MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE
LA RECHERCHE SCIENTIFIQUE
UNIVERSITE ABDELHAMID IBN BADIS - MOSTAGANEM

Faculté des Sciences Exactes et de l'Informatique
Département de Mathématiques et d'Informatique
Filière : Informatique

MEMOIRE DE FIN D'ETUDES
Pour l'Obtention du Diplôme de Master en Informatique
Option : **Ingénierie des Systèmes d'Information**

THEME :

Application d'édition collaborative mobile dans le cloud

Etudiant(e) : Djelil Fatiha Manel

Encadrant(e) : Mr Mechaoui Moulay Dris

Année Universitaire 2015/2016

Résumé :

Les appareils mobiles ne cessent d'évoluer et font partie intégrante de notre vie quotidienne, d'un autre coté les systèmes d'édition collaborative connaissent un large succès, les applications d'édition collaborative mobile ont vu le jour grâce à l'ascension de ces deux domaines.

Une application d'édition collaborative mobile a pour but de permettre à plusieurs utilisateurs munis d'un mobile d'éditer d'une façon collaborative un document partagé, chacun d'eux a une réplique du document sur son appareil sur laquelle il pourra apporter des modifications, cette application peut être utilisée en cas de catastrophe naturelle par une équipe de secouristes afin d'évaluer l'ampleur des dégâts, dans ce cas le document partagé sera une carte et les modifications apportées par les secouristes seront sous forme d'une description de l'état des lieux à une position donnée. Pour assurer la cohérence de la carte, on utilisera l'approche de la transformation opérationnelle (TO) qui est basée sur la réplication optimiste et qui supporte l'édition collaborative synchrone et on utilisera le cloud pour pallier les contraintes des mobiles en termes d'énergie, de mémoire et de capacité de calcul.

Sommaire

I.Introduction générale :

Les dispositifs mobiles font partie intégrante de nos vies, effectivement de plus en plus de gens en font un usage quotidien, ils leurs offrent un accès rapide et habile à différentes données et applications à partir de n'importe quel endroit, du fait de cette utilisation massive des mobiles dans divers domaines, la nécessité de développer des applications d'édition collaborative est née.

Aujourd'hui, il existe pas mal d'applications mobiles qui permettent d'éditer des documents de façon collaborative telles que : Google Docs pour Android, NetSketch pour iPhone ou encore Sky Drive de Microsoft, pour arriver à cette fin, le document à éditer doit être répliqué sur l'ensemble des dispositifs mobiles faisant partie de cette collaboration, on peut utiliser pour cela l'approche des transformées opérationnelles (TO) qui se base sur la réplication optimiste des documents et qui est bien adaptée à l'édition collaborative mobile, car elle permet à plusieurs utilisateurs de modifier simultanément un document partagé et permet également de travailler en mode déconnecté, son principe est d'exécuter immédiatement une opération locale sur la réplique ou elle a été générée puis de la diffuser et l'exécuter sur les autres répliques, en sachant que chaque opération est sauvegardée dans un log.

Néanmoins, avec l'approche (TO) lorsque la collaboration dure longtemps et qu'il y a beaucoup de modifications opérées sur le document partagé, les opérations peuvent vite s'accumuler dans le log destinées à les enregistrer ce qui va augmenter le temps d'intégration des opérations épuisant ainsi les batteries des mobiles, leurs puissances de calcul et occupant beaucoup d'espace mémoire, tout cela mène à la dégradation de la performance de l'application.

I.1 Problématique :

La problématique de notre travail est donc : Comment notre application d'édition collaborative peut-elle assurer la convergence des données partagées lorsque plusieurs utilisateurs font des mises à jour simultanément? Et comment remédier à l'insuffisance des ressources dans les mobiles afin d'assurer une édition collaborative efficace et à longue durée ?

I.2 Objectif :

Notre objectif est donc de de proposer une application d'édition collaborative avec un mécanisme qui supporte la collaboration synchrone tout en prenant en compte le manque de ressources qui réside dans les mobiles.

Notre mémoire est réparti sur 3 chapitres, dans le premier nous allons parler des éditeurs collaboratifs, du modèle CCI, de la réplication et ses types et de l'approche des transformées opérationnelles, nous allons aussi comparer quelques algorithmes qui se basent sur cette approche, puis nous nous intéresserons dans le deuxième chapitre au cloud qui est un paradigme en émergence et aux systèmes qui se basent sur lui puis nous dédierons le

troisième et dernier chapitre pour la conception et l'implémentation de notre application et mettrons en évidence tous les outils et les étapes par lesquelles nous sommes passés afin de la concevoir.

Liste des tableaux

Liste des figures

CHAPITRE 1

II.1. Introduction :

Les éditeurs collaboratifs sont en plein essor du fait de leurs avantages, en effet ils permettent à plusieurs utilisateurs de réaliser une tâche commune qui peut être l'édition d'un document par exemple, ce qui leur permet un gain de temps considérable.

Afin de mener l'édition collaborative à bien, il est nécessaire d'utiliser la réplication pour mettre à la disposition de chaque utilisateur une copie de l'objet partagé. En revanche, les modifications concomitantes des différentes répliques produisent une différence entre elles, c'est pourquoi il est nécessaire d'utiliser un mécanisme qui garantit la cohérence des données tel que l'approche des transformées opérationnelles dans le but de mener les répliques vers un état convergeant (c'est-à-dire le même contenu). [1]

Dans ce chapitre, nous allons définir chacune des notions d'éditeurs collaboratifs, de réplication et de transformées opérationnelles et démontrer les rôles de ces deux dernières dans l'élaboration d'un éditeur collaboratif valide.

II.2. Les éditeurs collaboratifs :

II.2.1. Définition :

Les systèmes d'édition collaborative servent à éditer un document de la part de plusieurs personnes qui peuvent être à des endroits éloignés géographiquement et ceci au même moment ou à des moments différents. [2]

Autrement dit un éditeur collaboratif est vu comme un groupe de sites qui travaillent sur un objet partagé, ce dernier est répliqué sur l'ensemble des sites participants à la collaboration, il peut être un texte, une image, une vidéo…etc. [1]

Avec l'évolution des smartphones, les éditeurs collaboratifs se sont déclinés en version mobile c'est-à-dire des applications destinées à l'édition collaborative, il existe de plus en plus d'applications de ce genre parmi lesquelles on cite : GoogleDocs.

En résumé, il y a trois caractéristiques principales qui définissent un éditeur collaboratif, en effet il doit être :

Temps-réel : C'est-à-dire que l'utilisateur doit avoir un temps de réponse minimale comme s'il été le seul utilisateur de ce système.

Distribué : Les utilisateurs peuvent se situer à des endroits différents et utiliser différentes machines.

Sans contrainte : Chaque utilisateur peut modifier sa réplique du document lorsqu'il le souhaite. [3]

II.2.2. Le modèle CCI :

Pour être valide, un éditeur collaboratif doit respecter le modèle CCI, c'est-à-dire assurer la **C**onvergence, la **C**ausalité et l'**I**ntention :

***Convergence* :** Lorsque les mêmes opérations sont exécutées au niveau de la réplique de l'objet partagé de chaque site dans des ordres différents pour chacune d'elles, on devrait obtenir des répliques identiques à la fin. [2]

Exemple :

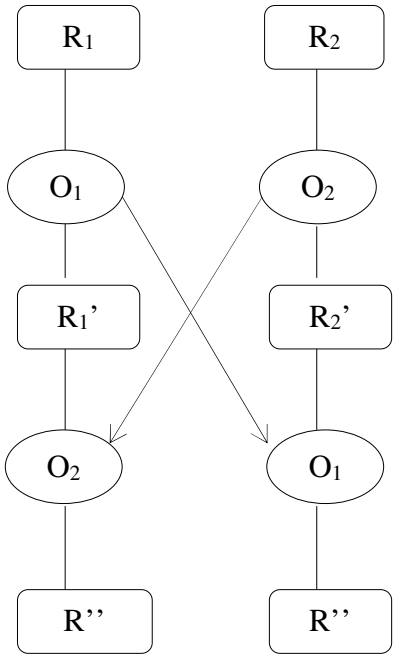

Figure.1. Scénario d'une convergence

Dans notre exemple nous avons deux répliques dont les versions initiales sont R_1 et R_2 et deux opérations O_1 et O_2, après avoir exécuté ces opérations sur les répliques dans deux ordres différents c'est-à-dire O_1 puis O_2 pour R_1 et O_2 puis O_1 pour R_2, on obtient alors une seule et même version pour les deux répliques qui est R''.

***Causalité* :** C'est une relation d'ordre entre les opérations, on dit qu'une opération O_1 précède causalement une opération O_2 si :

- L'opération O_1 a été générée sur une réplique avant que l'opération O_2 ne soit générée sur cette même réplique.

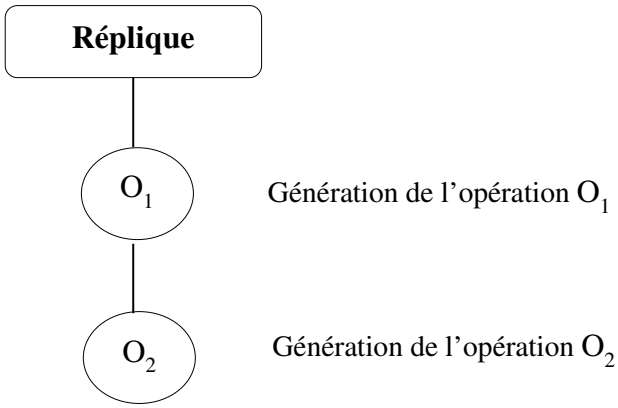

Figure.2. Première condition de la précédence causale

- Les opérations O_1 et O_2 ont été générées sur deux répliques différentes et O_1 a été exécutée sur la réplique où B a été généré avant que cette dernière ne soit générée.

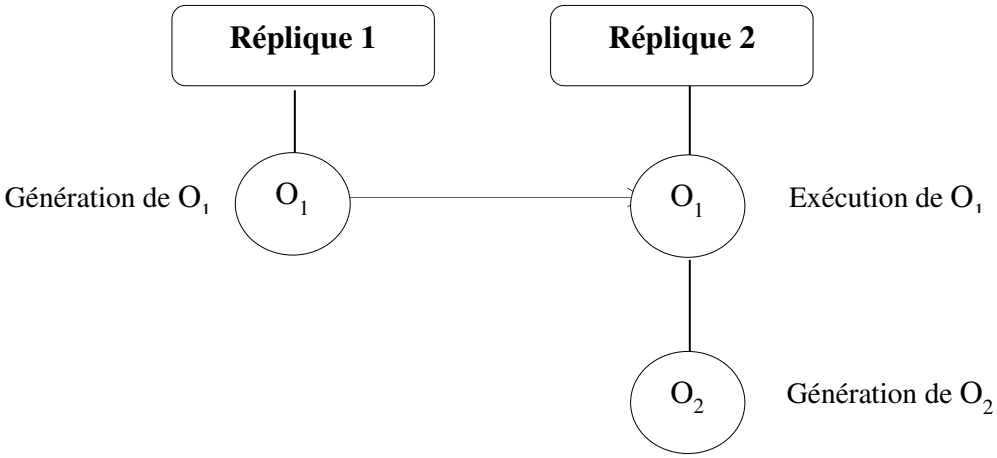

Figure.3. Deuxième condition de la précédence causale

La causalité est respectée si pour une opération O_1 qui précède causalement une opération O_2 celle-ci est exécutée après O_1 sur l'ensemble des répliques. [2]

Autrement dit les opérations qui ont une relation causale doivent être exécutées dans le même ordre sur l'ensemble des répliques. [4]

Exemple :

Dans la Figure 4, on illustre le respect de la causalité, l'opération O_2 doit s'exécuter après O_1 puisqu'elle dépend causalement d'elle, en effet l'opération O_2 a été générée à partir de la réplique 2 après l'exécution de l'opération O_1 sur cette même réplique (Condition 2 de la dépendance causale), c'est pour cette raison que l'exécution de l'opération O_2 au niveau de la réplique 3 doit être retardée en attendant la réception de l'opération O_1, une fois cette dernière exécutée, la réplique peut alors exécuter l'opération O_2.

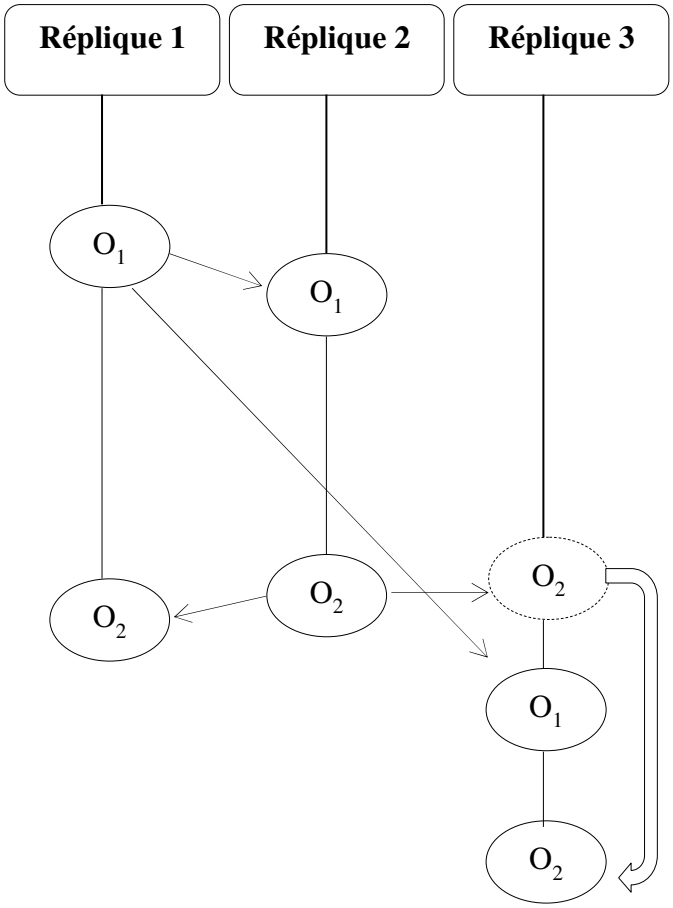

Figure.4. Scénario d'une causalité entre deux opérations

***Intention* :** C'est l'effet que produit une opération lorsqu'elle est exécutée au niveau de la réplique où elle a été générée, elle exprime alors l'effet désiré par l'utilisateur.

L'intention est respectée si toutes les opérations produisent sur toutes les répliques un effet similaire à celui qu'elles produisent sur la réplique ou elles ont été générées. [2]

En plus de tout ça, un système d'édition collaborative doit pouvoir passer à l'échelle sans que la collaboration ne soit perturbée, c'est-à-dire qu'un tel système doit être insensible au changement du nombre d'utilisateurs, et il doit également supporter la dynamicité des appareils dans les environnements mobiles ce qui veut dire leurs connexions et déconnexions fréquentes et fournir les deux modes de collaboration qui sont le mode synchrone dans le cas où les utilisateurs interagissent en même temps et le mode asynchrone où les utilisateurs interagissent à des moments différents. [5]

II.3. La réplication :

Dans les éditeurs collaboratifs chaque site doit disposer d'une copie de l'objet partagé afin que chaque utilisateur puisse apporter sa contribution dessus. Ces copies s'appellent des répliques.

II.3.1. Définition :

C'est la mise en place de copies d'un objet sur différents sites [6] afin d'assurer à différents systèmes une disponibilité des données. [7]

Les éditeurs collaboratifs utilisent la réplication afin de déposer une copie de l'objet partagé au niveau de chaque site participant à la collaboration [2] assurant ainsi une disponibilité de l'objet partagé et permettant au même temps aux utilisateurs de le manipuler de façon parallèle. [5]

Lorsqu'un utilisateur opère une modification sur une réplique ceci génère une opération qui sera immédiatement exécutée sur cette même réplique puis sera envoyée aux répliques restantes. [2]

II.3.2. Opérations sur les répliques :

Il existe deux types d'opérations qui sont appliquées sur les répliques :

Les opérations locales **:** Une opération est locale pour la réplique où elle a été générée. [2]

Les opérations distantes **:** Une opération est distante pour une réplique si elle la reçoit de la part d'une autre réplique, c'est-à-dire que cette opération a été générée sur une autre réplique. [2]

Prenons l'exemple de la figure 3, l'opération O_1 est locale pour la réplique 1 et est distante pour la réplique 2.

II.3.3. Les types de réplication :

On distingue deux types de réplication qui sont la réplication optimiste et la réplication pessimiste :

-La réplication optimiste : Ce type donne l'opportunité à chaque utilisateur de modifier sa copie lorsqu'il le souhaite et indépendamment des autres répliques. [5] Ceci va donner à un certain moment des répliques divergentes mais qui vont finir par converger, pour cela lorsqu'un utilisateur modifie sa copie ses modifications vont être diffusées aux autres copies.

Le principal avantage de la réplication optimiste c'est le temps de réponse court qu'elle offre, parmi les systèmes qui utilisent la réplication optimiste il y a : UseNet, CVS et IceCube. [1]

Exemple :

Dans cet exemple nous remarquons que l'opération O_1 précède causalement l'opération O_2 car elle a été générée et exécutée avant l'exécution de l'opération O_2 sur le site 1, dans ce cas l'opération O_1 doit être exécutée avant l'opération O_2 sur l'ensemble des sites c'est pour cela que l'opération O_2 a été annulée sur le site 2, elle doit attendre l'exécution de l'opération O_1 avant qu'elle ne puisse être ré-exécutée.

On voit aussi que chaque utilisateur modifie librement sa copie que ce soit au même temps avec l'autre utilisateur ou pas et que chaque opération générée sur un site est diffusée aux autres sites.

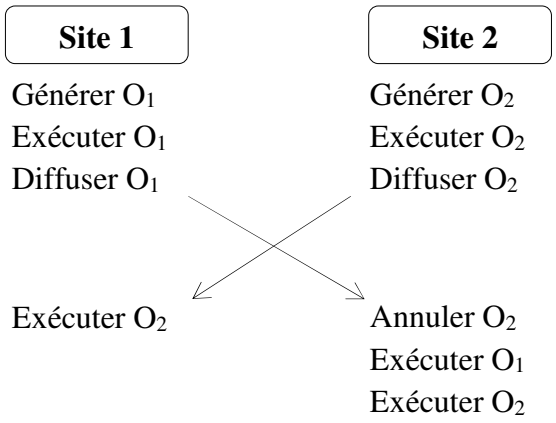

Figure.5. Réplication optimiste

-La réplication pessimiste : Elle donne à l'utilisateur l'impression qu'il n'existe qu'une seule copie de l'objet partagé et qu'il est le seul utilisateur du système, elle ne favorise pas la concurrence puisqu'elle utilise des verrous pour empêcher qu'une copie ne soit modifiée par plus d'un utilisateur à la fois. [5] autrement dit le verrou empêche l'exécution de plusieurs opérations simultanément alors lorsqu'un utilisateur est en possession du verrou les autres utilisateurs doivent attendre qu'il ait fini ses modifications afin d'obtenir le verrou chacun à leur tour puis exécuter leurs opérations et c'est ce temps d'attente qui constitue l'inconvénient majeur de ce type de réplication. [1]

Exemple :

Dans ce type de réplication, lorsqu'un utilisateur est en train de modifier sa réplique il obtient un verrou et les autres utilisateurs sont bloqués en attendant qu'il finisse ses modifications et qu'il libère le verrou.

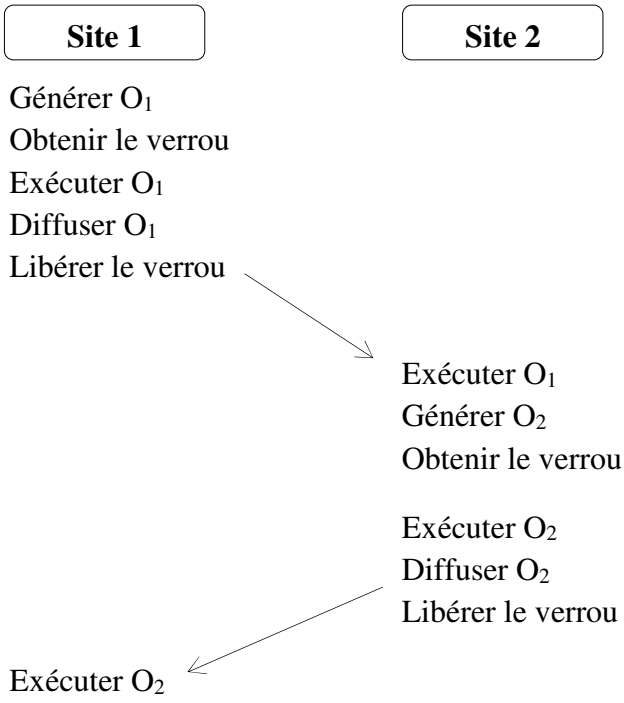

Figure.6. Réplication pessimiste

Ici l'utilisateur du site 2 ne peut générer aucune opération tant que l'utilisateur du site 1 n'a pas libéré le verrou.

Dans notre travail nous allons nous baser sur la réplication optimiste pour la liberté qu'elle donne aux utilisateurs ainsi que le gain de temps qu'elle offre. De plus, la réplication pessimiste ne correspond pas aux mobiles car leurs déconnexions fréquentes peuvent poser problème au moment de l'obtention ou de la libération du verrou. [8]

II.3.4. Avantages et inconvénients :

La réplication a comme avantages d'offrir une grande disponibilité des données et d'augmenter leur fiabilité ainsi qu'améliorer la performance en divisant la charge de travail.

Cependant, le maintien de la cohérence de toutes les répliques représente une vraie problématique qui nécessite l'intervention d'un mécanisme spécifique. [8]

II.4. L'approche des transformées opérationnelles (TO) :

Parmi les systèmes qui aident à maintenir la cohérence des données et qui établissent un ordre total sur les opérations nous citons l'approche des transformées opérationnelles, c'est une technique qui supporte la collaboration dans les environnements distribués et mobiles. [9]

II.4.1. Définition :

C'est une technique conçue pour remédier au problème de divergence des copies causées par les opérations concurrentes dans les éditeurs collaboratifs, elle a été introduite par Ellis et Gibbs dans le but de permettre d'effectuer des mises à jours simultanées sur le même document sans l'utilisation de verrou ni d'un site central [8] c'est l'une des techniques les plus adéquates pour supporter l'édition collaborative dans des environnements mobiles et distribués, Elle considère plusieurs sites possédant chacun une copie de l'objet partagé dont le placement est basée sur la réplication optimiste. [5]

Elle dote toutes les répliques d'un log (journal) dans lequel toutes les opérations exécutées seront stockées construisant ainsi ce qu'on appelle l'histoire de chaque réplique. [9]

Son but est de résoudre les conflits qui apparaissent lors des modifications concurrentes opérées sur l'objet partagé [10] .Elle se compose d'algorithmes d'intégration et de fonctions de transformation : [1]

- Algorithme d'intégration : Il est responsable de la réception, de la diffusion et de l'exécution des opérations et intègre les opérations distantes dans l'histoire (journal) de chaque site en prenant en compte les dépendances causales et les opérations concurrentes et sélectionnent celles qui doivent être transformées [2], les histoires ne sont pas forcément identiques mais doivent être équivalentes [1], il est à noter que l'histoire stocke les opérations d'un site suivant l'ordre de leur exécution. [10]

  Cet algorithme est indépendant du type de données manipulé et est correcte s'il respecte le modèle CCI. [11]

- Fonctions de transformation : Elles fusionnent (sérialisent) les opérations concurrentes [4] en tenant compte des opérations précédentes, elles sont spécifiques à un type de données particulier. [12]

II.4.2. Principe :

Le principe de l'approche (TO) est comme suit :

Chaque utilisateur possède une réplique de l'objet partagé, lorsqu'il génère une opération elle sera exécutée immédiatement sur la réplique locale puis elle sera envoyée et intégrée au niveau du log de toutes les autres répliques c'est-à-dire qu'elle sera transformée par rapport aux autres opérations puis sera exécutée. [10]

Lors de l'utilisation de l'approche des transformées opérationnelles, chaque site passe donc par les étapes suivantes :

-La génération d'une opération locale : Lorsqu'un utilisateur modifie sa réplique, une opération locale est générée, cette dernière est immédiatement exécutée puis sera stockée dans le log associée à cette réplique, il est nécessaire de stocker ces opérations afin d'intégrer les opérations distantes.

-Diffusion d'une opération : L'opération locale qui a été générée puis exécutée sur la réplique locale sera par la suite diffusée aux autres sites où elle sera considérée comme une opération distante.

-Réception d'une opération distante : Lorsqu'une réplique reçoit une opération distante elle ne l'exécute pas immédiatement mais elle l'intègre au niveau de son log afin de prendre en compte les opérations concurrentes et l'effet des opérations déjà exécutées et la transforme si nécessaire. [8]

-Exécution de l'opération distante : Une fois que l'opération distante qui se trouve dans le log est prête à être exécutée, elle sera appliquée sur la réplique.

Seule l'approche (TO) est indépendante du type des données manipulé, garantit la convergence et passe à l'échelle avec des fonctions de transformation vérifiant les conditions TP1 et TP2 (voir section II.4.5), parmi les algorithmes qui se basent sur l'approche (TO) nous citons : Ellis et Gibbs (1989), Ressel et al (1996), Sun et al (1998), Suleiman et al (1998), Imine et al (2003) et SO6. [5]

II.4.3. Catégories des fonctions de transformation :

Les fonctions de transformation doivent avant toute chose assurer la convergence des données [14], elles peuvent être divisées dans l'approche (TO) en deux catégories :

-La transformation inclusive (Transposée en avant): IT(O_1, O_2), elle transforme l'opération O_1 par rapport à l'opération O_2 dans la mesure ou l'opération O_1 prenne en compte l'effet de l'exécution de l'opération O_2, c'est-à-dire que cette dernière inclue cet effet. [15]

-La transformation exclusive (Transposée en arrière) : ET(O_1, O_2), elle transforme l'opération O_1 par rapport à l'opération O_2 dans la mesure ou l'opération O_1 ne prenne pas en compte l'effet de l'exécution de l'opération O_2, c'est-à-dire que cette dernière exclue cet effet. [15]

Dans le tableau qui suit, nous allons citer quelques systèmes basés sur l'approche des transformées opérationnelles et le type de fonctions de transformation qu'ils utilisent :

Systèmes	Algorithme d'intégration	Fonctions de transformation
GROVE	dOPT	IT
DistEdit	Selective-undo	IT et ET
JOINT EMACS	adOPTed	IT
Jupiter	Jupiter OT	IT
Google Wave	Google Wave OT	IT
REDUCE, CodoxWordCoPPT	GOT, GOTO, AnyUndo	IT et ET
CodoxWord, CoPPT,CoMaya, CoVim,CoFlash, CoCKEditor	COT	IT

Tableau.1. Les fonctions de transformation utilisées dans différents systèmes basés sur l'approche (TO)

II.4.4. Types d'algorithme d'intégration :

En sachant que chaque site est doté d'un log (journal) dans le quel ses opérations sont stockées, lorsqu'un site reçoit une opération distante O l'algorithme d'intégration fait alors les étapes suivantes :

-Il divise l'histoire du site en deux partie : la première correspond aux opérations qui précédent l'opération O selon la relation de dépendance causale et la deuxième correspond aux opérations qui sont concurrentes à O.

-Il fait appel aux fonctions de transformation afin d'obtenir l'opération O' qui est la transformation de l'opération O par rapport à la deuxième partie de l'histoire, c'est-à-dire par rapport aux opérations concurrentes.

-Il exécute l'opération O' sur le site et l'ajoute à son log.

Il existe deux types d'algorithmes d'intégration: [16]

Centralisé : Dans ce genre d'algorithme d'intégration, il y a un élément central qui assure que les opérations concurrentes soient exécutées dans le même ordre au niveau de tous les sites.

Parmi ces algorithmes, on cite :

-SOCT4 : Dans cet algorithme, les opérations sont ordonnées grâce à un séquenceur (serveur central) c'est ce qu'on appelle l'estampillage puis seront livrées par la suite aux autres sites dans cet ordre, la différence dans cet algorithme est que les transformations sont faites au niveau du site ou les opérations sont générées et non pas sur le site qui reçoit les opérations, ce dernier enregistre donc l'opération telle quelle dans son log. [17]

-Jupiter : Il est dérivé de l'algorithme dOPT, il tourne autour d'un serveur central et chaque client possède une réplique du document et il en est de même pour le serveur, néanmoins, la communication se fait qu'entre le serveur et le client.

Lorsqu'un client génère une opération, il l'envoie au serveur qui la transforme, l'applique sur sa réplique du document puis la diffuse aux autres clients. Lors de la réception d'une opération propagée à partir du serveur central, un site client peut transformer cette opération si nécessaire

-COT : L'originalité dans cet algorithme est qu'il n'utilise pas de log pour stocker les opérations mais un vecteur de contexte pour garder la trace de l'état du document suite à la création d'une opération, COT induit que les opérations soient exécutées sur le même état du document afin qu'il n'y ait pas de divergence.

Le problème avec cet algorithme est qu'il est couteux en place mémoire à cause de son vecteur de contexte. [10]

Dans ce type d'algorithmes, l'inconvénient majeur est le serveur central, ce dernier est un point sensible car si jamais il sature ou il tombe en panne tous les mécanismes vont échouer car la collaboration et les performances globales du système reposent uniquement sur lui.

Décentralisé : Dans ce genre d'algorithme d'intégration, il n'y a pas d'élément central pour ordonner les opérations, ces dernières peuvent donc être exécutées dans des ordres différents au niveau des sites.

Parmi ces algorithmes, on cite :

-dOPT : L'approche TO a été utilisée pour la première fois dans l'algorithme dOPT dans lequel deux propriétés doivent être satisfaites :

La causalité : Lors de la réception d'une opération distante elle doit être causalement prête avant d'être exécutée ce qui veut dire qu'une opération ne peut être exécutée sur un site que si toutes les opérations qui la précèdent ont été exécutées sur ce même site.

Pour satisfaire cette priorité, cet algorithme utilise un vecteur d'état SVs associé à chaque site S, la composante SVs[j] équivaut au nombre d'opérations générées par le site j, reçues par le site S et exécutées sur la réplique de l'objet partagé avec $1 \leq j \leq N$, N étant le nombre de sites, cette composante est incrémentée sur le site S à chaque fois qu'une opération qui est générée sur le site j est exécutée, il peut s'agir de l'exécution d'une opération locale si S=j ou d'une opération distante si S!=j après qu'elle ait été délivrée bien sûr.

Une opération O générée sur un site S ne peut être délivrée à un site S' que si toutes les opérations qui la précédent causalement ont été délivrées c'est-à-dire $SVs'[i] \geq SVs[i]$.

La convergence : Cette propriété est satisfaite lorsque les opérations distantes sont transformées par rapport aux opérations concurrentes stockées dans le log avant d'être exécutées grâce aux fonctions de transformation.

L'intention : Pour préserver l'intention de l'utilisateur, cet algorithme se base sur les fonctions de transformation et plus précisément les fonctions de transformation inclusives IT.

Il a cependant été prouvé quand dans certains scénarios l'algorithme dOPT ne parvient pas à maintenir la consistance du document notamment lorsque deux opérations sont générées à partir de deux états différents du document. [8]

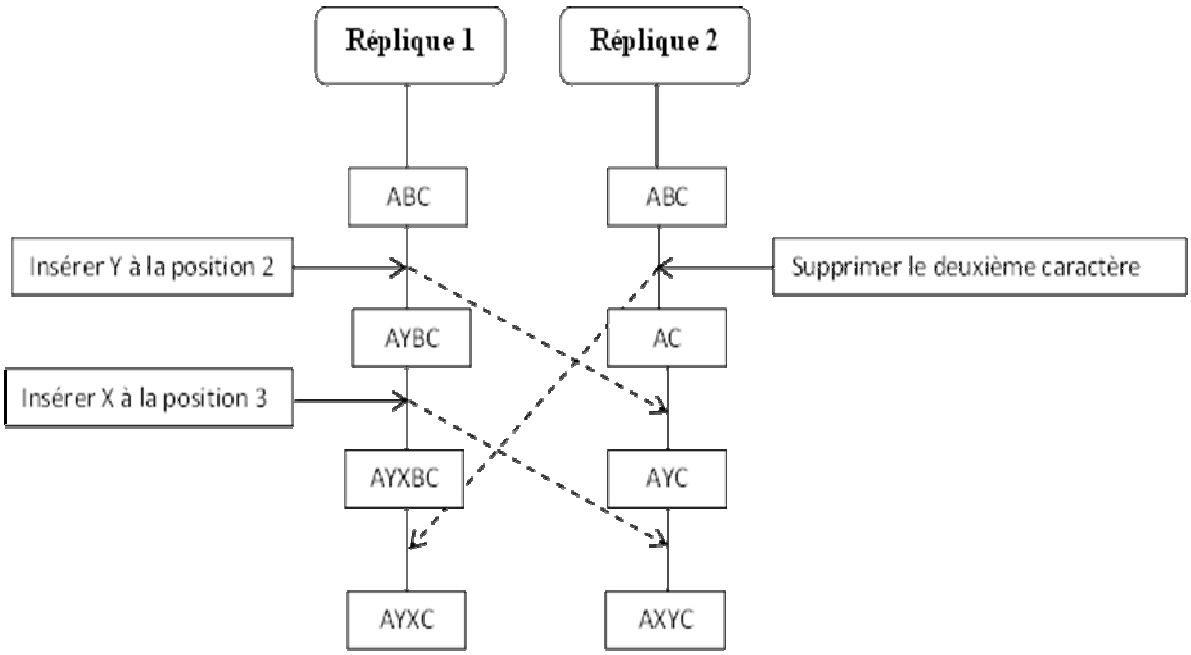

Figure.7. Le puzzle de dOPT

-Adopted : il a été proposé par Russel et All en raison des problèmes que présente dOPT, il se base sur un vecteur d'état afin d'assurer le respect de la causalité et les fonctions de transformation pour le respect de l'intention.

Cet algorithme utilise une histoire multidimensionnelle qui est représentée par un graphe d'interaction n-dimensionnel ou n représente le nombre de sites en collaboration, ce graphe est formé d'un ensemble de sommets et d'un ensemble d'arcs, les sommets représentent les différents états par lesquels passe le site et les arcs les opérations qui ont permis le passage d'un état à un autre, donc chacun des sites possède un graphe qui représente l'histoire multidimensionnelle des opérations locales et distantes qui y sont exécutées. [17]

Le graphique d'interaction fournit un modèle très utile pour visualiser la relation de transformation entre les opérations originales et transformées. [18]

Les requêtes r envoyé par chaque site sont de la forme suivante (u, k, v, o) où u est l'utilisateur qui a généré cette requête, k est son numéro de série, v est le vecteur d'état et o est l'opération elle-même. [19]

Adopted permet aux opérations concurrentes d'être transformées et exécutées correctement, mais le problème majeur de cet algorithme est l'envoi du vecteur d'état à chaque message ce qui lorsque le nombre d'utilisateurs augmente pose un problème de bande passante, son deuxième inconvénient est la place mémoire qu'occupe le graphe multidimensionnelle qui fait office d'histoire pour chaque site. [17]

-SOCT2 : C'est un algorithme qui se base sur l'approche des transformées opérationnelles afin de garantir la cohérence des données, il utilise un vecteur d'état pour assurer la causalité, l'intention de l'utilisateur et la convergence quant à elles sont assurées grâce aux fonctions de transformations et à leur respect des conditions TP1 et TP2.

Lorsqu'une opération est générée, elle est exécutée localement sur le champ puis va être diffusée aux autres copies ainsi qu'à la copie locale afin qu'elle soit enregistrée et intégrée au niveau du log de chaque copie.

La satisfaction de la condition TP2 est la contrainte majeure de cet algorithme. [17]

-Got : Tout comme dans SOCT2, la causalité est respectée en utilisant un vecteur d'état, le respect de l'intention se fait grâce aux fonctions de transformation.

Dans cet algorithme, un ordre de sérialisation unique est défini sur les opérations, ce dernier est basé sur l'ordre causal.

Sa contrainte est du à cet ordre qui exige parfois que certaines opérations soient défaites à l'arrivée d'une nouvelle opération. [17]

Dans notre application nous allons travailler avec un algorithme d'intégration décentralisé afin d'éviter tout problème lié au serveur central et qui est bien adapté aux besoins de notre application.

II.4.5. L'approche (TO) et le modèle CCI :

Pour être correct, l'algorithme d'intégration doit respecter le modèle CCI :

Convergence : Pour que la convergence soit assurée la préservation de la causalité et de l'intention ne sont pas suffisantes, il faut que les fonctions de transformation respectent les deux conditions TP1 et TP2 :

Condition TP1 : L'exécution de deux opérations concurrentes sur différentes répliques et dans n'importe quel ordre donne le même état. [10]

si on applique sur une réplique A l'opération 1 puis l'opération t(o2,o1) on doit obtenir le même état que si on applique l'opération 2 suivi de l'opération t(o1,o2) c'est-à-dire :

op1 T(op2; op1) = op2 T(op1; op2) [2]

Exemple :

Dans cet exemple nous avons deux répliques qui ont le même état initial, après avoir exécuté l'opération 1 puis l'opération 2 sur une réplique, et l'opération 2 puis l'opération 1 sur l'autre réplique nous avons obtenu le même résultat qui est « xayzb », ceci illustre le respect de la condition TP1.

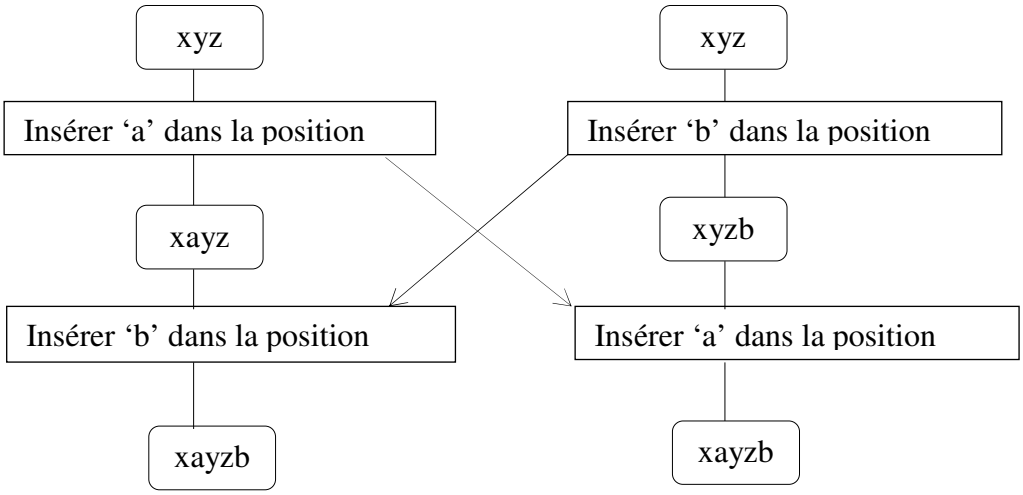

Figure.8. Condition TP1

Condition TP2 : L'ordre de transformation des opérations d'une séquence ne doit pas influer sur le résultat de la transformation d'une opération par rapport à cette séquence.

T(T(op3; op1); T(op2; op1)) = T(T(op3; op2); T(op1; op2))

Exemple :

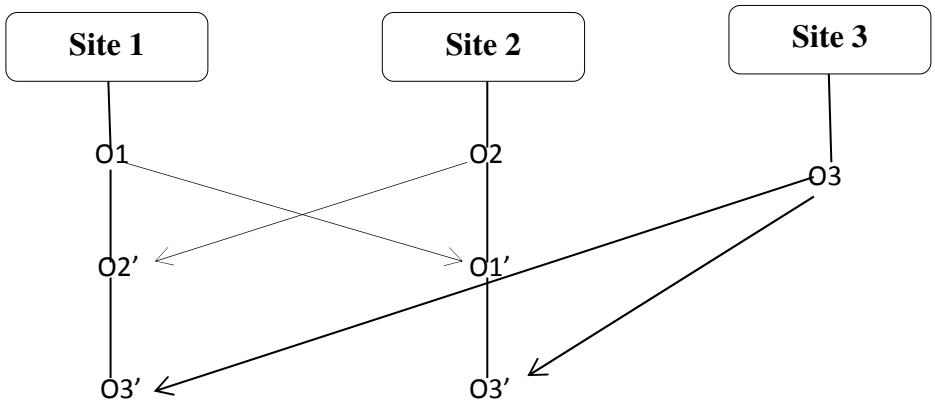

Figure.9. Condition TP2

Causalité : L'approche (TO) garantit que l'ordre d'exécution des opérations qui ont une dépendance causale reste le même sur l'ensemble des répliques en utilisant plusieurs solutions comme les vecteurs d'horloge ou les vecteurs d'état.

Intention : Elle préserve l'intention grâce aux fonctions de transformation. [11]

II.4.6. Avantages et inconvénients :

L'approche (TO) permet une collaboration sans contraintes c'est-à-dire que chaque utilisateur peut modifier sa copie à tout moment et échanger par la suite ses mises à jour avec les autres, [5] elle permet alors aux utilisateurs de travailler sur leurs répliques locales en mode déconnecté et de se synchroniser avec les autres utilisateurs une fois qu'il se reconnecte, [13] et elle établit un état de convergence sans perdre aucune mise à jour.

Cependant cette technique a l'inconvénient d'être coûteuse en termes d'énergie et de puissance de calcul pour les dispositifs mobiles : L'approche utilise un log (journal) pour stocker les opérations, la taille de ce dernier augmente tant que les utilisateurs mettent à jour leurs répliques, ceci affaiblie la performance de notre application car le temps de réponse va s'accroitre selon la taille du log à cause de la génération et de l'intégration des opérations qui vont prendre plus de temps et ceci va aussi occuper beaucoup de mémoire en sachant que les mobiles ont une mémoire limitée, donc plus la collaboration dure, plus nous consommons de l'énergie
Aussi, quand le log est de grande taille il est de plus en plus difficile pour le mobile d'exécuter et d'intégrer les opérations car ses capacités de calcul sont limitées. [9]

L'approche (TO) est alors adéquate pour la construction de systèmes d'édition collaborative corrects pour tous les avantages qu'elle offre et parcequ'elle respecte le modèle CCI.

II.5. Conclusion :

Dans ce chapitre nous avons étudié les éditeurs collaboratifs, leurs modes de fonctionnement et les différents mécanismes nécessaires pour qu'ils fonctionnement correctement, comme la réplication qui est essentielle pour garantir une disponibilité des données et une grande réactivité des utilisateurs et l'approche des transformées opérationnelles qui a aussi un rôle primordial pour maintenir la cohérences des données, nous avons aussi comparé une multitude d'algorithmes d'intégration à propos desquels on a constaté que le serveur central posait problème en ce qui concerne les algorithmes centralisés comme SOCT4 et Jupiter et que la taille des histoires posait problème du côté des algorithmes décentralisés comme Adopted et soct2.

Dans notre travail nous allons choisir les outils les plus avantageux pour construire un éditeur collaboratif efficace et qui respecte les contraintes des appareils mobiles à savoir la réplication

optimiste qui est souple et qui offre un gain de temps et un algorithme d'intégration basé sur l'approche TO qui n'exige pas de site central et qui s'adapte très bien aux réseaux mobiles.

CHAPITRE 2

III.1. Introduction :

La technologie mobile et le cloud computing sont deux domaines en plein essor, en effet, de nos jours, il existe de plus en plus d'applications destinées à l'édition collaborative en version mobile parmi elles y en a qui sont déployés dans le cloud telles que SkyDrive ou GoogleDrive.

Les appareils mobiles ont fait de larges progrès en termes de capacité de calcul et de stockage, ils sont devenus les appareils numéro un pour faire profiter aux utilisateurs d'une multitude d'applications tout en supportant leurs dynamicité ainsi que leurs collaboration en temps réel, néanmoins, les appareils mobiles souffrent d'un manque remarquable de ressources (batterie, capacité de calcul, mémoire).

Effectivement, la capacité de calcul et de stockage limitées de ces appareils représentent un obstacle pour une utilisation efficace et à longue durée d'applications d'édition collaborative mobiles, c'est pour cela que de plus en plus d'applications travaillent avec le cloud, ceci permet de décharger les opérations couteuses pour le mobile sur le cloud ce qui économisera leurs ressources.

Dans ce chapitre, nous allons présenter le cloud qui représente une réelle solution pour ce genre de contraintes.

III.2. Le cloud :

III.2.1. Définition :

Le cloud est basée sur la virtualisation des ressources qui seront stockées dans des serveurs distants accessible via internet par le biais de différents dispositifs ce qui augmente la disponibilité des données. [4]

Autrement dit c'est le stockage de données et logiciels sur des serveurs distants au lieu de serveurs locaux. Dans notre cas on va faire du mobile cloud computing qui est l'accès aux services offerts par le cloud en utilisant des appareils mobiles. [20]

III.2.2. Virtualisation :

C'est le fait de transformer des ressources matérielles pour créer une machine virtuelle en utilisant un logiciel spécialisé. Ces ressources seront placées par la suite des dans des serveurs localisés à distance

III.2.3. Modes d'utilisation :

Le cloud se décline en trois modes d'utilisation (trois couches) qui sont :

Iaas : Mettre à disposition de l'utilisateur des applications comme un service [22]

Saas : Mettre à la disposition de l'utilisateur un environnement de développement prêts à l'emploi, exemple : app engine de google, windows azur de microsoft. [21]

Paas : Mettre à la disposition de l'utilisateur une infrastructure matérielle c'est-à-dire plus concrètement des capacités de calcul et de stockage…etc [21]

Dans le cadre de notre application, nous allons utiliser le cloud en mode Infrastructure As A Service.

III.2.4. Types de cloud :

Il existe trois types de cloud qui sont :

- Le cloud privé : Ce type de cloud est réservé à un seul client, ce type de cloud peut être externe c'est-à-dire géré par un prestataire ou bien interne c'est-à-dire géré par le prestataire lui-même.
- Le cloud public : Ce type de cloud peut avoir plusieurs clients à la fois qui peuvent y héberger leurs données.
- Le cloud hybride : Ce type de cloud mixe entre l'utilisation de cloud privé et de cloud public, par exemple on pourrait héberger une application dans un cloud public qui utilisera des données stockées dans un cloud privé.

III.2.5. Avantages et inconvénients :

Le cloud a pour avantages coté mobile d'augmenter la fiabilité des applications mobiles en stockant les données auxquelles on peut avoir un accès immédiat, de prolonger la durée de vie de la batterie ainsi que d'économiser de l'espace de stockage sur les smartphones en prenant en charge les opérations qui lui sont coûteuses.

Néanmoins le cloud a quand même un inconvénient qui est le manque de confidentialité et de sécurisation lors de l'accès à ses services à cause de la connexion à internet. [4]

III.2.6. Mobile cloud computing :

Avec l'avènement du cloud computing, de nombreuses entreprises telles que Microsoft et Google adoptent ce nouveau paradigme pour leurs produits.

Les applications mobiles aussi commencent à bénéficier du cloud comme Shazam ou encore Google Voice car malgré tout le développement des smartphones mais ils souffrent toujours d'un manque sérieux de ressources (batterie limitée, capacité de calcul et de stockage réduites) alors que les applications en demandent de plus en plus.

III.2.6.1. Définition :

Le mobile cloud computing est une technologie qui exploite les ressources qu'offre le cloud pour augmenter la performance des mobiles ce qui peut venir à bout des problèmes cités

précédemment et ceci en déchargeant les opérations qui sont fastidieuses pour le mobile sur le cloud [23]

Le mobile cloud computing fait alors référence à une infrastructure dans laquelle le stockage des données et l'accomplissement de certaines tâches sont réalisés en dehors du mobile.

III.2.6.2. Avantages et inconvénients:

Le mobile cloud computing combine les avantages de deux domaines : le cloud computing et la technologie mobile offrant ainsi aux utilisateurs d'applications mobiles la possibilité d'étendre la durée de vie des batteries des smartphones, gagner de l'espace de stockage, assurer une disponibilité des données et d'améliorer la fiabilité des applications et leurs temps de réponse.

Malgré tous les avantages qu'offre le mobile cloud computing, il présente tout de même un inconvénient qui le même que celui du cloud, à savoir la sécurité. [24]

III.2.7. Systèmes utilisant le cloud :

Plusieurs systèmes d'édition collaborative basés sur le cloud ont été proposés dans la littérature, parmi ces systèmes :

-Google Wave : C'est un système d'édition collaborative basé sur le cloud et s'exécutant dans un navigateur, il permet à plusieurs utilisateurs de collaborer en temps réel sur un document xml.
Ce système est basé sur l'approche des transformées opérationnelles et sur l'algorithme d'intégration Jupiter.
Le seul souci avec ce système est la difficulté de vérifier l'intention des utilisateurs. [19]

-SPORC : C'est un système d'édition collaborative temps réel proposé par FeldMan, il travaille avec le cloud et permet à plusieurs utilisateurs d'éditer un document de façon concurrente, pour maintenir la cohérence de ces documents SPORC utilise l'approche des transformées opérationnelles et se base sur un serveur central situé dans le cloud afin d'ordonner les mises à jour concurrentes des utilisateurs, en effet, ce serveur attribue à chaque opération qu'il reçoit un numéro unique et il gère aussi leurs diffusion.

Dans ce système il y a :

Une histoire cryptée associée au serveur central et qui regroupe l'ensemble des opérations qu'il ordonne.

Une histoire associée à chaque site client qui regroupe l'ensemble des opérations diffusées aux clients dans l'ordre établi par le serveur.

Une file d'attente associée à chaque site client qui regroupe les opérations locales de chaque client qui ont été déjà été appliquées sur la réplique locale et qui attendent l'obtention d'un numéro de séquence de la part du serveur central.

Lorsqu'un client génère une opération, il l'applique directement sur sa réplique puis l'ajoute à la fin de sa file d'attente ensuite l'opération va être cryptée et envoyée au serveur central, ce dernier va lui attribuer un numéro de séquence et va l'ajouter à la fin de son histoire cryptée puis l'enverra à tous les clients y compris au client qui l'a générée qui va à ce moment retirer l'opération de sa file d'attente et la place dans son histoire, il n'est pas nécessaire qu'il exécute cette opération car il l'a déjà fait lors de sa génération, par contre, le fait d'avoir exécuté cette opération avant d'avoir obtenu un numéro de série de la part du serveur peut causer des conflits. Une fois l'opération reçue de la part des clients, c'est l'approche TO qui intervient avant de pouvoir l'exécuter. [25]

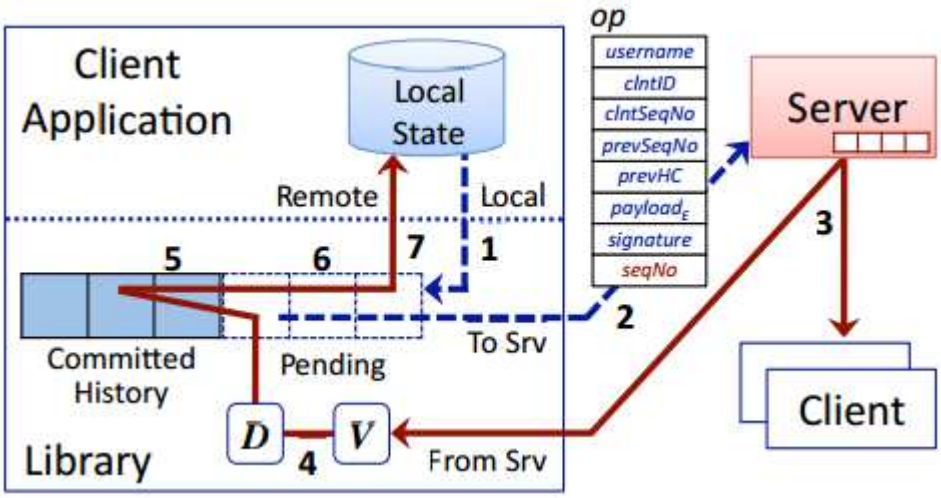

Figure.10. Fonctionnement de SPORC

L'inconvénient de ce système est l'utilisation d'un serveur central qui peut vite être saturé. [26]

-CloneDoc : C'est un système d'édition collaborative temps réel qui a été proposé par Kosta après SPORC qui a le même but que lui et qui vise à pallier à ses limites. Il utilise la plateforme Clone2Clone qui regroupe les clones des mobiles. [25]

Définition du clonage :

Le clonage : En général, c'est l'opération qui permet de créer une copie d'un objet, dans notre cas, un clone est une copie virtuelle du mobile qui contiendra les mêmes données que lui et assurera les mêmes fonctions.

Afin de préserver les ressources du mobile, ce système lègue les tâches qui lui sont fastidieuses au clone situé dans une plateforme cloud nommée ClonetoClone (c2c), cette dernière associe à chaque mobile un clone dans le cloud et relie les clones avec un réseau pair à pair ce qui favorise l'échange de données entre les clones.

Tout comme dans SPORC, l'ordonnancement et la diffusion des opérations concurrentes se fait grâce à un serveur central. [26]

Le clone situé au niveau du cloud reçoit les opérations provenant du mobile et s'occupe de plusieurs tâches à sa place puis le met à jour, le clone soumet les opérations qu'il reçoit au serveur central puis transforme les opérations des autres utilisateurs qu'il reçoit de la part du serveur.

Là aussi comme pour SPORC, le clone maintient une file d'attente et une histoire et se comporte de la même façon qu'un client dans SPORC, c'est-à-dire qu'il soumet au serveur les opérations qu'il reçoit du mobile,

Quand l'utilisateur se déconnecte, son clone continue à recevoir les opérations des autres utilisateurs et les exécute localement, lorsque l'utilisateur se reconnecte, le clone le met à jour automatiquement.

Des études ont montré que l'utilisation de CloneDoc et c2c permet une économie d'énergie pour les mobiles en raison du déchargement des tâches lourdes dans le cloud et une économie de bande passante grâce au déchargement de la communication, néanmoins le serveur central peut causer des conflits. [27]

Ce que nous allons faire dans notre application c'est d'utiliser des clones placés dans le cloud afin de préserver les ressources des mobiles mais ils communiqueront par un réseau pair à pair, ce qui nous évitera les conflits liés au serveur central tout comme dans SPORC et CloneDoc et nous permettra d'établir un échange de données dynamique entre les clones.

III.3. Conclusion :

Dans ce chapitre, nous avons parlé du cloud qui constitue une solution efficace afin de pallier au manque de ressources dont souffrent les ressources, en effet, il leur permet d'augmenter leurs capacités de stockage et d'augmenter la durée de vie de leurs batteries en gérant les tâches fastidieuses à leurs places.

D'ailleurs il existe plusieurs systèmes d'édition collaborative basés sur le cloud dont SPORC et CloneDoc, nous verrons par la suite comment nous allons utiliser cette solution dans notre application.

Chapitre 3

IV.1. Introduction :

Dans ce chapitre, nous allons nous intéresser à la conception et à l'implémentation de notre application qui est une application d'édition collaborative mobile dans le Cloud. Pour cela nous allons mettre en vue tous les outils avec lesquels nous avons travaillé mais aussi présenter l'architecture de notre application qui est une application mobile basée sur android et qui manipulera un document partagé un peu spécial qui est une carte géographique (map), cette application peut être utilisée dans différents domaines notamment dans celui du tourisme ou les utilisateurs postent des descriptions dans différentes positions de la carte ce qui facilitera la vie des prochains visiteurs de ces endroits ou encore dans le cas d'une catastrophe naturelle afin d'aider les secouristes à connaître l'état des lieux et intervenir.

IV.2. Présentation du modèle :

Nous visons à permettre à plusieurs utilisateurs équipés de dispositifs mobiles et connectés à travers un réseau d'éditer une carte commune à tout moment sans utiliser un serveur central pour synchroniser la mise à jour sur la carte partagée.
Pour assurer la disponibilité des données, chaque utilisateur à sa propre copie de la carte partagée. Cependant, un problème crucial se présente lors de la conception d'une carte partagée avec une architecture répliquée et une communication de messages arbitraires entre les appareils mobiles. Ce problème est le maintien de la cohérence (ou de convergence) de toutes les cartes partagées avec une consommation minimale des ressources des appareils mobiles (batterie et puissance de calcul).

Pour mieux comprendre le problème, on considère le scénario d'un groupe de touristes qui visite une grande ville où les parkings de stationnement des véhicules sont souvent pleins.
Un groupe de touristes équipé de dispositifs mobiles (ordinateurs portables, téléphones intelligents et PDA), décide de faire une promenade a pied dans cette grande ville pour mieux la découvrir.

Les touristes sont dispersés dans différentes zones de la ville et chacun dispose d'une application d'édition collaborative qui s'exécute dans son dispositif mobile et qui permet de compter le nombre de places vides dans les parkings et les endroits de stationnement de la ville afin de faciliter la vie des autres touristes véhiculés et qui ont du mal à trouver une place pour stationner. Cette application permet de trouver aussi le parking le plus près des sites historiques, restaurant, cinéma, ect. Avant de commencer la promenade, le leader du groupe divise les membres du groupe sur plusieurs zones. Chaque zone est couverte par plusieurs touristes.

La carte (map) se compose d'un ensemble de paires (p, d) où p est la position sur la carte (paire de latitude et longitude) et d est une description de la position p. La description d contient une valeur textuelle et le nombre de places disponibles dans un parking.

Par exemple, (P1, [place, 4]) est un élément de la carte où, P1 est sa position sur la carte et [place, 4] représente sa description. L'application utilise les opérations primitives Add (position, description) et Del (position, description) pour ajouter et supprimer des éléments respectivement. Considérons le scénario dans la figure suivante :

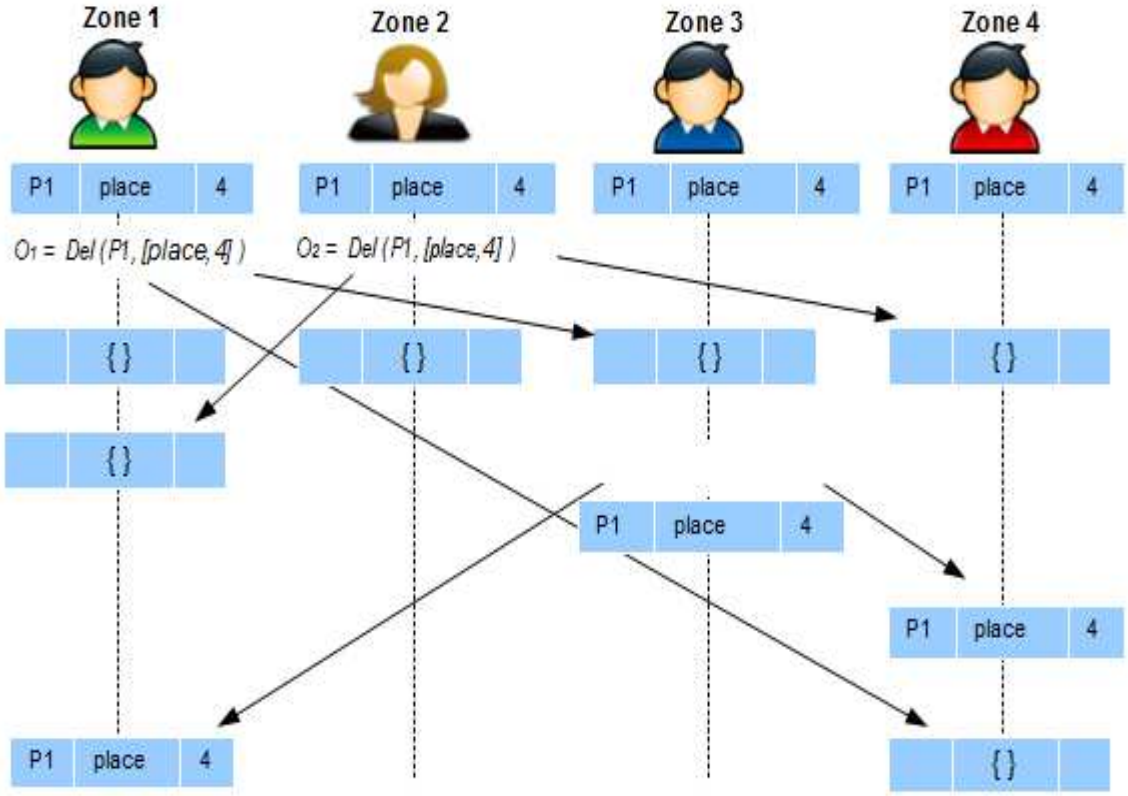

Figure .11. Scénario de divergence

Il est supposé dans ce cas que l'opération est exécutée immédiatement dans la zone locale (site), puis propagée aux autres zones distantes pour être exécutée selon son ordre d'arrivée.
Les flèches du graphique représentent la propagation des opérations de la zone locale vers les zones distantes. Chaque ligne verticale dans le graphique représente les actions effectuées par un touriste dans sa zone de promenade.

Par exemple, dans la zone 1'opération O1 est exécutée en premier, puis elle est suivie par les opérations O2 et O3.
Dans un premier temps, les états des cartes contiennent l'élément (P1, [place, 4]). Cela veut dire qu'il existe 4 places pour stationnement dans la position P1.
Pendant sa promenade, un touriste de la zone 1 effectue l'opération Del (P1, [place, 4]) et un autre touriste de la zone 2 applique simultanément l'opération Del (P1, [place, 4]) pour supprimer l'élément (P1 , place, 4), comme illustré sur la figure 8.

Ces opérations sont propagées à toutes les zones et exécutées selon leurs formes reçues. Le scénario de la figure 11 illustre deux problèmes d'incohérence qui peuvent se produire dans une édition concurrente. Le premier problème c'est la divergence, et le second c'est la violation de causalité.

IV.2.1. Contribution :

Pour faire face à ces problèmes, nous proposons d'utiliser l'approche de transformation opérationnelle (TO) pour maintenir la cohérence des cartes communes. Il permet d'assurer la cohérence d'une manière décentralisée sans avoir besoin de l'ordre global. Elle permet aux utilisateurs de modifier simultanément les données partagées et d'échanger leurs mises à jour dans un ordre quelconque.

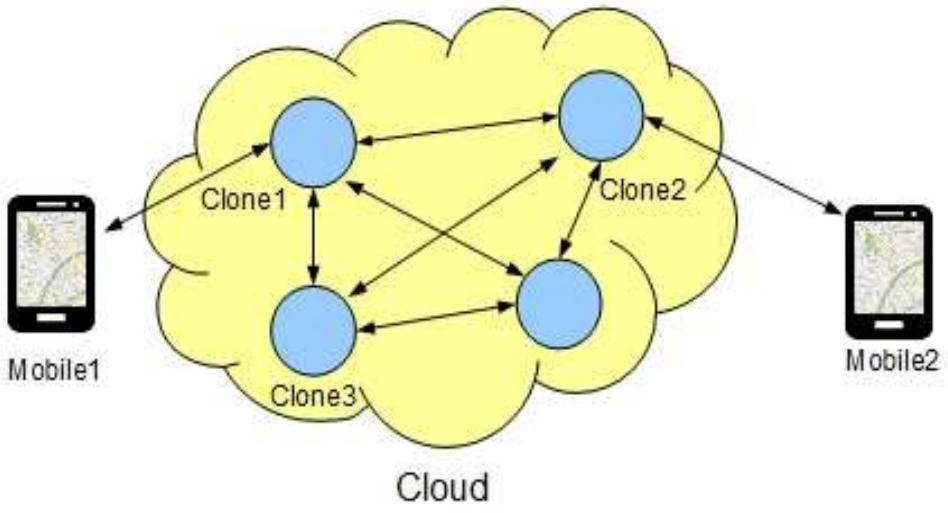

Figure.12. Communication entre mobiles participant à la collaboration

D'autre part, pour économiser les ressources de appareils mobiles, nous proposons de déléguer l'exécuter des taches lourdes en terme de consommation de mémoire de stockage et d'énergie de batterie au Cloud.

Pour chaque mobile doté de notre application, la communication entre lui et les autres mobiles va se faire via le Cloud, c'est-à-dire qu'on va cloner les mobiles qui participent à la collaboration au niveau du Cloud, le clone aura pour mission d'intégrer et de transformer l'opération locale crée et exécutée au niveau du mobile puis de la diffuser aux autres clones, les clones qui reçoivent les opérations distantes vont les intégrer et les transformer puis les renvoyer aux mobiles qui leurs sont associés, les clones auront tous le même rôle et vont communiquer sans passer par un élément central (en pair à pair) ce qui garantit les propriétés suivantes : la dynamicité (les utilisateurs peuvent rejoindre ou quitter le réseau quand ils le

désirent), le passage à l'échelle (c'est la capacité à supporter un grand nombre d'utilisateurs) et la résistance aux pannes.
Le fait de laisser les clones situés dans le Cloud s'occuper de quelques opérations dont les mobiles sont responsables va augmenter la performance de l'application.

IV.3. Protocole de contrôle de concurrence :

Nous avons conçu un nouvel algorithme pour gérer toutes les mises à jour simultanées qui se produisent dans l'application d'édition collaborative mobile.
Cet algorithme repose sur la réplication des cartes partagées sur l'ensemble des mobiles participant à la collaboration afin de fournir un accès aux données sans contraintes, et le modèle de cohérence basé sur la convergence des répliques et la dépendance causale entre les opérations émises par les utilisateurs.

IV.3.1. Carte (Map) partagée :

La carte est constituée d'un ensemble de paires de positions *p* auxquelles les utilisateurs vont ajouter des descriptions, une position est constituée d'un couple (longitude, latitude) et la description correspond à une valeur textuelle, pour modifier la carte, nous avons deux opérations essentielles qui sont ***Add(p,d)*** qui ajoute une description *d* à la position *p* et ***Del(p,d)*** qui supprime la description *d* qui se trouve à la position *p* et ***Nop()*** qui a aucun effet sur la carte partagée.
Il est à noter qu'une seule et même position peut contenir plusieurs descriptions et ces modifications en un seul point de la carte peuvent être concurrentes.

IV.3.2. Modèle d'éditeur collaboratif :

Notre application d'édition collaborative vise à permettre à plusieurs utilisateurs d'éditer de façon collaborative une carte (map) partagée à tout moment en ajoutant chacun d'eux une description à une position donnée de la carte, pour cela chaque utilisateur dispose d'une copie de la carte partagée qu'il peut modifier librement puis envoie ses modifications aux autres utilisateurs.

Notre éditeur collaboratif est distribué et les mobiles communiquent entre eux sans passer par un serveur central. Notre éditeur est basé sur l'approche des transformées opérationnelles afin de garantir la cohérence des données, pour cela il associe à chaque site un log (buffer) afin d'y enregistrer toutes les requêtes exécutées.

Chaque requête de l'utilisateur est de la forme *(c, n, dep, o)* avec *c* qui est l'identifiant du site, *n* c'est son numéro de série, *dep* c'est l'identifiant de la requête qui précède causalement celle-ci en sachant que l'identifiant d'une requête est formé de *c* + n (si *dep* correspond à null

cela veut dire que cette requête ne dépend causalement d'aucune autre requête) et enfin *o* c'est l'opération à exécuter sur la carte.

Étant donné un journal (log) *L*, *L* [i] désigne la i ^{ième} requête de *L*; | *L* | est la longueur de *L*, *L* [i, j] est le sous-journal (sub-log) de *L* allant de son i^{ème} à la j^{ème} requêtes avec $0 < i \leq j \leq n-1$ tel que $n = | L |$.
Nous nous sommes intéressés au modèle d'éditeur collaboratif proposé par IMINE [1], car il offre un nouveau cadre de coordination pour les éditeurs collaboratifs en temps réel qui se caractérisent par les aspects suivants:

i. L'utilisation d'un système de réplication optimiste fournissant un accès simultané à des documents partagés.

ii. la conservation de lien de causalité grâce à une nouvelle technique simple appelée dépendance sémantique.

En outre, ce modèle supporte les groupes dynamiques dans le sens où les utilisateurs peuvent quitter ou joindre les groupes à tout moment. Quand un nouvel utilisateur veut joindre un groupe de collaboration, il demande l'état courant du document et le journal (log) actuel de l'utilisateur le plus proche afin de commencer la collaboration avec les membres de ce groupe.

Un état stable dans un éditeur collaboratif en temps réel est atteint lorsque toutes les opérations générées ont été effectuées sur tous les sites. Pour cela les critères suivants doivent être assurés [1]:

Définition 1. (Critères de cohérence) Un éditeur collaboratif en temps réel est cohérent si est seulement si, il satisfait les propriétés suivantes:

1) la préservation de la dépendance: si O_1 dépend de O_2 alors O_1 est exécuté avant o_2 sur tous les sites.

2) la convergence: lorsque tous les sites ont effectué le même ensemble d'opérations, les copies des documents partagés sont identiques.

A l'état stable, journaux (logs) des sites ne sont pas nécessairement identiques parce que les opérations simultanées peuvent être exécutées dans un ordre différent. Néanmoins, ces journaux doivent être équivalents dans le sens où elles doivent aboutir au même état final.

Néanmoins, ces journaux doivent être équivalents dans le sens où elles doivent aboutir au même état final.

Définition 2. (Journal Equivalent) Deux journaux sont équivalents si et seulement si ils produisent le même état lorsqu'ils sont appliqué à un état donné *Map*.

Pour éviter le problème de divergence, une classe de journaux est définit dans [1] pour permettre de construire des chemins de transformation menant à une convergence des données.

Définition 3. Un journal L est canonique si et seulement si L est la concaténation de deux sous-journaux L_{Add} (ensemble d'opérations Add insertion) et L_{Del} (ensemble de opérations supprimer del) de telle sorte que L_{Add} ne contient pas de requêtes de suppression et L_{Del} ne contient pas de requêtes d'insertion.

Les journaux L_{Add} et L_{Del} peuvent également contenir des requêtes de modification. Dans un journal canonique, on impose un ordre sur les requêtes d'insertion et de suppression: une requête d'insertion doit toujours être avant les requêtes de suppression. Il à noter que les journaux vides et les journaux contenant uniquement des requêtes d'insertions et / ou suppressions sont également canoniques.

IV.3.3. Relations de dépendance :

Pour résoudre le problème de la cohérence des données, nous utilisant une relation de dépendance minimale [1] qui est indépendante du nombre d'utilisateurs et, par conséquent, permet à des groupes dynamiques.
A chaque site est associé un log L dans lequel on garde une trace de toutes les opérations exécutées, les opérations ont une dépendance causale avec les opérations exécutées auparavant selon la condition 1 et 2 de la précédence causale.
Dans le cadre de notre application, nous allons définir les dépendances causales suivantes :
On dit que O_1 précède causalement O_2 si :

- O_1= Add(p,d) et O_2=Del(p,d') dans ce cas O_1 doit s'exécuter avant O_2 sur toutes les répliques. **(df1)**

- O_1= Del(p,d') et O_2= Add(p,d) dans ce cas O_1 doit s'exécuter avant O_2 sur toutes les répliques. **(df2)**

Sinon, les opérations sont concurrentes est peuvent être exécuter a importe quel ordre.
Nous définissons deux relations de dépendance causale, (i) la suppression de la description dépend de l'opération qui a inséré cette description selon (**df1**), et (ii) l'ajout de la même description à la même position dépend de l'opération qui a ajouté cette description selon (**df2**).

En outre, il n'y a pas de dépendance entre les demandes de suppression et elles peuvent être exécutées entre eux dans un ordre quelconque. Dans ces cas, chaque opération n'a plus qu'à stocker l'identifiant de l'opération dont elle dépend directement.

IV.3.4. Fonctions de transformations :

Nous allons utiliser deux formes de fonctions de transformation proposé par IMINE (Univ Nancy & LORIA) et MECHAOUI (Univ Mostaganem) : La transformation inclusive (IT) et la transformation exclusive (ET).
La transformation inclusive va prendre en considération l'effet de toutes les opérations concurrentes, la transformation exclusive quant à elle va nous permettre de détecter les dépendances causales au sein du log.

```
1: IT(q1, q2) = q'
2: q'1 ⟵ q1
3: Choice of q1 and q2
4: Case: q1 = Add1 and q2 = Add2
5: if (p2 = p1 and d2 = d1 ) then Nop()
6:    else q'1.o ⟵ Add(p1;d1)
7: Case: q1 = Add1 and q2 = Del2
8: if (p2 = p1 and d2 = d1 ) then Nop()
9:    else q'1.o ⟵ Add(p1;d1)
10: Case: q1 = Del1 and q2 = Add2
11: if (p2 = p1 and d2 = d1 ) then q'1.o ⟵ Del(p1;d1)
12: Case: q1 = Del1 and q2 = Del2
13: if (p2 = p1 and d2 = d1 ) then Nop()
14:   else q'1.o ⟵ Del(p1;d1)
15: end choice
16: return q'1
```

Algorithme 1 : Transformation inclusive.

L'algorithme IT prend en charge les opérations concurrentes, l'algorithme 1 pressente tous les cas possible. Cet algorithme permet d'inclure l'effet d'une opération sur l'ensemble des opérations existantes dans le Log du clone.

L'algorithme de transformation inclusive TI est utilisé pour exécuter des requêtes simultanées dans un ordre quelconque. Tous les cas de notre algorithme TI sont donnés dans l'algorithme 1.

```
1: ET(q1, q2) = q'1
2: q'1 ⟵ q1
3: Choice of q1 and q2
4: Case: q1 = Add1 and q2 = Add2
5: if (p1 = p2) or (p1 = p2 and d1 = d2) then
   q'1.o ⟵ Add(p1;d1)
6: else q'1.o ⟵ Add(p1;d1)
7: Case: q1 = Add1 and q2 = Del2
8: if (p1 = p2 and d2 = d1 ) then return "Undefined"
9: else q'1.o ⟵ Add(p1;d1)
10: Case: q1 = Del1 and q2 = Add2
11: if (p1 = p2 and d2 = d1 ) return "Undefined"
12: else q'1.o ⟵ Del(p1;d1)
13: Case : q1 = Del1 and q2 = Del2
14: if (p1 6= p2) or (p1 = p2 and d1 = d2) then
    q'1.o ⟵ Del(p1;d1)
15: else q'1.o ⟵ Del(p1;d1)
16: end choice
17: return q'1
```

Algorithme 2 : Transformation Exclusive.

L'algorithme ET permet de détecter les dépendances causales entre les opérations. Autrement dit, il permet de d'exclure d'effet d'une opération de chacune des opérations du log. Cela permet de calculer un contexte d'exécution minimal de cette opération dans les autres clones.

Lorsque deux requêtes insèrent deux éléments à la même position avec la même description (ils sont en conflit), un choix doit être fait (lignes 4-5). L'algorithme IT retourne la requête d'inactivité (idle) dont le paramètre de fonctionnement est Nop () qui a un effet null sur l'état partagé. D'autre part, dans le cas de deux requêtes de suppression à la même position, l'algorithme TI renvoie la requête Nop () (elle n'a aucun effet sur l'état du map).

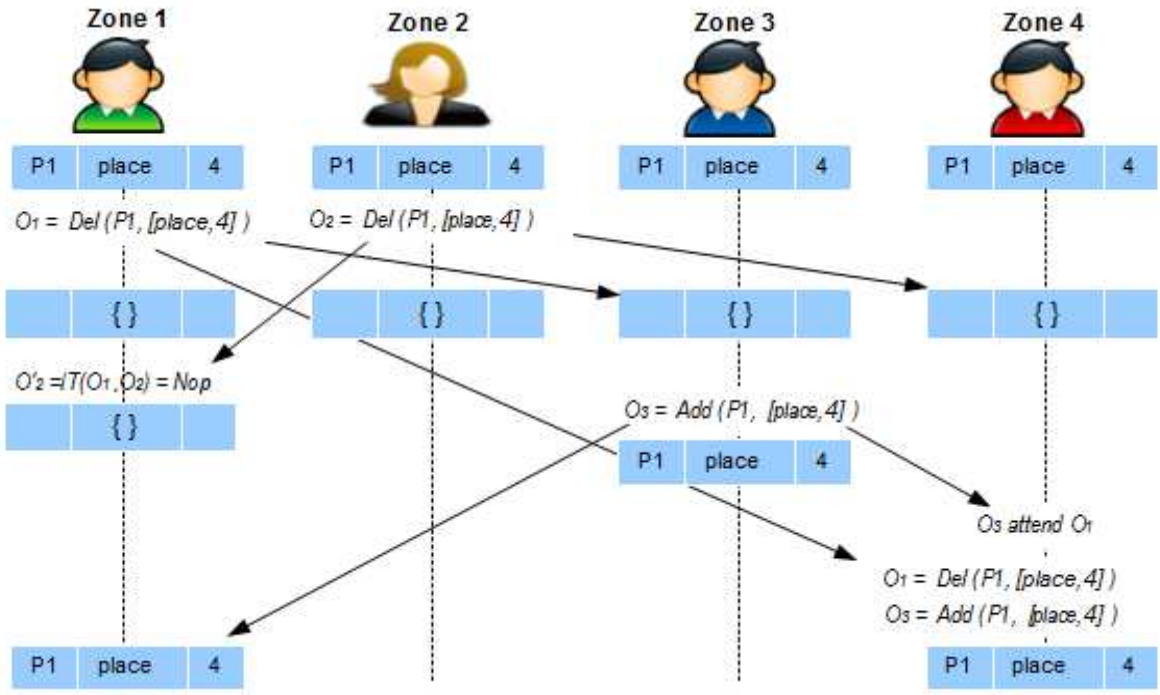

Figure.13. Scénario de convergence

L'utilisation de l'algorithme de transformation exclusive ET nous permet de détecter la relation de dépendance causale entre les requêtes dans le Log.
Lorsque ET (q1, q2) retourne « Undefined » alors la requête q2 dépend causalement de la requête q1.
Par exemple, q1 est une requête d'insertion et q2 et une requête de suppression, ces deux requêtes ont ajouté et supprimé la même position et la même description, l'élément ajouté par q1 doit précéder (ou il est avant) celle ajoutée par q2 (voir l'algorithme 2 ligne 7- 8). Tous les différents cas de notre algorithme ET sont présentés dans l'algorithme 2.

La figure 13 illustre notre solution de proposition de problème de divergence (présenté dans la figure 11) à l'aide des fonctions de transformation.

Dans la zone 1, lorsque O2 est reçu, elle est transformée inclusivement par rapport O1 pour donner O'2 = IT (O1, O2) = Nop où O'2 n'a aucun effet sur l'état de la carte, car elle est déjà vide.
Pour régler le problème de la causalité, quand le touriste de la Zone3 génère O3, elle est transformée exclusivement sur O1 (ET (O3, O1)) et détecte que cela dépend de O1. Donc, O3 doit être exécutée après O1 dans toutes les autres zones.
Quand O3 est reçu dans la Zone 4, elle attend O1 avant de s'exécuter.

IV.4. Protocole de synchronisation :

Notre système d'édition collaborative est basé sur deux couches de synchronisation.
La première est entre le mobile et son clone assurée par une communication unicast et la deuxième et entre les clones qui forment le groupe de collaboration avec une communication multicast.

IV.4.1. Communication :

IV.4.1.1. Entre Mobile et Clone :

La communication entre le mobile et son clone situé dans le Cloud se fera en mode unicast, c'est-à-dire que le mobile ne doit envoyer ses mises à jour qu'au clone qui lui est associé, cette communication est faite grâce à JGroups.
JGroups est un ensemble d'outils conçu pour s'échanger des messages de façon fiable, il peut être utilisé pour créer des clusters dont les nœuds peuvent communiquer en s'envoyant des messages les uns aux autres, un nœud peut alors envoyer des messages à tous les autres nœuds ou bien à un seul d'entre eux.
Ce système notifie lorsqu'un nouveau nœud rejoint le groupe ou quand un ancien nœud le quitte, il est à noter que chaque groupe est identifié par son nom. [28]
JGroups permet aux développeurs d'établir une messagerie fiable sans rien implémenter chose qui lui fait économiser beaucoup de temps.

IV.4.1.2. Entre Clone et Clone:

La communication entre clones se fera via un réseau pair à pair, c'est-à-dire que tous les clones joueront le même rôle et il n'y aura pas d'élément central.
Lorsqu'un clone va recevoir une mise à jour émanant du mobile, il va la diffuser aux autres clones en multicast grâce à l'utilisation JGroups.

IV.4.2. Synchronisation :

En sachant que chaque mobile est muni d'un log qui sert à sauvegarder les opérations exécutées et que les clones aussi avec une file d'attente en plus, la synchronisation se fait comme suit :

a) La génération d'une opération :

1- Lorsqu'un utilisateur modifie sa map dans son mobile, il génère une opération *op* qui sera exécutée directement au niveau de l'interface puis sera sauvegardée dans le log. La position de l'opération *op* est donnée automatiquement par le GPS du mobile.

2- Le mobile envoie l'opération *op* au clone qui lui est associé s'il est connecté, sinon il exécute toutes ses opérations localement, les sauvegarde dans le log puis les envoie en totalité au clone qui lui est associé une fois qu'il se connecte.

3- Le clone traite les opérations une par une en passant par les étapes suivantes :

- Il crée la requête *q* pour l'opération *op* reçue.
- Le clone calcule le contexte d'exécution minimal de chaque requête q reçu du mobile. Une requête peut dépendre des requêtes précédentes en fonction de l'ordre d'exécution. Le suivi de cette dépendance dans un journal permet d'identifier les opérations qui doivent être exécutées sur tous les clones selon le même ordre. Le protocole de synchronisation du clone utilise une relation de dépendance minimale qui est indépendante du nombre d'utilisateurs, et par conséquent, il est bien adapté pour les groupes dynamiques. En d'autres termes, au lieu de considérer q comme étant dépendante de toutes les requêtes de Log, cette étape permet de réduire ce contexte, en excluant autant que possible des opérations de Log pour donner le fonctionnement transformée q'. Pour résumé cette étape, le clone transforme exclusivement *q* par rapport à son log afin de détecter une éventuelle dépendance causale, et là deux cas sont possibles:

  - La requête q est concurrente à toutes les requêtes qui sont dans le log, c'est-à-dire qu'elle ne dépend d'aucune autre requête, dans ce cas elle peut s'exécuter dans n'importe quel ordre au niveau des autres clones.

  - La requête q dépend d'une autre requête dans le log selon les relations de dépendances **(df1)** et **(df2)** définies dans la section IV.3.3, dans ce cas, cette requête ne doit pas être exécutée avant l'exécution de la requête dont elle dépend.

4- Le clone sauvegarde la requête transformée q' dans son log.

5- Le clone envoie la requête aux autres clones du groupe de collaboration.

b) L'intégration d'une opération :

Quand un clone reçoit une requête distante q' d'un autre clone, l'intégration de cette requête passe par les étapes suivantes:

6- Le clone reçoit la requête distante q' et il vérifie si elle dépend d'une autre requête (dep != null) faisant partie de son log, et là aussi deux cas sont possibles :

- Si la requête dont elle dépend n'est pas encore arrivée alors la requête q' est ajoutée dans une file d'attente (cette file contient les requêtes qui ne sont pas causalement prêtes).
- Sinon, il y a deux possibilités :

  - La requête dont elle dépend existe dans le log alors une transformation inclusive est appliquée sur le log du clone à partir de l'emplacement (indice dans le log) de la requête de dépendance.

  - La requête est concurrente à toutes les autres requêtes sauvegardées (dep=null) alors une transformation inclusive est appliquée sur tout le log.

7- Le clone enregistre la requête transformée q'' dans son log.

8- Le clone envoie l'opération q''.o au mobile qui lui correspond.

9- Une fois reçue, le mobile applique l'opération q''.o sur sa map partagée et la sauvegarde dans son log.

Dans le diagramme de séquence qui suit, on résume toutes les étapes citées précédemment :

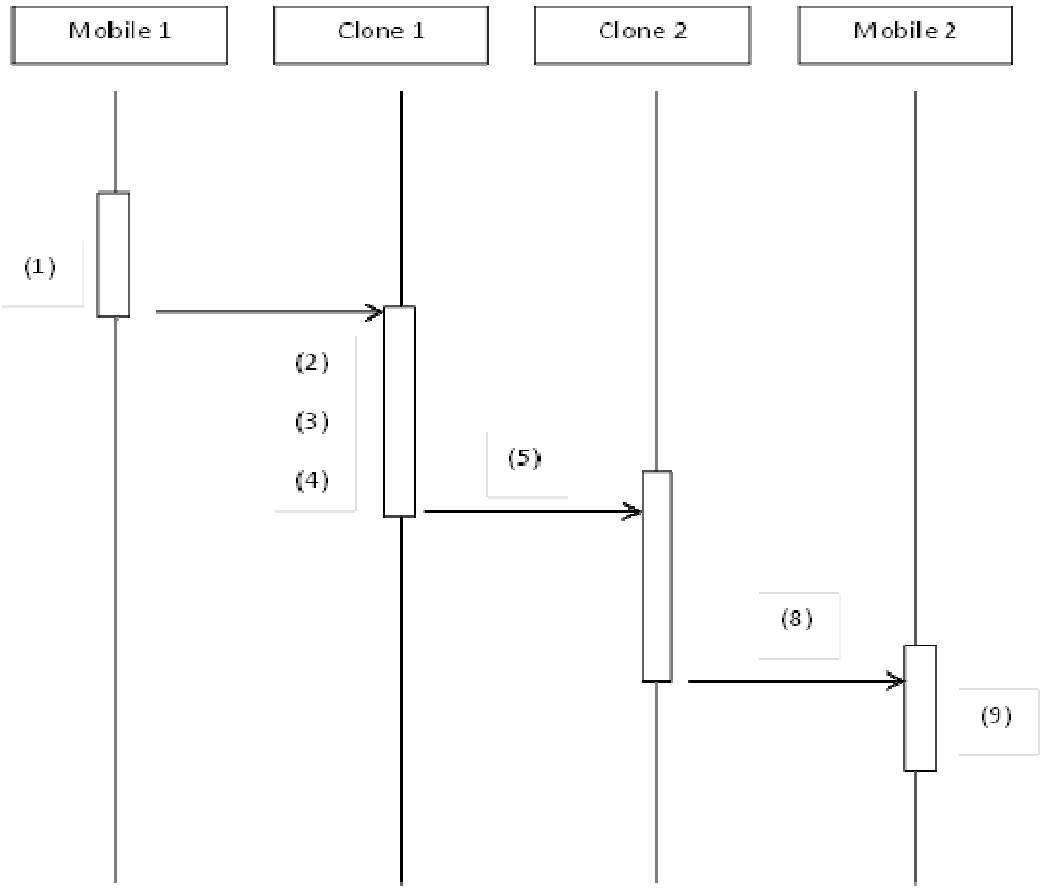

Figure.14. Diagramme de séquence résumant toutes les étapes de la synchronisation

IV.5 Scénario d'exécution :

Pour l'utilisation d'une application d'édition collaborative mobile dans le Cloud, on suppose le scénario suivant :
Une catastrophe naturelle s'est produite, dans ce cas on a un manque sérieux de ressources alors comme alternative on va utiliser des appareils mobiles connectés à un Cloud pour communiquer.

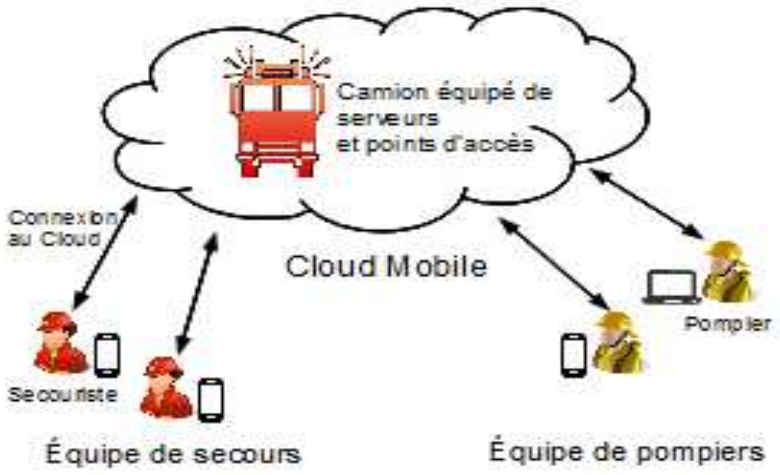

Figure.15. Scénario d'utilisation dans le cas d'une catastrophe naturelle

Une équipe de secouristes est dispersée sur la zone atteinte et chacun d'eux est chargé de faire un état des lieux et envoyer son constat à ses co-équipiers afin de prendre des décisions importantes pour leur mission de sauvetage, ils vont utiliser pour cela des appareils mobiles sur lesquels une application d'édition collaborative est installée, cette dernière aura comme document partagé une carte (map) sur laquelle chacun des secouristes apportera une description comportant le nombre de victimes et la position sur la carte.

Chaque modification va générer une opération qui sera exécutée localement sur le champ, puis l'application donnera la main au clone situé au niveau du Cloud pour propager l'opération aux autres clones après l'avoir exécutée lui-même, les clones exécutent l'opération puis ils vont à leurs tours mettre à jour les mobiles qui leur sont associés en leur diffusant l'opération à exécuter, le Cloud contenant le clone peut être placé dans un camion de pompiers, son utilisation permettra aux mobiles de conserver leur énergie, leurs capacités de calculs et permettre une collaboration à longue durée.

IV.6. Implémentation :

IV.6.1. Environnements de développement :

Afin de concevoir notre application d'édition collaborative mobile, les outils suivants sont nécessaires :

Android : est un système d'exploitation Open Source édité par Google et destiné aux appareils mobiles tels que les smartphones et les tablettes, on le retrouve aussi sur des téléviseurs, des montres ou encore des GPS. [29]
Il supporte plusieurs fonctionnalités parmi lesquelles on cite la géolocalisation qui permet d'intégrer Google Map dans nos applications.
Pour développer avec android, il nous faut un environnement de développement, le SDK (software develoment kit) d'android et le JDK (java development kit) de java. [30]

Le JDK (Java Development Kit) **:** C'est un environnement logiciel pour le développement d'applications java, il contient d'une part le JRE qui est l'ensemble des composants nécessaires au lancement de ces applications ainsi qu'une machine virtuelle Java, et d'autre part un ensemble d'outils pour compiler et débugger le code. [31]

Le SDK (Software Development Kit): Le SDK spécifique à Android nous fournit les outils pour coder en Android, il contient un plugin qui s'intègre à l'environnement de développement, des bibliothèques, un débogueur et un émulateur Android, qui simulera le fonctionnement d'un smartphone, ce qui nous permettra de tester nos applications avec différentes versions d'android. [29]

Android studio **:** C'est le nouvel environnement de développement officiel pour concevoir des applications android proposé par Google, il offre un éditeur complet et robuste avec des outils conçus pour accélérer le développement d'applications et permet de les tester sur différents appareils émulés. [32]

Google Map V2 : C'est un api (panel d'outils proposé par Google Inc) qui nous permet de créer des applications web et mobiles et y intégrer des cartes géographiques, des images satellitaires ou bien des itinéraires grâce à l'obtention d'une clé android que l'on ajoute à notre code, c'est cette dernière qui va invoquer la carte et va nous permettre de l'afficher dans notre application. [33]

IV.7. Présentation de l'application :

Dans cette section, nous allons présenter quelques captures d'écran des fonctionnalités principales de notre application.

A l'ouverture de notre application, nous verrons s'afficher une carte géographique.

Figure.16. Interface de l'application

Puis nous verrons s'afficher la position actuelle de l'utilisateur défini par un marqueur. A ce moment l'utilisateur n'aura plus qu'à cliquer sur le menu qui est associé à sa position actuelle afin de saisir sa description dans la zone de texte et générer ainsi une opération, cette dernière sera envoyé au clone via JGroups afin qu'il puisse l'exécuter et la diffuser aux autres clones à travers JGroups aussi.

Figure.17. Zone de saisie de la description

Après les intégrations et transformations nécessaires citées dans la section IV.4.2, les clones mettent à jour les mobiles qui leurs correspondent.

IV.8 Conclusion :

Dans ce chapitre, nous avons présenté une application collaborative mobile low-cost qui permet d'éditer simultanément une carte partagée. La cohérence de la carte est assurée par un contrôle c protocole de concurrence spécifique basé sur une approche OT. Cette application collaborative peut être utilisée dans différents domaines (urgence, tourisme, médical). Nous espérons que notre application représente une solution simple et efficace pour faire face surtout aux problèmes de support de communication en cas d'urgence (catastrophe naturelle).

Conclusion générale :

Dans ce mémoire, nous avons passé en vue tous les facteurs participant au développement d'une application collaborative mobile fiable. Cependant, Concevoir le contrôle de concurrence pour les applications collaboratives mobiles est considérée comme un défi, étant donné que les dispositifs mobiles sont limités par l'insuffisance des ressources qui doivent être considérées lors de la combinaison des environnements mobiles et de Cloud. Notre plus est donc de travailler avec le cloud qui est un modèle en émergence pour pallier aux contraintes des mobiles (autonomie de la batterie, mémoire limitée) car nous avons remarqué qu'il augmente les performances de ce genre d'applications en offrant toute une panoplie de ressources et en prenant en charge les opérations qui sont couteuses pour le mobile en terme d'énergie, de mémoire et de capacité de calcul.

Nous avons présenté dans ce mémoire une solution pour un problème de cohérence des données lorsque les utilisateurs mobiles sont clonés dans le Cloud et modifient simultanément les données partagées répliquées dans le dispositif mobile et son clone. Cette solution est inspirée du protocole proposé par IMINE [1], ce dernier est basé sur l'approche des transformées opérationnelles (TO) qui est l'une des meilleures techniques qui supporte la collaboration en utilisant des dispositifs mobiles, elle est utilisée par les éditeurs collaboratifs en temps réel pour permettre à plusieurs utilisateurs de modifier le même document partagé simultanément. En plus, cette approche tolère la collaboration même en mode déconnecté.

Notre objectif est la réalisation d'une application mobile décentralisée qui permet de modifier simultanément une carte (map) partagée entre plusieurs utilisateurs sans contraintes. C'est-à-dire pas de serveur central pour la synchronisation des données et pas de consommation d'énergie et d'espace de stockage dans le mobile. Ceci est réalisé grâce à la technique de clonage des mobiles dans le Cloud. Toutes les taches lourdes de notre application telles que la synchronisation et la communication entre les autres clones en mode pair à pair sont assurées par le clone. Dans cette application nous avons utilisés de nouvelles fonctions de transformation proposées par IMINE (Univ Nancy & Loria) et MECHAOUI afin de s'adapter avec les caractéristiques spécifiques de la carte (map).

Enfin, ce travail nous a permis d'enrichir nos connaissances sur les nouveaux domaines de développent d'applications mobiles qui sont en plein essor ces dernières années et nous donne aussi de nouvelles ambitions pour continuer sur le chemin de la recherche scientifique.

[1] : Abdessamad Imine, Conception Formelle d'Algorithmes de Réplication Optimiste Vers l'Edition Collaborative dans les Réseaux Pair à Pair, Thèse de doctorat, Université Henri Poincaré Nancy 1, 2006.

[2] : Stéphane Weiss, Edition collaborative massive sur réseaux Pair à Pair, Thèse de doctorat, Ecole doctorale IAEM Lorraine, 2010.

[3] : Clarence Leung, Operational transformation in cooperative software systems, Article de la revue McGill Science Undergraduate Research, School of Computer Science, McGill University, Montreal, QC, 2013.

[4] : Mounir Tlili, Infrastructure P2P pour la Réplication et la Réconciliation des Données, Thèse de doctorat, Université de Nantes Ecole doctorale STIM, 2011.

[5] : Randolph Aurel Josias Oboubé, Convergence et sécurité d'accès dans les systèmes d'édition collaborative massivement répartis, Thèse pour l'obtention du diplôme de docteur en philosophie, Université de Montréal, 2014.

[6] : Philippe Quénnec, Gérard Padiou , Systèmes et algorithmes répartis-Données réparties, Cours, 2015.

[7] : Yasushi Saito, Optimistic replication, Journal of ACM Computing Surveys, page : 42-81, 2005.

[8] : Sandy Citro, A Framework for Real Time Collaborative Editing in a Mobile Replicated Architecture, Thèse pour l'obtention du diplôme de docteur en philosophie, RMIT University Australia, 2007.

[9] : Moulay Driss Mechaoui, Abdessamad Imine, Fatima Bendella, Un modèle générique de Garbage Collection pour les éditeurs collaboratifs basé sur l'approche TO dans les environnements P2P et mobiles, article CIIA, Saida, 2011.

[10] : Stéphane Martin, Édition collaborative des documents semi-structurés, Thèse de doctorat, Université de Provence, 2011.

[11] : Gérald Oster, Réplication optimiste et cohérence des données dans les environnements collaboratifs répartis, Thèse de doctorat, Université Henri Poincaré Nancy 1, 2005.

[12] : Moulay Driss Mechaoui, Chawki Benchehida, Bilal Benaouda, Abdessamad Imine, Coordination Model for Mobile Collaborative Mapping, Rapport de recherche, University of Sciences and Technology Oran 'Mohamed Boudiaf' USTO-MB Algeria , 2015.

[13] : Bin Shao, Du Li, Ning Gu, A sequence transformation algorithm for supporting cooperative work on mobile devices. Article du Proceedings de la conférence ACM intitulée Computer supported cooperative work, page : 159-168, 2010.

[14] : Chengzheng Sun, Clarence Ellis, Operational Transformation in Real-Time Group Editors: Issues, Algorithms, and Achievements, Article du Proceedings de la conférence ACM intitulée Computer supported cooperative work, page : 59-68, USA, 1998.

[15] : Santosh Kumawat, Ajay Khunteta, A Transformation based New Algorithm for Transforming Deletions in String Wise Operations for Wide-Area Collaborative Applications, Article du journal International Journal of Computer Applications, 2010.

[16] : Aurel Randolph , Hanifa Boucheneb , Abdessamad Imine , and Alejandro Quintero, On Consistency of Operational Transformation Approach, Article de recherche, Ecole Polytechnique de Montr´eal Canada.

[17] : Nicolas Vidot, Convergence des Copies dans les Environnements Collaboratifs Répartis, Thèse de doctorat, Université Montpellier II France, 2002.

[18] : Santosh Kumawat, Ajay Khunteta, A Survey on Operational Transformation Algorithms Challenges, Issues and Achievements, Article du journal International Journal of Computer Applications, 2010.

[19] : Christoffer HirsimaaI, Martin Nycander, An Analysis on Operational Transforms, Thèse de bachelor, Royal Institute of Technology Suisse, 2010.

[20] : Mobile cloud computing, http://fr.slideshare.net/AdniBipa/mobile-cloud-computing-28084220?qid=eb4d19c9-a58e-4c34-973b-e2c8ee7dd1ce&v=&b=&from_search=10, Consulté le : 09/02/2016.

[21] : Thibaud Chardonnens, Les enjeux du Cloud Computing en entreprise, Université de Fribourg, Suisse, Rapport de recherche, 2012.

[22] : SaaS, PaaS, IaaS, Cloud : définitions-site archivé-CMS-SPIP, http://icp.ge.ch/sem/cms-spip/spip.php?article962, Consulté le : 09/02/2016.

[23] : Antonio Miguel Rosado Da Cruz, Sara Paiva, Modern Software Engineering Methodologies for Mobile and Cloud Environments, Livre, Etats Unis d'Amérique, 2016.

[24] : Mobile cloud computing, http://fr.slideshare.net/AdniBipa/mobile-cloud-computing-28084220?qid=eb4d19c9-a58e-4c34-973b-e2c8ee7dd1ce&v=&b=&from_search=10, Consulté le : 10/05/2016.

[25] : Sokol Kosta, Claudiu V.Perta, Julinda Stefa, Pan Hui, Alessandro Mei, Clone2Clone (C2C): Enable Peer-to-Peer Networking Smartphones on the Cloud, Rapport de recherche, Université technique de Berlin, 2012.

[26] : Ladjel Bellatreche, Yannis Manolopoulos, Model and Data Engineering, Livre, Suisse, 2015.

[27] : Sokol Kosta, Vasile Claudiu Perta, Julinda Stefa, Pan Huiy, and Alessandro Mei, CloneDoc: Exploiting the Cloud to Leverage Secure Group Collaboration Mechanisms for Smartphones, Rapport de recherche, Sapienza University of Rome Italie.

[28] : Reliable group communication with JGroups, http://www.jgroups.org/manual/html/index.html, Consulté le : 15/05/2016.

[29] : Android—SDK Android, http://www-igm.univ-mlv.fr/~dr/XPOSE2011/SDKAndroid/android.html, Consulté le : 14/05/2016.

[30] : Android Studio premier contact, http://fr.slideshare.net/djafaka/android-studio-premier-contact, Consulté le : 14/05/2016.

[31] : What is Java Development Kit (JDK) ?, https://www.techopedia.com/definition/5594/java-development-kit-jdk , Consulté le : 15/05/2016.