche / écrasés.
- Détection et suppression des nœuds écrasés.
- Envoi et réception de messages de nœud à cluster (point à multipoint).
- Envoi et réception de messages de nœud à nœud (point à point). [16]


**GenMyModel** est un outil UML hébergé gratuitement par navigateur pour les développeurs et les architectes logiciels. Sa force principale : créer des modèles conformes UML en ligne et générer du code. GenMyModel a été publié en version bêta en 2013. Il supporte maintenant les diagrammes de cas de classe et d'utilisation et fonctionne avec GitHub pour héberger le code généré. [17]

### 4.2.3 Gestion du projet

Pour la gestion de notre projet, nous avons utilisé GantProject qui est un gestionnaire de tâches. La gestion des différents projets est présentée sous forme de diagramme de GANTT avec une catégorisation par ordre d'importance. La figure 26 illustre une capture d'écran du diagramme de GANTT en utilisant GantProject.

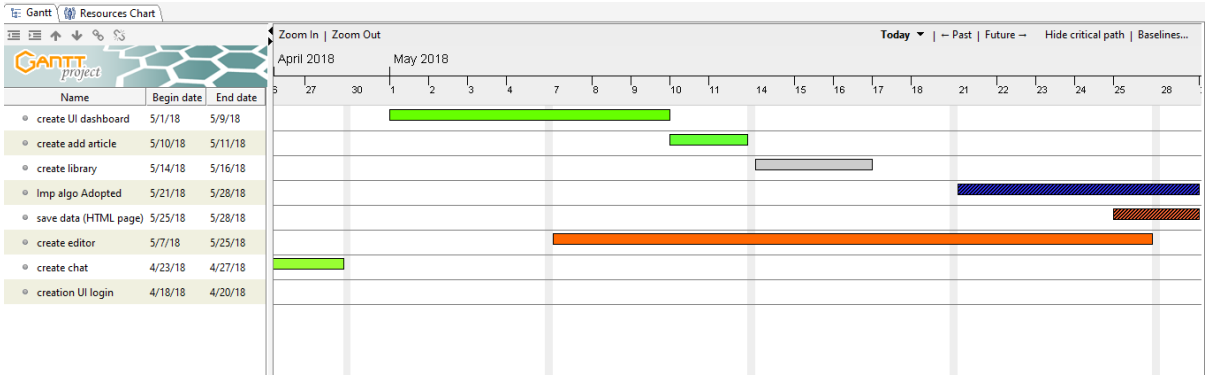



Figure 26 – Diagramme de Gantt réel

### 4.2.4 Présentation de l'application ' WikiLight '

#### 4.2.4.1 Principe de fonctionnement

Notre application permet à un ensemble d'utilisateurs de se communiquer entre eux à travers un réseau pair-à-pair et créer des groupes pour éditer des documents en collaboration et en temps réel où chaque utilisateur du groupe peut créer ou éditer un article qui existe déjà. Chaque utilisateur modifie la réplique de l'article qui lui a été associée. Une fois les utilisateurs mettent à jours leurs répliques, toutes les répliques se convergent vers le même contenu.

Pour mieux comprendre le fonctionnement de l'application, nous allons décrire quelques captures d'écran de notre application en représentant toutes les interactions entre l'utilisateur de l'application et ces interfaces. Dans cette partie, nous allons présenter quelques cas d'utilisations, sous forme d'un guide utilisateur.

#### 4.2.4.2 Interface d'authentification

Pour accéder à l'application l'utilisateur doit s'authentifier. La figure 27 donne l'interface à travers laquelle l'utilisateur s'identifie.

Lors de premier accès, l'utilisateur entre son nom, son mot de passe, et la validation du mot de passe est clique sur créer. Le système vérifie l'identifiant et le mot de passe, si aucune erreur n'est détectée, le système génère un UID pour l'utilisateur avant être dirige vers l'interface d'accueil.

Si l'utilisateur a déjà un compte au paravent, il est incité à saisir son nom et son mot de passe et valide sa connexion. Le système génère un UID, charge les articles déjà créés et affiche l'interface d'accueil. En cas d'erreurs de connexion, le système affiche des messages d'erreur et invite l'utilisateur à ressaisir ses données.


Figure 27 – Interface d'authentification

### 4.2.4.3 Interface d'accueil

Après l'authentification de l'utilisateur une interface s'affiche, c'est l'interface principale de notre application. Elle est composée d'une barre de menu à gauche et une barre de menu en haut contenant le nom de l'utilisateur.

A partir de cette interface, l'utilisateur peut gérer les articles à éditer en collaboration, il peut aussi créer des nouveaux articles, consulter la liste de ses contacts et se communiquer avec eux.

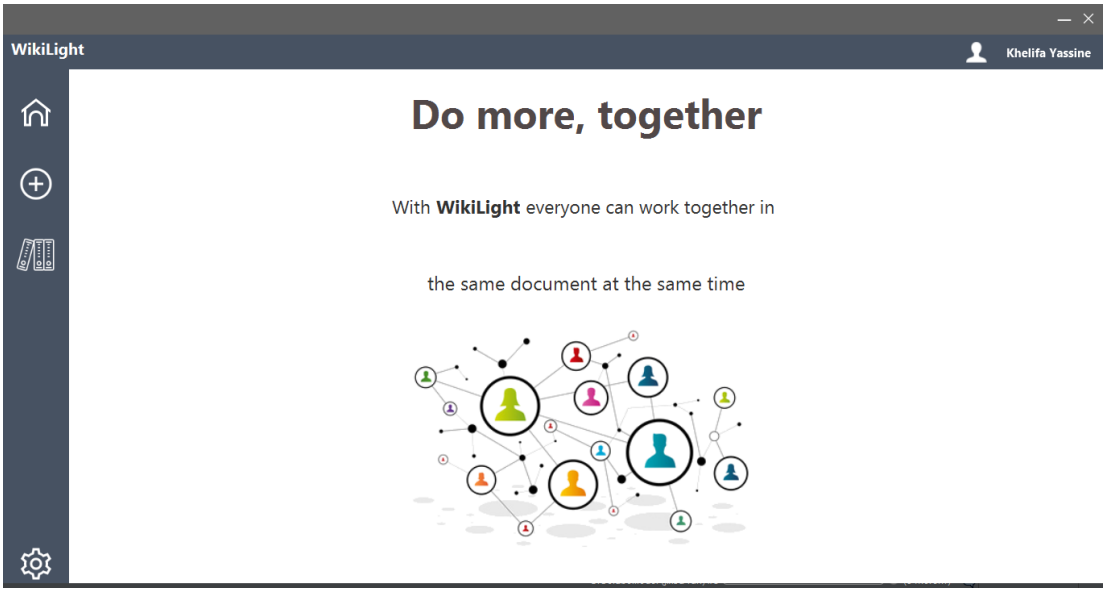



Figure 28 – Interface d'accueil

#### 4.2.4.4 Interface d'ajout d'un article

Un clic sur le bouton d'ajout d'article dans l'interface d'accueil et une fenêtre s'affiche en invitant l'utilisateur à remplir le titre de l'article et les utilisateurs éditeurs.

- Si le nom n'est pas vide et la liste des éditeurs participants contient au moins un participant l'ajout peut se faire avec succès et les participants sont invités à joindre l'édition de l'article.
- Sinon un message d'erreur s'affiche.

#### 4.2.4.5 Interface d'invitation pour l'édition collaborative d'un article

A partir de cette interface, l'utilisateur peut joindre l'édition de l'article avec d'autres utilisateurs éditeurs via un clic sur le bouton d'entrée situé au centre de la fenêtre.

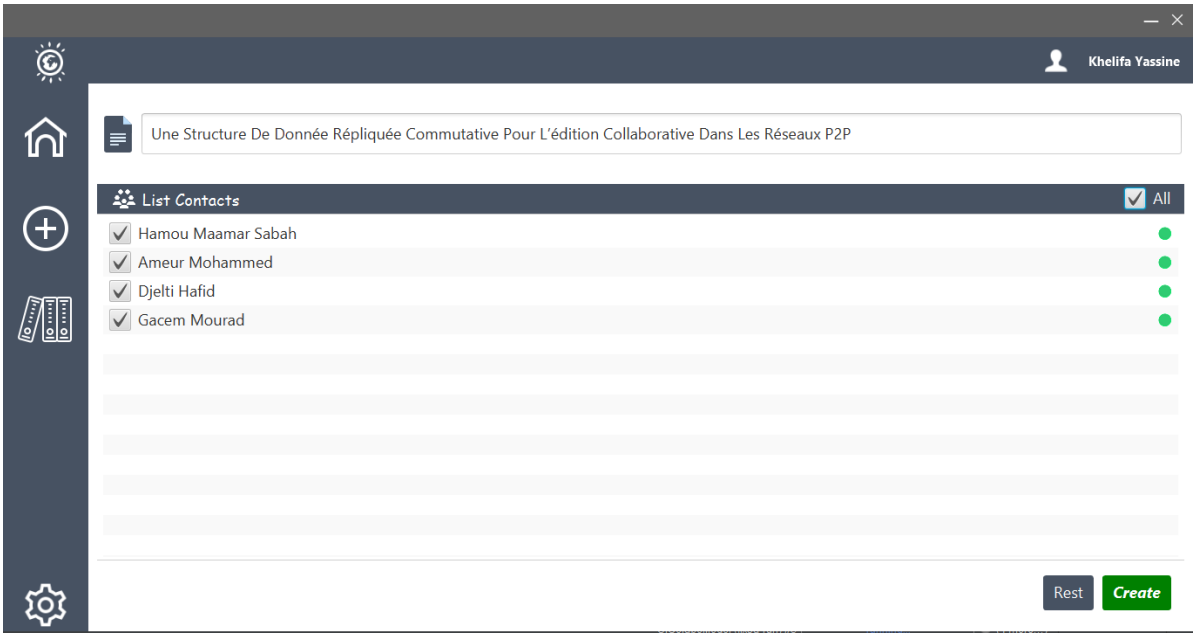



Figure 29 – Interface d'ajout d'un article

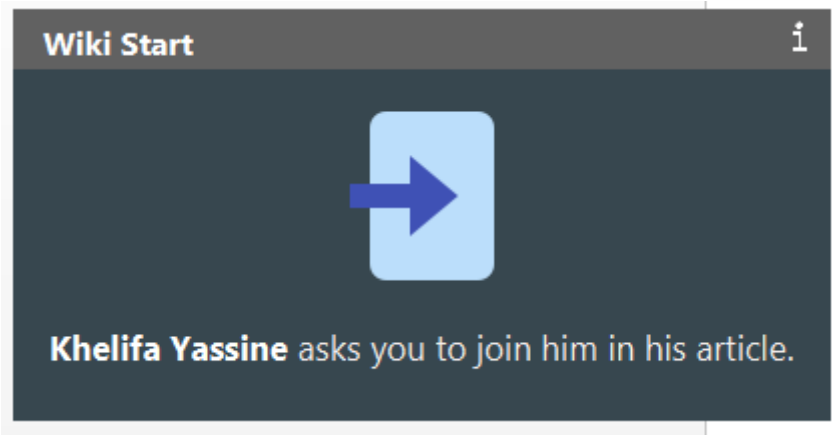



Figure 30 – Interface d'invitation pour joindre l'édition d'un article

#### 4.2.4.6 Interface de l'éditeur d'article

Cette interface contient la liste des utilisateurs éditant le même article et ceux éditant le même paragraphe (à gauche de la figure en dessous), elle contient aussi l'éditeur de texte, le sommaire, la liste des paragraphes d'article et un chat de groupe.

L'utilisateur peut à tous moments rafraîchir l'état d'édition et sauvegarder le ficher résultant.

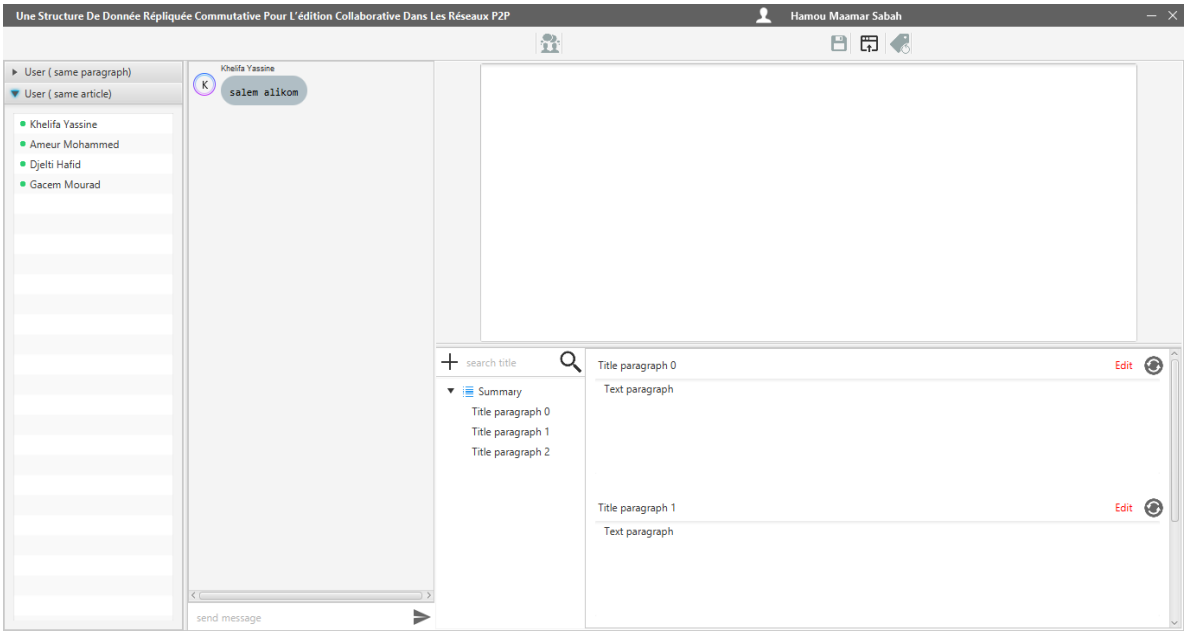



Figure 31 – Interface de l'éditeur d'article

### 4.2.4.7 Interface de chat

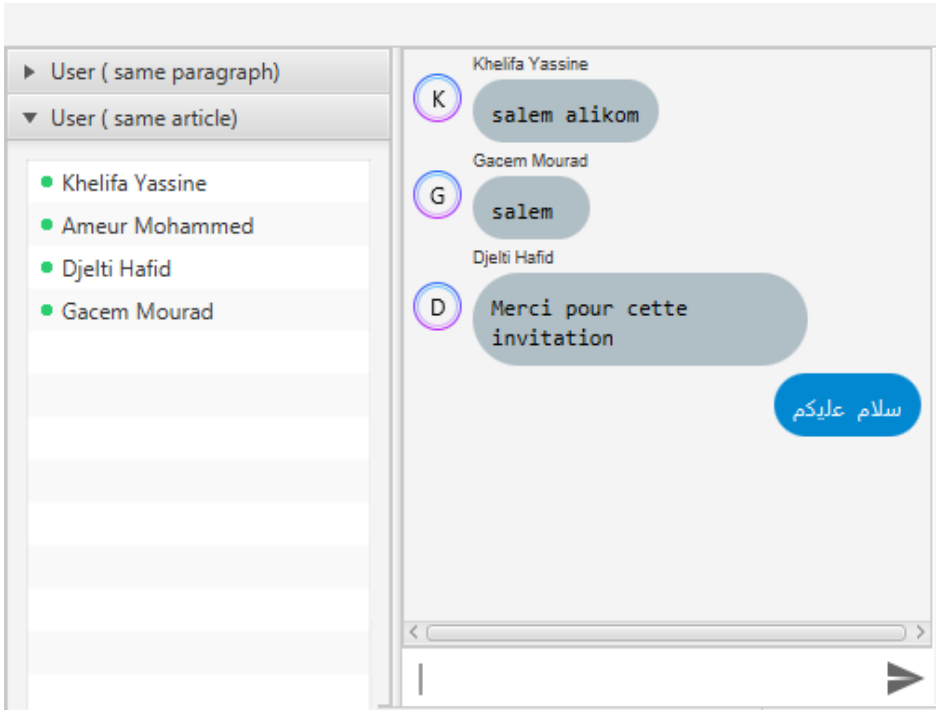



Figure 32 – Chat

Lorsque que l'utilisateur veut contacter les personnes éditant l'article en collaboration, il suffit d'utiliser le panel du chat comme le montre la figure 32 ci-dessous.

#### 4.2.4.8 Interface de la zone d'édition & l'affichage d'article

A partir de l'interface indiquée dans la figure 33, l'utilisateur peut modifier les paragraphes de l'articles à travers un clic sur le bouton éditer associé à chaque paragraphe. Il peut rafraichir et mettre à jour un paragraphe à travers le bouton rafraichir associé à chaque paragraphe.

Le menu en haut permet à l'utilisateur de mettre à jour, de consulter l'article via le web, ou de sauvegarder sa copie.

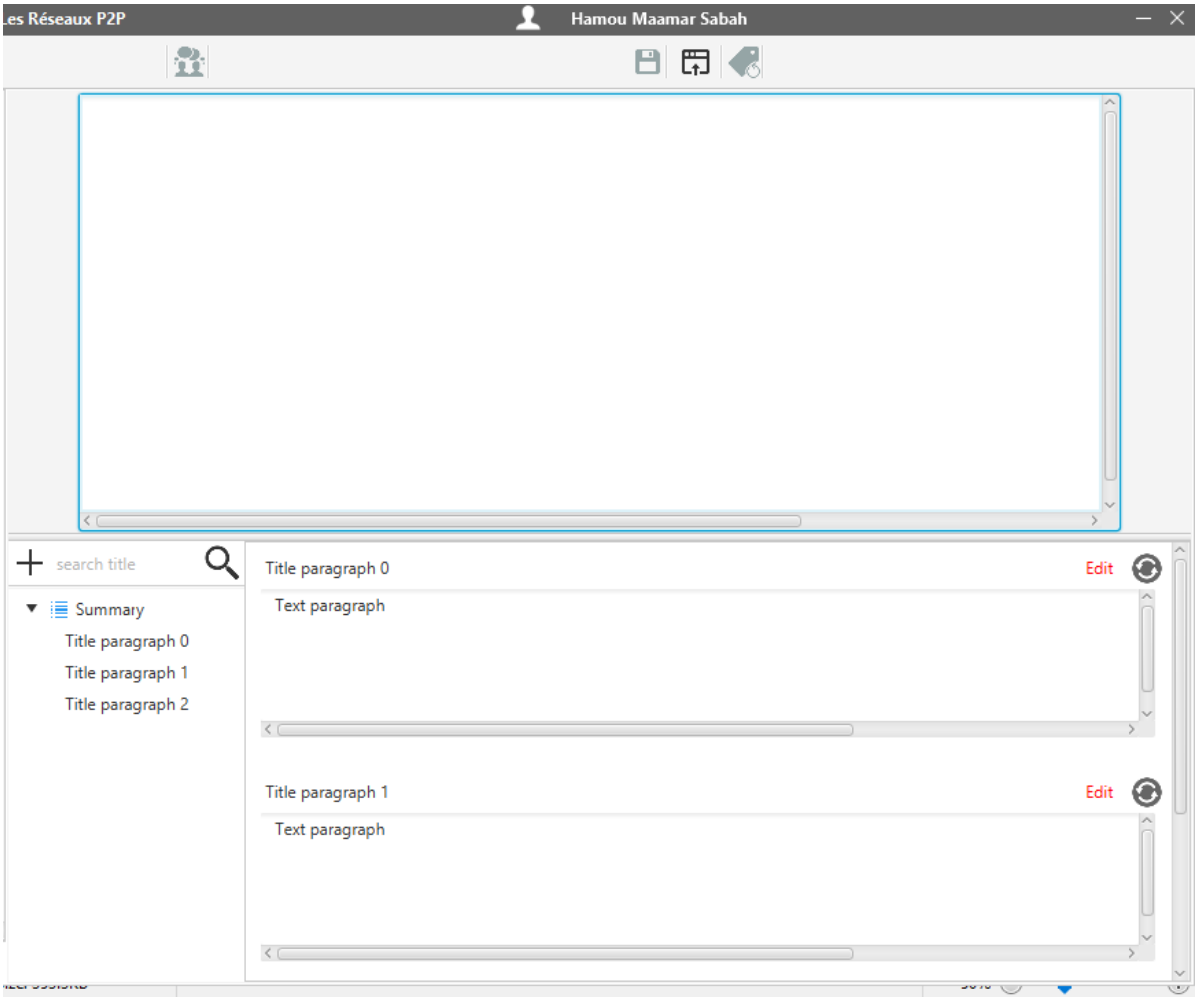



Figure 33 – Zone d'édition & affichage d'article

#### 4.2.4.9 Scenario d'édition collaborative

Dans ce qui suit, nous allons faire un scénario d'une édition collaborative faite par des étudiants rédigent un mémoire en collaboration.

Un utilisateur "Sabah" se connecte, crée un article intitulé "Les réseaux P2P", crée sept paragraphes dont les titres sont successivement "Introduction", "Chapitre 1", "Chapitre 2", "Chapitre 3", "Chapitre 4", "Conclusion", "Bibliographie" comme l'indique la figure 34 et commence à rédiger le paragraphe "Introduction".

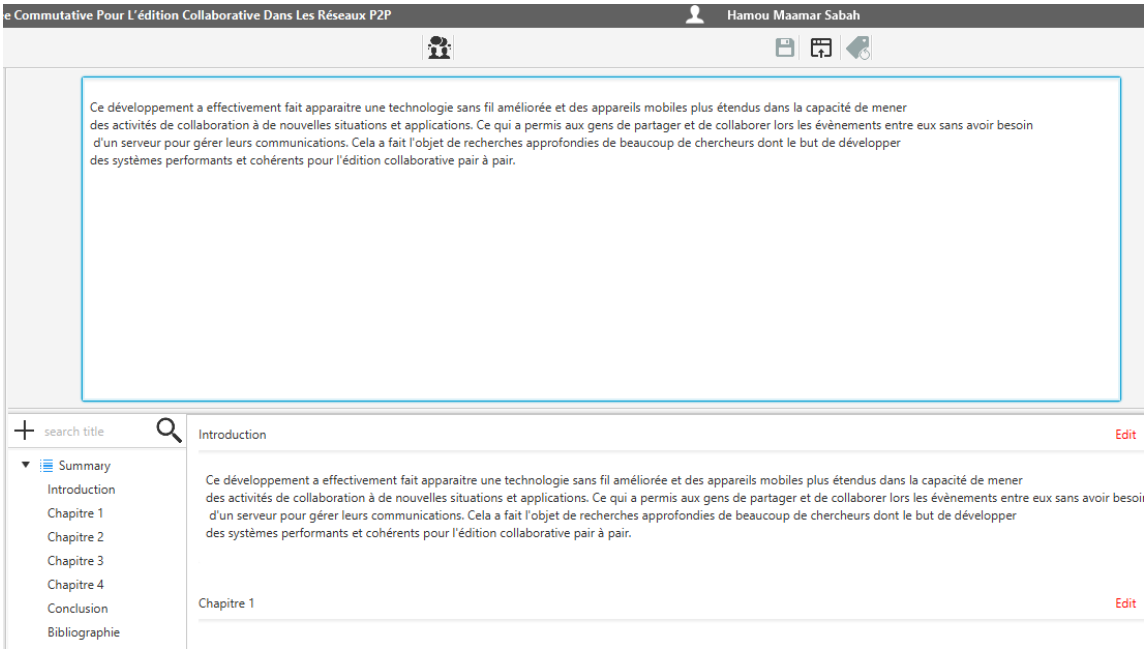



Figure 34 – Création de nouveau article par l'utilisateur "Sabah"

Un deuxième utilisateur "Yassine" rentre à l'édition collaborative et commence à rédiger le paragraphe intitulé "Introduction" simultanément avec l'utilisateur "Sabah" comme montré dans la figure 34. Il détecte que l'utilisateur "Sabah" est en train de modifier le même paragraphe comme l'indique le menu à gauche du figure 35.

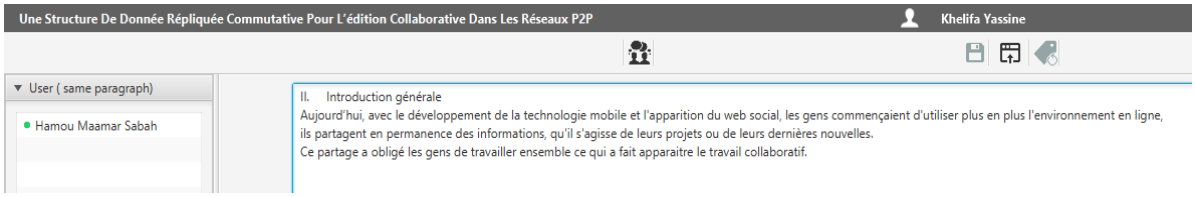



Figure 35 – Edition du paragraphe "Introduction" par l'utilisateur "Yassine"

Un troisième utilisateur "Hafid" rentre à l'édition collaborative et commence à rédiger le paragraphe intitulé "Chapitre 1" comme montré dans la figure 36.

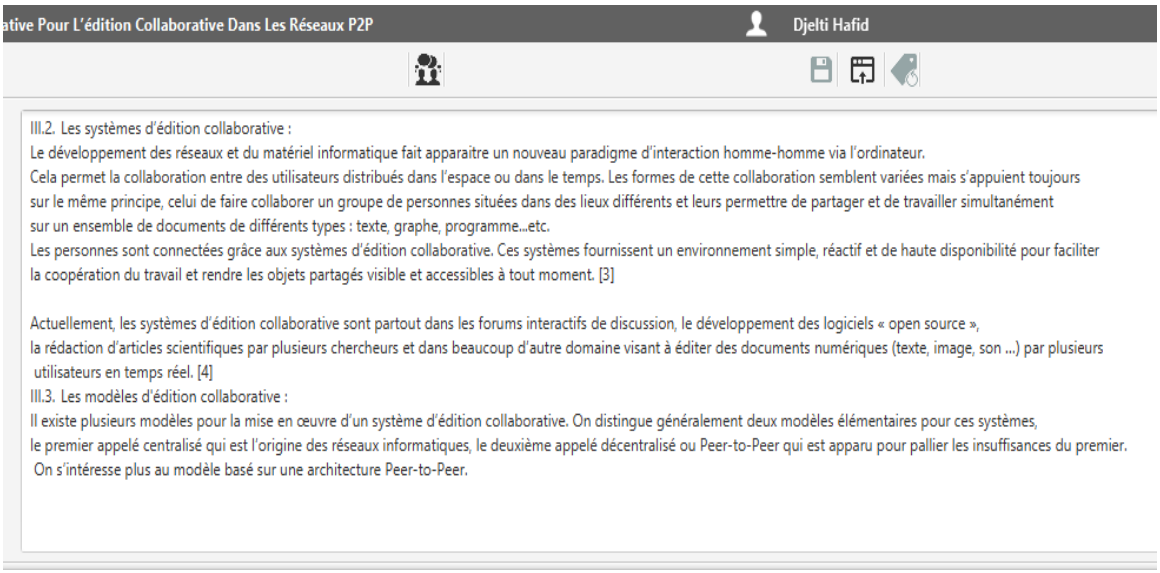
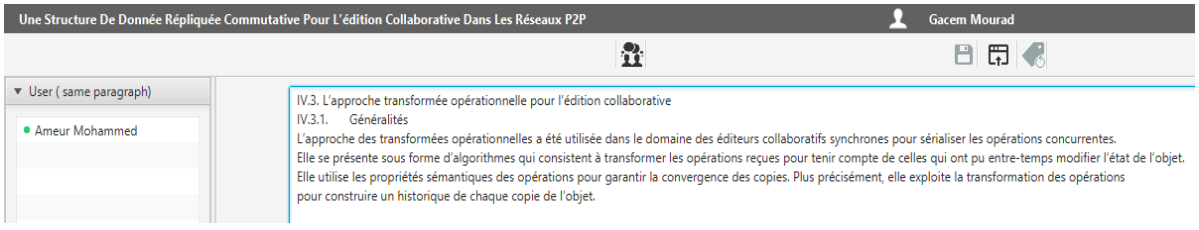



Figure 36 – Edition du paragraphe "Chapitre 1" par l'utilisateur "Hafid"

Deux autres utilisateurs "Mohamed" et "Mourad" rentrent à l'édition collaborative et commencent à rédiger un même paragraphe intitulé "Chapitre 2" comme montré dans la figure 37.



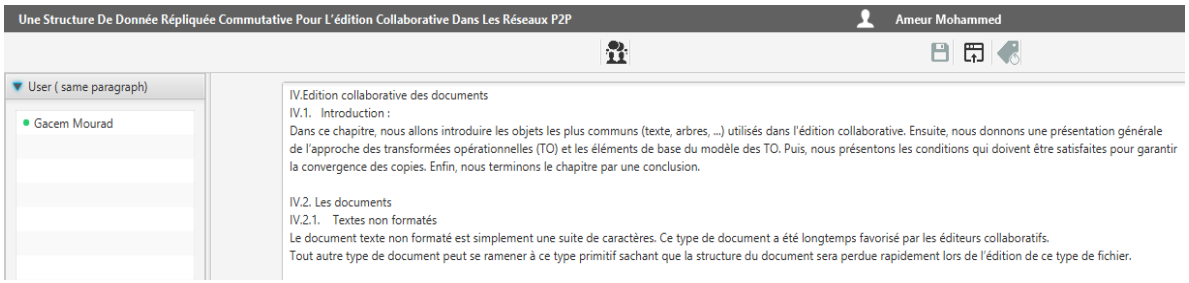
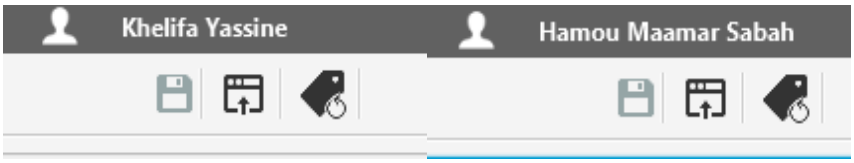



Figure 37 – Edition du paragraphe "Chapitre 2" par "Mohamed" et "Mourad"

Les utilisateurs "Yassine " et "Sabah " mettent à jour l'état du paragraphe "Introduction" qu'ils modifiaient en collaboration à travers un clic sur le bouton à droite du menu. Comme résultat, ils obtiennent la même version finale du paragraphe "Introduction".



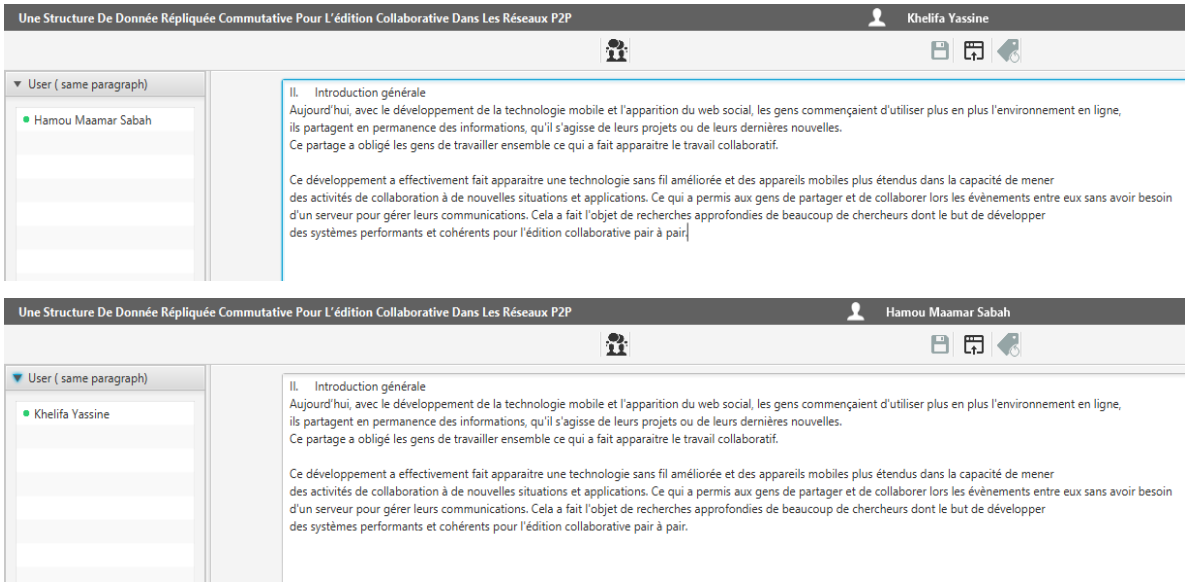




Figure 38 – Mettre à jour le paragraphe "Introduction" par "Yassine" et "Sabah"

Les autres utilisateurs "Mohamed", "Hafid" et "Mourad" mettent à jour leurs répliques ce qui fait apparaitre le bouton vert de mise à jour des paragraphes pour les différents utilisateurs en

collaboration. L'utilisateur "Sabah" met à jour ses paragraphes et obtient les contenus comme l'indique la figure 39.

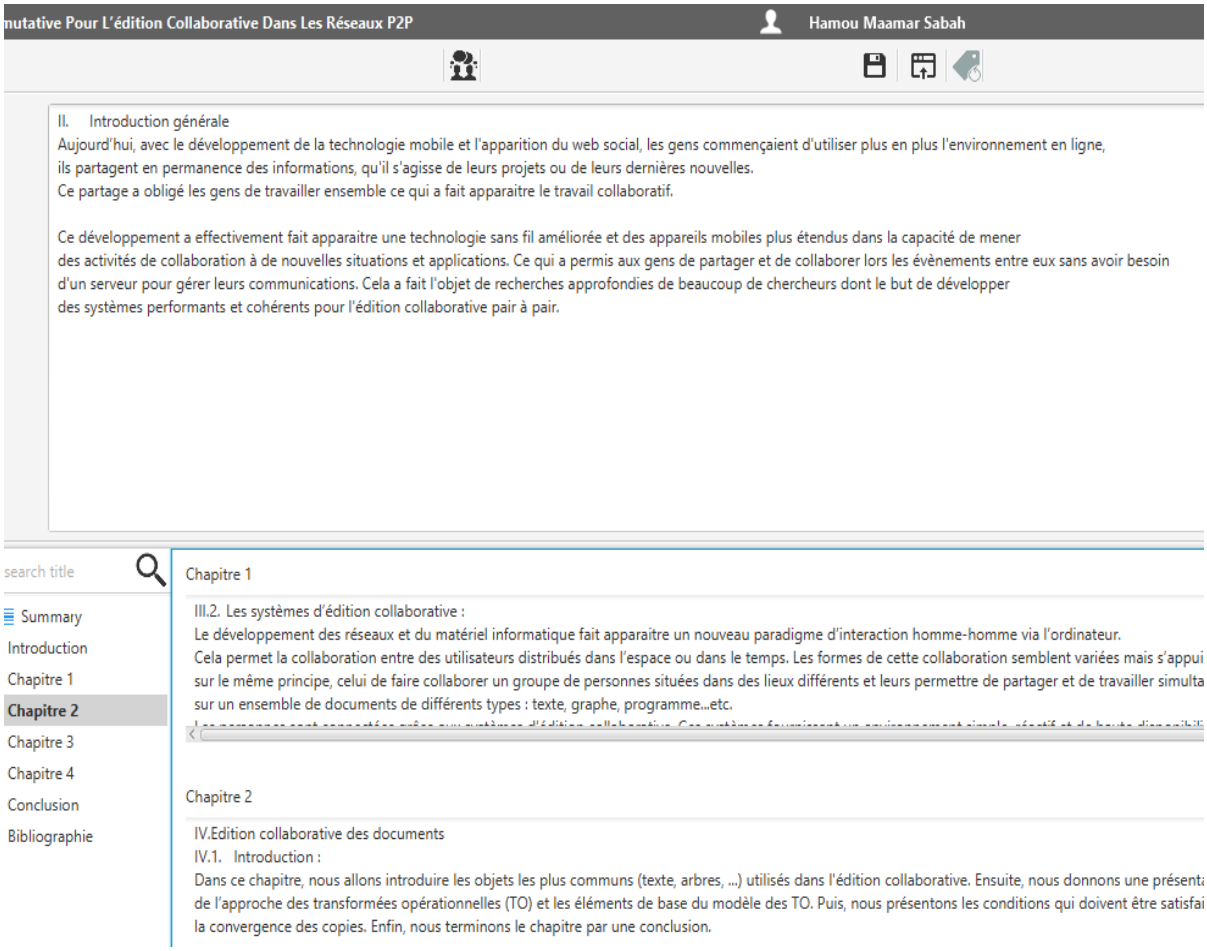



Figure 39 – Mettre à jour les paragraphes "Chapitre 1" et "Chapitre 2" par "Sabah"

### 4.2.4.10 Interface de la liste des personnes éditant le même paragraphe

Lorsque l'utilisateur veut consulter les utilisateurs qui éditent le même paragraphe, il suffit de cliquer sur l'item « User - same paragraph ». L'autre item de la liste « User - same article » fait apparaitre l'ensemble des personnes qui travaillent en collaboration sur le même article.

Chaque personne dans la liste est présentée par un élément contenant son nom et l'état de connectivité comme indiqué sur la figure 40 ci-dessous.

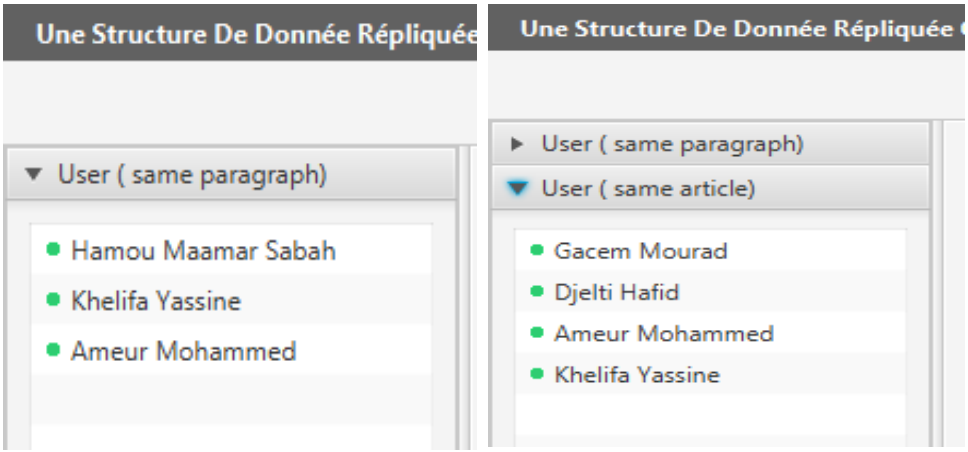



Figure 40 – Editeurs de même paragraphe et même article

### 4.2.4.11 Interface de La liste des contacts

Lorsque l'utilisateur veut consulter la liste de ses articles, il suffit de cliquer sur l'élément « Articles » pour que l'ensemble des articles s'affiche contenant la date, l'utilisateur créateur, une description, l'ID et les personnes éditeurs en collaboration.

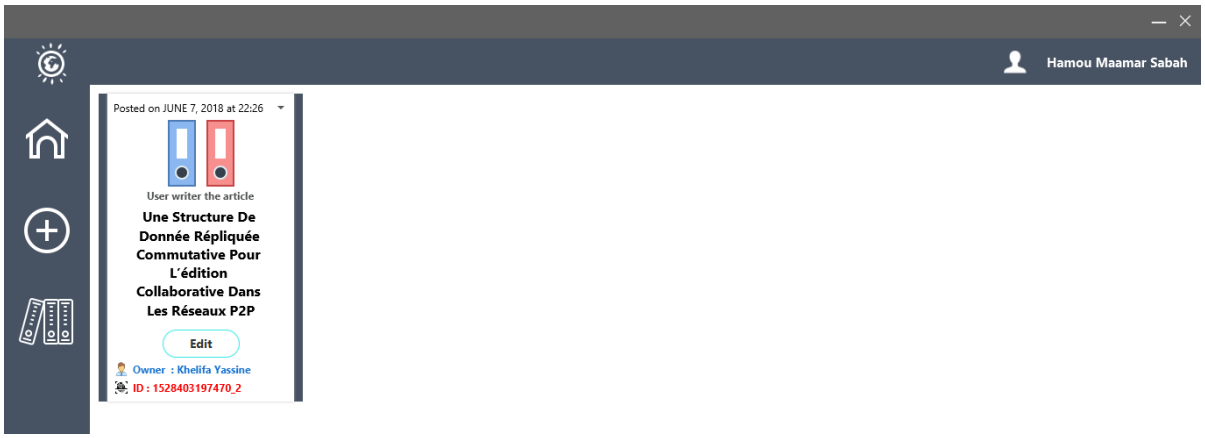



Figure 41 – liste des articles

### 4.2.4.12 Menu raccourci d'article :

Chacun des articles est muni d'un menu de raccourci qui sert à consulter l'article, le publier en web, l'exporter en PDF ou le supprimer comme l'indique la figure 40.


Figure 42 – Menu raccourcis d'un article

#### 4.2.4.13 Interface de la consultation des articles via le navigateur

Les articles sont sauvegardés d'une manière qui permet de les consulter via le navigateur. La figure 43 indique les détails de l'article tels que le titre, les paragraphes, le créateur et les gens éditeurs.

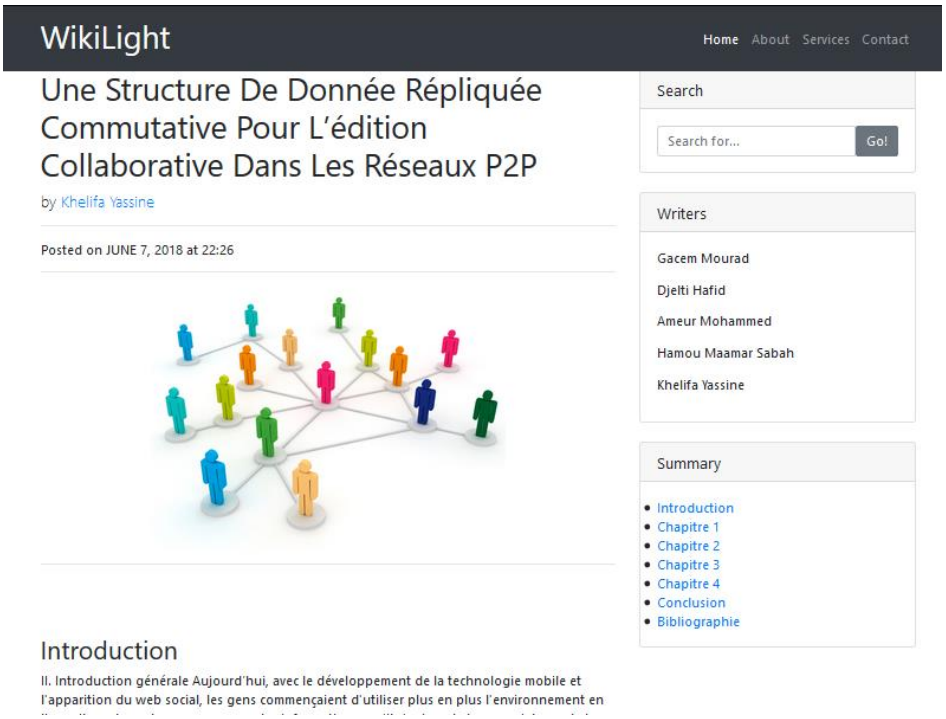



Figure 43 – Accès aux articles via le web

#### 4.2.4.14 Publication d'un article sur un serveur distant

A travers le bouton "Publish", l'utilisateur peut publier cet article ce qui fait le sauvegarder dans un serveur distant à partir duquel l'article devient publique pour que les gens peuvent le consulter. L'interface web est montrée par la figure 44.


Figure 44 – Publication d'article

#### 4.2.4.15 Interface de l'exportation en PDF

Un utilisateur peut exporter l'article en PDF à travers le menu raccourcis présenté précédemment. La figure 45 affiche un article exporté en format PDF.

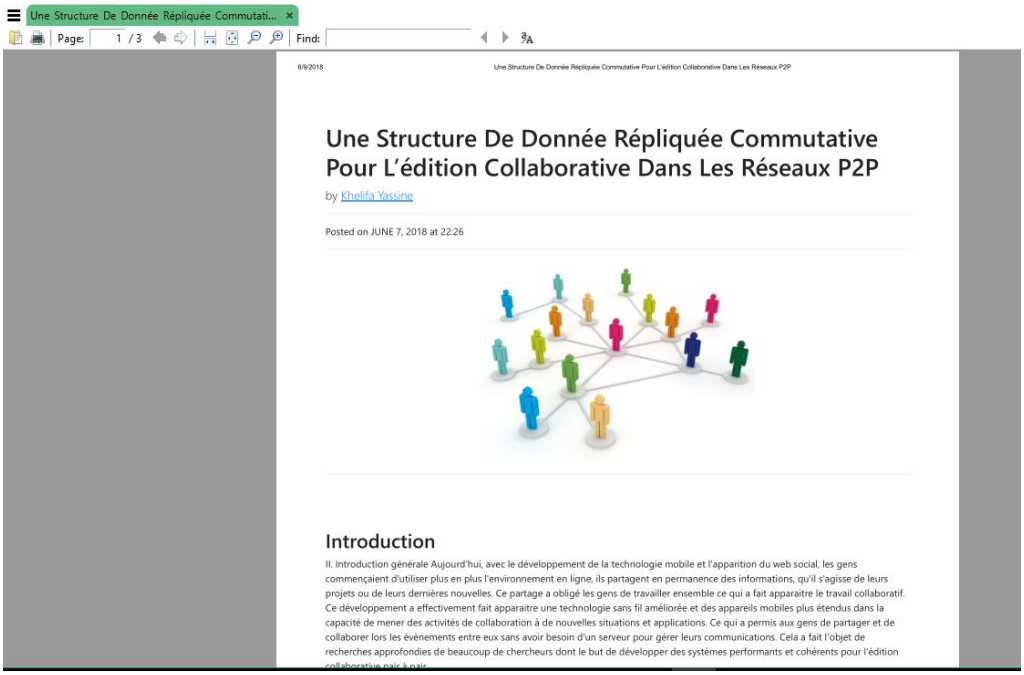



Figure 45 – Exportation d'un article en PDF

### 4.2.5 Conclusion

# Conclusion Générale

Le flot continu et croissant des informations sur les réseaux empêche les systèmes d'édition collaborative de s'adapter aux caractéristiques du nouveau phénomène portant le nom du Pair-à-Pair.

A travers ce mémoire, nous avons introduit la notion de l'édition collaborative, ses modèles, ses propriétés ainsi que les mécanismes utilisés pour répondre aux exigences dans les réseaux P2P en particulier l'approche dite des « Transformées Opérationnelles », utilisé pour sérialiser par transformation des opérations concurrentes en satisfaisant les conditions pour garantir la convergence des répliques dans les sites du système.

L'objectif principal de ce travail est d'utiliser l'approche des transformées opérationnelles TO afin de résoudre les conflits d'édition lors d'une édition concurrente d'un document partagé dans un environnement paire-à-paire tout en supportant les caractéristiques de cet environnement tels que le nombre massif d'utilisateurs ainsi que leur mouvement dynamique (joindre / quitter) dans ces environnements.

Pour cela, nous avons proposé une structure de données qui peut être utilisée par des algorithmes d'intégration basés sur l'approche des transformées opérationnelles afin de résoudre le problème de la concurrence partielle. Pour mettre en évidence cette nouvelle structure de données, nous avons proposé une version revisitée de l'algorithme ADOPTED en remplaçant l'ancienne structure de données (texte) et ses fonctions de transformations par des nouvelles fonctions qui sont appropriés à la structure de données proposée.

Finalement, nous avons implémenté une application d'édition collaborative dédiée aux environnements paire-à-paire appelé ***WikiLight***. Cette application est basée sur la version revisitée de l'algorithme ADOPTED, et permet à plusieurs utilisateurs de modifier simultanément des pages web ou des parties de ces pages sans aucunes contraintes (nombres

d’utilisateurs, ordre de réception des mises à jour). Les conflits sont réglés de façon automatique par cette nouvelle version de ADOPTED avec une convergence des copies.

Comme perspectives, Il serait aussi intéressant d’utiliser cette structure de données sur d’autres algorithmes d’édition collaborative distribués basés sur l’approche des transformées opérationnelles comme SOCT2, par exemple, afin d’éviter le problème de passage à l’échelle causé par l’utilisation des vecteurs d’états.

# Bibliographie

**Article d'un ouvrage collectif**


[1] Abdessamad IMINE, Conception Formelle d'Algorithmes de Réplication Optimiste Vers l'Edition Collaborative dans les Réseaux Pair-à-Pair, Thèse doctorat, université de Lorraine, France, 2006.

[2] Marc Shapiro and Nuno Pregui¸ Designing a commutative replicated data type. Rapport de recherche INRIA RR-6320, INRIA, Octobre 2007.

[3] Randolph Aurel Josias Oboubé. Convergence et sécurité d'accès dans les systèmes d'édition collaborative massivement répartis. Thèse de doctorat école polytech. Montréal Canada, 2014.

[4] Hanaa Mazyad. Une Approche Multi-agents à Architecture P2P pour l'Apprentissage Collaboratif. Thèse de doctorat en informatique. Université du littoral Côté D'Opale France, 2013.

[5] Rüdiger Schollmeier, A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications, Proceedings of the First International Conference on Peer-to-Peer Computing, IEEE, 2002.

[6] Gharzouli Mohamed, Composition des Web Services Sémantiques dans les systèmes Peer-to-Peer, Thèse de doctorat, Université de Constantine, Algérie 2011.

[7] G. Doyen. Supervision des réseaux et services pair-à-pair. Thèse de doctorat, Université Henri Poincaré - Nancy I, France, 2005.

[8] Mounir Tlili. Infrastructure P2P pour la Réplication et la Réconciliation des Données. Base de données [cs.DB]. Université de Nantes, France, 2011.

[9] Pascal Molli, Gerald Oster, Hala Skaf-Molli, Abdessamad Imine. Safe Generic Data Synchronizer. Rapport de recherche. LORIA, France, 2003.

[10] Stéphane Martin. Edition collaborative des documents semi-structurés. Algorithme et structure de données. Thèse de doctorat, Université de Provence - Aix-Marseille I, France, 2011.

**Rapport Technique**

[11] HACHELAF, Une Transformée Opérationnelle pour l'édition collaborative dans les réseaux P2P, mémoire de master université de Mostaganem, 2016

**Livre**

[12] D.Holmes, J. Gosling , K. Arnold, "Le langage de programmation Java,4éme édition O'Reilly", 2005

**Documents web**

[13] Netbeans.org: https://netbeans.org/features/index.html

[14] Oracle.com: https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfxoverview.htm#JFXST784

[15] jsoup.org : https://jsoup.org/

[16] jgroups.org : http://www.jgroups.org/

[17] modeling-languages.com: https://modeling-languages.com/genmymodel-towards-collaborative-uml-modeling-in-the-cloud/