**RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE**
**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**
**Université Abderrahmane Mira de Béjaia**

# Mémoire de fin de cycle

*Pour l'Obtention du Diplôme de Master en Informatique*
*Option : Réseaux et Systèmes Distribués*

## *Thème*

# *Architecture pour le Cloud Computing Mobile en mode P2P centralisé*

*Présenté par :*
***ADNANI Yahia***
***BELHABIB Amine***

**Devant le jury composé de :**

| | |
|---|---|
| **Président :** | Dr SIDER Abderrahmane. |
| **Examinateur :** | Melle HOUARI Rima. |
| **Examinateur :** | Dr OMAR Mawloud. |
| **Promoteur :** | Mr SAADI Mustapha. |
| **Co-Promoteur :** | Melle IKKEN Sonia. |

**Année 2013**

# Table des matières

**Table des matieres** **i**

**Liste des figures** **vi**

**Liste des tableaux** **vii**

**Introduction générale** **3**

**1 *Les Concepts de base du Cloud Computing*** **6**

1.1 Introduction . . . . . 6

1.2 Qu'est-ce-que le Cloud Computing ? . . . . . 7

1.3 Offres du Cloud Computing . . . . . 7

1.4 Les modèles de déploiements du Cloud Computing . . . . . 9

1.4.1 Cloud publique . . . . . 9

1.4.2 Cloud privé . . . . . 10

1.4.3 Cloud communautaire . . . . . 10

1.4.4 Cloud hybride . . . . . 10

1.5 Conclusion . . . . . 10

**2 *Cloud Coumputing Mobile*** **12**

2.1 Introduction . . . . . 12

2.2 Qu'est-ce-que le Cloud Computing Mobile ? . . . . . 12

2.3 Architecture du Cloud Computing Mobile : . . . . . 13

2.4 Avantages du Cloud Computing Mobile . . . . . 14

2.4.1 Extension de la durée de vie de la batterie . . . . . 15

2.4.2 Amélioration de la capacité de stockage et la puissance de calcul : . 15

2.4.3 Amélioration de la fiabilité . . . . . 15

2.5 Application du Cloud Computing Mobile : . . . . . 16

2.5.1 Commerce mobile . . . 16
2.5.2 Mobile Learning . . . 16
2.5.3 Services médicaux mobiles . . . 17
2.5.4 Mobile Gaming . . . 18
2.5.5 Autres applications pratiques . . . 18
2.6 Conclusion . . . 19

**3 *Etat de l'art sur le Cloud Computing* 20**
3.1 Introduction . . . 20
3.2 Cloud computing P2P . . . 20
3.2.1 Systèmes P2P . . . 20
3.2.1.1 Définition . . . 21
3.2.1.2 Architectures . . . 21
3.2.2 Cloud P2P . . . 24
3.2.2.1 Architectures de Cloud Computing . . . 24
3.2.2.2 Pourquoi un Cloud P2P ? . . . 25
3.2.2.3 P2P VS Volunteer computing (VC) . . . 25
3.2.2.4 Arechitecture P2P CS . . . 26
3.3 Méthodes de débarquement du calcul . . . 28
3.3.1 Communication client/serveur . . . 28
3.3.2 Virtualisation . . . 30
3.3.3 Agent mobile . . . 31
3.3.4 Discussion . . . 31
3.4 Gestion de mobilité . . . 32
3.5 Conclusion . . . 33

**4 *Architecture pour le Cloud mobile en mode P2P centralisé* 34**
4.1 Introduction . . . 34
4.1.1 Principe de la solution . . . 34
4.2 Conception d'une architecture pour le Cloud mobile en mode P2P . . . 35
4.3 Mécanisme de collaboration en temps réel . . . 36
4.3.1 Présentation du mécanisme . . . 36
4.3.2 Description des algorithmes : . . . 39

4.3.2.1 Algorithme du mobile $M_i$ . . . 39
4.3.2.2 Algorithme du clone $C_i$ . . . 39
4.3.2.3 Algorithme du SC . . . 42
4.4 Système de partage de fichiers . . . 47
4.4.1 Presentation du système . . . 47
4.4.2 Description des algorithmes . . . 48
4.4.2.1 Algorithme $\mathbf{M}_i$ (mobile i). . . . 49
4.4.2.2 Algorithme $\mathbf{C}_i$ (clone i). . . . 49
4.4.2.3 Algorithme SC (super clone). . . . 50
4.5 Gestion de mobilité . . . 51
4.6 Discussion de la solution proposée . . . 53
4.6.1 Avantages . . . 53
4.6.2 Inconvénients . . . 53
4.7 Conclusion . . . 54

**5 *Implementation* 55**

5.1 Introduction . . . 55
5.2 Description de RMI . . . 55
5.2.1 Utilisation de RMI . . . 56
5.2.1.1 L'interface . . . 56
5.2.1.2 L'implémentation . . . 56
5.2.1.3 Stubs et Skeletons . . . 56
5.2.1.4 Le registre RMI . . . 57
5.2.1.5 Le serveur . . . 57
5.2.1.6 Le client . . . 57
5.2.1.7 Lancement . . . 58
5.2.1.8 Schéma récapitulatif . . . 58
5.3 Intégration du mécanisme de collaboration en temps réel . . . 59
5.3.1 Communication mobile/clone . . . 59
5.3.2 Communication clone/super clone . . . 60
5.3.3 Les méthodes invoquées à distance . . . 61
5.3.4 Communication clone/clone . . . 61
5.3.5 Schéma récapitulatif . . . 62

5.4 Interfaces graphiques . . . . . 63
5.4.1 Interface du Mobile . . . . . 63
5.4.2 Interface du Clone . . . . . 63
5.5 Conclusion . . . . . 64

**Conclusion et perspectives** **65**

**Bibliographie** **67**

# Liste des figures


1.1 Métaphore du nuage. . . . . 6
1.2 La pile de Cloud Computing. . . . . 8

2.1 Architecture du Cloud Computing Mobile. . . . . 13
2.2 L'architecture orientée services du Cloud Computing. . . . . 14

3.1 Schéma d'un architecture pair à pair de première génération . . . . . 22
3.2 Schéma d'une architecture pair à pair décentralisé non-structurée . . . . . 22
3.3 Architecture du système Chord . . . . . 23
3.4 Schéma d'une architecture pair à pair de troisième génération . . . . . 24
3.5 Les dimensions du Cloud Computing . . . . . 24
3.6 Architecture en couches des P2P CS. . . . . 27
3.7 Le découpage d'un nuage P2P en tranches. . . . . 28

4.1 Description des étapes de création d'un clone. . . . . 36
4.2 Exemple illustratif d'une collaboration en temps réel entre les acteurs de C2C. . . . . 38
4.3 Etat de la plateforme. . . . . 43
4.4 Arbre de la tâche T. . . . . 44
4.5 Scénario de déroulement des algorithmes de mécanisme de collaboration en temps reel sur l'exemple précédant. . . . . 46
4.6 Exemple illustratif d'un scénario de partage de fichier. . . . . 48
4.7 Scénario de déroulement des algorithmes de mécanisme de partage de fichier. 51
4.8 gestion de mobilité. . . . . 52

5.1 Schéma récapitulatif du RMI. . . . . 58
5.2 Communication mobile/clone. . . . . 60
5.3 Communication clone/super clone. . . . . 60

5.4 Communication clone/clone. . . . . 61
5.5 Schéma récapitulatif. . . . . 62
5.6 Interface du Mobile. . . . . 63
5.7 Interface du Clone . . . . . 64

# Liste des tableaux

3.1 Comparaison des Clouds IaaS, du Volunteer computing et Clouds P2P . . . 26

3.2 Vue globale sur l'ensemble des méthodes de débarquement du calcul. . . . 32

# Liste des abreviations

| | |
|---|---|
| **P2P** | **P**eer to **P**eer |
| **ASP** | **A**pplication **S**ervice **P**rovider |
| **IBM** | **I**nternational **B**usiness **M**achines |
| **NIST** | **N**ational **I**nstitute of **S**tandards and **T**echnology |
| **IP** | **I**nternet **P**rorocol |
| **IaaS** | **I**nfrastructure as a **S**ervice |
| **PaaS** | **P**latform as a **S**ervice |
| **SaaS** | **S**oftware as a **S**ervice |
| **OS** | **O**perating **S**ystem |
| **HPC** | **H**igh **P**erformance **C**omputing |
| **AAA** | **A**uthentication **A**uthorization **A**ccounting |
| **HA** | **H**ome **A**gent |
| **SOA** | **S**ervice **O**riented **A**rchitecture |
| **LAN** | **L**ocal **A**rea **N**etwork |
| **DHT** | **D**istributed **H**ash **T**able |
| **QOS** | **Q**uality **O**f **S**ervice |
| **CPU** | **C**entrals **P**rocessing **U**nit |
| **VCPU** | **V**irtual **C**entrals **P**rocessing **U**nit |
| **RAM** | **R**andom **A**ccess **M**emory |
| **VRAM** | **V**irtual **R**andom **A**ccess **M**emory |
| **RMI** | **R**emote **M**ethode **I**nvocation |
| **RMP** | **R**emote **M**ethode **P**rotocol |
| **API** | **A**pplication **P**rogramming **I**nterface |

| | |
|---|---|
| **GSM** | **G**lobal **S**ystem for **M**obile |
| **GPS** | **G**lobal **P**ositioning **S**ystem |
| **JNDI** | **J**ava **N**aming and **D**irectory **I**nterface |
| **JDK** | **J**ava **D**evelopement **K**it |
| **JVM** | **J**ava **V**irtuel **M**achine |

# Introduction générale

Une grande partie de la littérature relative au Cloud mobile implique des appareils mobiles agissant comme des clients légers et tirent partie des centres de données pour des fins de calcul. Dans la pratique, les applications basées sur le Cloud mobile entraînent le partitionnement statique des applications où l'appareil mobile est chargé d'exécuter le client face à des tâches telles que l'interface utilisateur, tandis qu'un centre de données héberge le serveur, le middleware et les volumes des bases de données. Il n'y a que quelques années, les smartphones et les tablettes sont équipés d'un traitement embarqué, d'une mémoire et des ressources de stockage à semi-conducteurs équivalents aux ordinateurs portables avec une spécification raisonnable. De telles améliorations dans le Cloud Mobile sont appelées à se poursuivre, en témoigne la récente annonce par Qualcomm, pour offrir un processeur double coeur de 1,5 GHz pour les dispositifs mobiles et ceux de Apple de passer aux processeurs multi-cœurs dans les iPhones. Contrairement au calcul traditionnel, ces nouveaux ordinateurs est encapsulée dans un appareil de poche, ce système de poche que les utilisateurs portent avec eux partout et tout le temps.

Les environnements typiques de Cloud Computing impliquent le déploiement d'une infrastructure de serveurs à grande échelle de façon propriétaire aux fournisseurs de services tels que Google, IBM, Amazon, Apple et Facebook. Ces fournisseurs de services gèrent un réseau mondial de centres de données, qui combinent la puissance de calcul de plusieurs dizaines de milliers de serveurs de produit de base. Les plates-formes de Cloud Computing sont arrivées à un stade de développement précoce, comme les centres de données,en grande partie isolées, utilisées pour la mise à disposition d'applications spécifiques ou des ressources d'infrastructure absolues. La vision de Cloud Computing devrait être

plus souple que ces modèles, où les utilisateurs en général, y compris les mobiles, peuvent participer dans le Cloud autant que consommateurs et fournisseurs de ressources. Si le Cloud est pour le calcul ce qu'est l'Internet pour les données, la possibilité pour les ordinateurs mobiles pour participer à la prestation de services doit être étudiée.

Il ya une définition plus floue d'un appareil mobile compte tenu de la prolifération des smartphones et des tablettes accessibles sur Internet, les livres électroniques et d'autres appareils. Une opportunité potentiellement importante existe afin d'éxploiter la puissance de ces dispositifs que ce soit pour le traitement simultané, le stockage fédéré ou pour la fourniture des applications web mobiles personnalisées hébergées directement sur une application mobiles . Le passage de ces dernières tirent parti simplement des applications comme un client léger, directement à travers les services d'hébergement, cela pourrait conduire à une grande variété de nouvelles applications, de nouveaux systèmes et des architectures innovantes.

Des modèles et infrastructures pour le Cloud Mobiles sont proposées comme les systèmes P2P. Nous pourrions envisager un réseau P2P sans fil à l'échelle mondiale pour les dispositifs mobiles, et lui faire exécuter sur lui des applications qui tirent avantage de l'énorme quantité de données rassemblées, sensibles et stockées par ces dispositifs, aussi bien par la nature distribuée et sociale de ce système. En réalité, les applications mobiles actuelles sont restreintes par la durée de vie de leur batterie , ce qui les rend très compliquées pour que de tels systèmes existent. Ces systèmes présentent une surcharge importante en termes de calcul et en particulier, de communication.

Vu l'augmentation de l'utilisation des dispositifs mobiles tels que les téléphones portables et les tablettes PC d'un côté et leurs limites en terme de puissance de calcul, stockage et applications développées d'un autre côté, un Cloud Mobile doit garantir des communications fiables et sans coupure à un utilisateur mobile tandis qu'il se déplace ainsi que les interactions en temps réel en essayant de réduire la consommation d'énergie des dispositifs mobiles.

La motivation de ce mémoire est triple. D'abord, il explore les forces qui ont abouti

à l'émergence du Cloud Computing. Deuxièmement, les implications nœuds d'un serveur mobile dans un environnement de Cloud Computing sont considérées. Enfin, ce mémoire porte sur la mise en œuvre d'une architecture de Cloud mobile pour l'exécution des calculs distribués. Notre objectif est de réaliser une architecture pour le Cloud Mobile en mode P2P qui a pour but le débarquement du calcul des dispositifs mobiles vers le Coud, cette dernière permet des services tels que la collaboration en temps réel entre les utilisateurs des dispositifs mobiles, en plus elle contient un mécanisme de partage de fichiers. D'ailleurs, le haut débit du réseau P2P fourni par le Cloud a typiquement l'excellente disponibilité et certainement n'a pas des problèmes de capacité de réseau. En présence d'un tel système, à chaque fois qu'un utilisateur doit exécuter un travail lourd sur son dispositif, il pourrait déléguer le travail, mais peut également le distribuer. Afin de surmonter les problèmes liés à la mobilité des dispositifs mobiles lors de leurs communications avec leurs délégués dans le Cloud nous présentons un mécanisme qui garantit la bonne gestion de mobilité.

Afin de répondre à la problématique posée pour ce travail de master , nous organisons ce document en six chapitres. En premiers lieu, nous présentons les concepts de base du Cloud Computing. Ensuite nous entamons le chapitre 2 sur le principe du Cloud Computing mobile. Le chapitre trois porte sur le déploiement d'une architecture de Cloud sur un réseau P2P. Le chapitre quatre présente quelques travaux antérieurs sur les approches existantes pour le débarquement du calcul ainsi que la gestion de mobilité. Le chapitre cinq décrit la solution proposée. Le chapitre six présente les détails sur l'implémentation de notre approche. Et enfin, nous clôturons ce mémoire par une conclusion et quelques orientations futures.

Chapitre 1

# Les Concepts de base du Cloud Computing

## 1.1 Introduction

Il y a quelques années, on nommait ASP (Application Service Provider) le fait de proposer une application sous forme de service. En remontant un peu plus loin en arrière, dans les années 60, IBM proposait déjà l'informatique à la demande . Les années 80 furent aussi le début des concepts de virtualisation. Tous ces concepts ont amené, petit à petit, à inventer une nouvelle manière de proposer l'informatique comme un service.

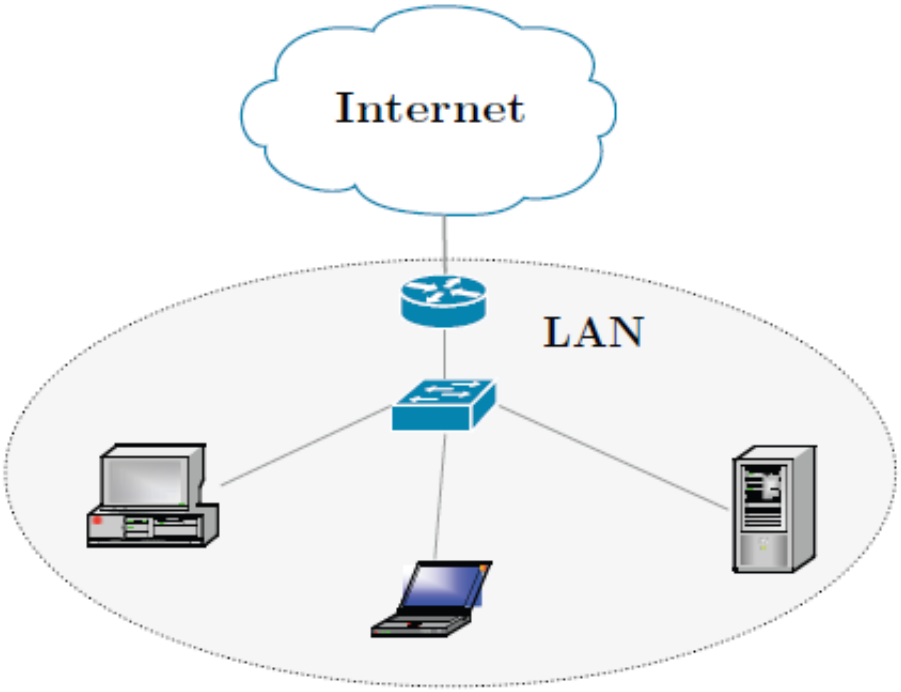


Figure 1.1 – Métaphore du nuage.

## 1.2 Qu'est-ce-que le Cloud Computing ?

Le Cloud Computing, ou « informatique dans les nuages », est un « nouveau » modèle informatique qui consiste à proposer des services informatiques sous forme de services à la demande, accessibles de n'importe où, n'importe quand et par n'importe qui. Cette approche n'est pas tout à fait nouvelle (modèle ASP). La réelle nouveauté réside dans son approche systématique.

Nous tentons toutefois, à travers de ce mémoire, de donner toutes les clés pour comprendre le Cloud Computing du fait que de nombreuses définitions ont été proposées pour le Cloud Computing, en se concentrant sur les différents aspects qui caractérisent le paradigme. Nous citons plusieurs approches complémentaires fournies par Gartner [15], Wikipédia [2], le 451group [1] et IBM[3] :

- Le Gartner définit le Cloud Computing comme « des technologies de l'information possédant des capacités de traitement (serveurs) massivement évolutives qui sont fournies en tant que service à l'aide des technologies Internet, à de multiples clients externes ».
- Pour Wikipédia, il s'agit d' « un concept de déportation sur des serveurs distants des traitements informatiques traditionnellement localisés sur le poste client ».
- Pour le 451group : « Le Cloud est le système d'information appréhendé comme un service, délivré par des ressources SI(Système d'Information) ».
- Pour IBM il représente : «Une plateforme de type Cloud provisionne, configure, reconfigure et déprovisionne des serveurs à la demande. Les applications dans le Cloud utilisent de grands centres de données, et des serveurs performants qui hébergent des applications web ainsi que des services web».

## 1.3 Offres du Cloud Computing

Le Cloud Computing offre un modèle d'accès à la demande à des ressources informatiques partagées et configurables (des réseaux, des serveurs, du stockage, des applications et des services) qui peuvent être rapidement provisionnées et mises à disposition avec un effort minimum de gestion et d'interactions avec le fournisseur. La référence aux Clouds évoque une dématérialisation de l'infrastructure informatique. L'idée est de proposer une capacité de traitement informatique sans se soucier des ressources physiques nécessaires à

son maintien : ni à sa localisation, ni à sa puissance, [15] etc. L'offre de Cloud Computing se décline en trois modèles distincts :

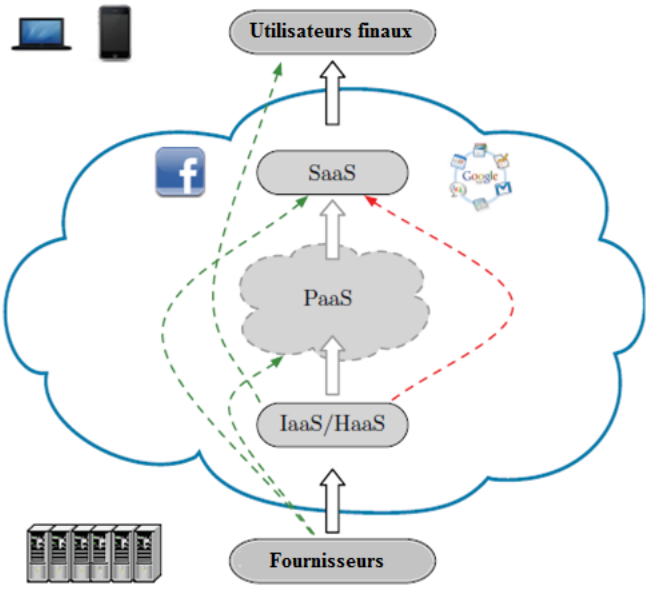


FIGURE 1.2 – La pile de Cloud Computing.

- **Infrastructure as a Service (IAAS)** : C'est le niveau le plus bas de la pile de Cloud Computing, fournissant aux utilisateurs un accès à la demande aux ressources virtualisées, qu'ils peuvent configurer entièrement. Les utilisateurs peuvent louer des ressources de calcul, de stockage ou de mise en réseau pour des périodes prédéfinis. Ils ont accès aux services de gestion d'infrastructure tels que le déploiement automatique de ressource et l'échelle dynamique du nombre de noeuds loués.
- **Plateforme as a Service (PAAS)** : Les fournisseurs de Cloud offrent une couche d'abstraction additionnelle établie sur des fonctionnalités d'IaaS. Les services de PaaS consistent en un environnement intégré à un niveau élevé qui permet à des utilisateurs d'établir, d'examiner et d'exécuter leurs propres applications. Il fournit un Système d'Exploitation (OS) ainsi que tous les logiciels serveurs associés : serveur d'application, bases de données, d'intégration, les runtimes. Il s'agit donc d'une plateforme logicielle complète, sur laquelle il est possible de développer ou migrer ses propres applications. Ces systèmes s'appuient aussi sur des environnements standardisés et multitechnologies (J2EE, .NET, PHP). Ces systèmes fournissent de plus un environnement complet d'exploitation en fournissant des

solutions d'équilibrage de charge, de surveillance et de supervision.

- **Software as a Service (SAAS)** : Au niveau de la hiérarchie le plus élevé, SaaS fournit des logiciels spécialisés offert sur le Cloud, accessible aux clients seulement par l'Internet. De telles applications sont habituellement établies sur le Cloud de PaaS ou d'IaaS. Les principes capitaux des applications SaaS sont qu'elles peuvent se fonder sur la puissance des services de Cloud les plus bas pour réaliser une meilleure disponibilité et d'exécution en influençant des dispositifs comme le traitement distribué ou l'échelle automatique.

La solution est intégralement prise en charge par le fournisseur de la solution : la création et la maintenance du logiciel, son hébergement, l'infrastructure matérielle, l'OS, la maintenance des logiciels "middleware" (serveur d'applications, base de données). Le Cloud Computing comprend donc plusieurs modèles, permettant de s'adapter à chaque problématique, à chaque entreprise :

- hébergement d'une solution propriétaire : l'utilisateur y installe ses logiciels middlewares, ses applications et ses données,
- hébergement d'une application développée par l'utilisateur sur des standards de l'industrie informatique,
- utilisation d'une solution logicielle complète proposée par le fournisseur d'offre Cloud.

## 1.4 Les modèles de déploiements du Cloud Computing

Les entreprises, ainsi que les organisations scientifiques, ont des besoins différents en ce qui concerne le type, la disponibilité ou la sécurité des services qu'ils emploient. Pour y remédier, plusieurs modèles de déploiements du Cloud ont été proposés, chacun d'eux présentent des points spécifiques [15].

### 1.4.1 Cloud publique

Le Cloud publique offre ses services à n'importe quel client qui accède à l'Internet. En règle générale, les services sont gérés par des fournisseurs de l'extérieur du site, qui contrôlent également le réseau et les paramètres de sécurité, tandis que l'utilisateur peut obtenir à la demande des ressources facturés en se basant sur l'utilisation par payement.

### 1.4.2 Cloud privé

Dans ce cas, l'infrastructure est exclusivement utilisée par une seule organisation. L'avantage du Cloud privé est qu'il offre le plus haut degré de sécurité et de fiabilité. Sur le plan négatif, avoir un accès à un Cloud privé implique d'avoir un accès complet à l'infrastructure physique et aussi à la construction et à la gestion dans le Cloud.

### 1.4.3 Cloud communautaire

Ce type de Cloud est dédié à une communauté spécifique qui intègre plusieurs organisations ayants des intérêts communs. L'infrastructure de Cloud est partagée par des organisations. Exemples de Cloud d'un domaine spécifiques : les Clouds scientifiques [16]et les Clouds du calcul haute performance HPC (High Performance Computing) . Le premier exemple se rapporte à un Cloud conçu pour fournir des moyens expérimentaux à des projets scientifiques et éducatifs. Les Clouds HPC offrent un soutien adéquat à la communauté HPC.

### 1.4.4 Cloud hybride

Le Cloud hybride vise à répondre aux limites des autres modèles de déploiements cités ci-dessus. C'est une combinaison entre les deux Clouds privés et publiques, le modèle hybride préserve de garantir la sécurité du premier modèle, l'améliorer avec la possibilité à l'échelle dynamique en louant des ressources des fournisseurs du Cloud publique, lorsque les besoins de calcul surcharge les ressources locales. Atteindre un tel degré de fexibilité est cependant une tâche diffcile, car de nombreuses interfaces ou des services de Cloud sont incompatibles et ne peuvent pas être accessibles via les outils du même client.

## 1.5 Conclusion

L'informatique dans les nuages est un paradigme qui offre un nouveau modèle de distribution et de consommation de ressources informatiques à grande échelle. Les technologies associées à cette discipline permettent aux propriétaires de grands centres de traitement de données de louer les ressources inutilisées dont ils disposent, et de ce fait d'augmenter la rentabilité de leur investissement matériel. Les clients de Cloud bénéficient

également de ce modèle de distribution de ressources, car il leur permet d'assouplir leur mode d'investissement en ressources informatiques, par exemple en ajustant la capacité de traitement de leur infrastructure informatique au fur et à mesure que leurs besoins évoluent. Le chapitre suivant, décrira l'émergence du Cloud Computing pour les applications mobiles.

Chapitre 2

# *Cloud Coumputing Mobile*

## 2.1 Introduction

Avec le fort développement des terminaux mobiles (Smartphones et tablettes) et leur adoption progressive dans le monde professionnel, le nomadisme en entreprise devient un enjeu majeur, ouvrant des perspectives nouvelles en termes d'organisation du travail. Que ce soit pour gagner en productivité, réactivité, qualité ou souplesse dans l'organisation de travail, les cas d'usage des terminaux mobiles sont multiples : équipement des commerciaux et autres populations nomades, travail collaboratif à distance, consultation du système d'information en situation nomade, animation de réunion, visioconférence, etc.

Vu les limites des dispositifs mobiles en terme de puissance de calcul, stockage et traitement de données le Cloud Computing s'intègre dans le traitement Mobile afin de répondre aux demandes des utilisateurs en surmontant les obstacles liées à l'environnement discutés dans le Cloud Computing.

## 2.2 Qu'est-ce-que le Cloud Computing Mobile ?

De nombreuses définitions du Cloud Computing Mobile ont été mises en œuvre afin de permettre une meilleure compréhension de ce paradigme, Nous vous proposons plusieurs approches complémentaires fournies par Techopedia [4], Wikipédia et IBM :

- Pour Techopedia le Cloud Computing Mobile représente «un modèle dans lequel les applications mobiles sont construites, réalisée et hébergée en utilisant la technologie de Cloud Computing».
- Pour Wikipédia le Cloud Computing Mobile «est la combinaison de Cloud Computing et des réseaux mobiles pour apporter des avantages pour les utilisateurs mobiles, opérateurs de réseaux, ainsi que les fournisseurs de Cloud»
- Pour IBM le Cloud Computing Mobile représente «des services dont le stockage

des données et/ou les opérations de calcul et traitement sont effectués sur un serveur et non sur un terminal de consultation».

## 2.3 Architecture du Cloud Computing Mobile :

L'architecture générale du Cloud Computing Mobile peut être illustrée sur la figure 2.1 [17] :

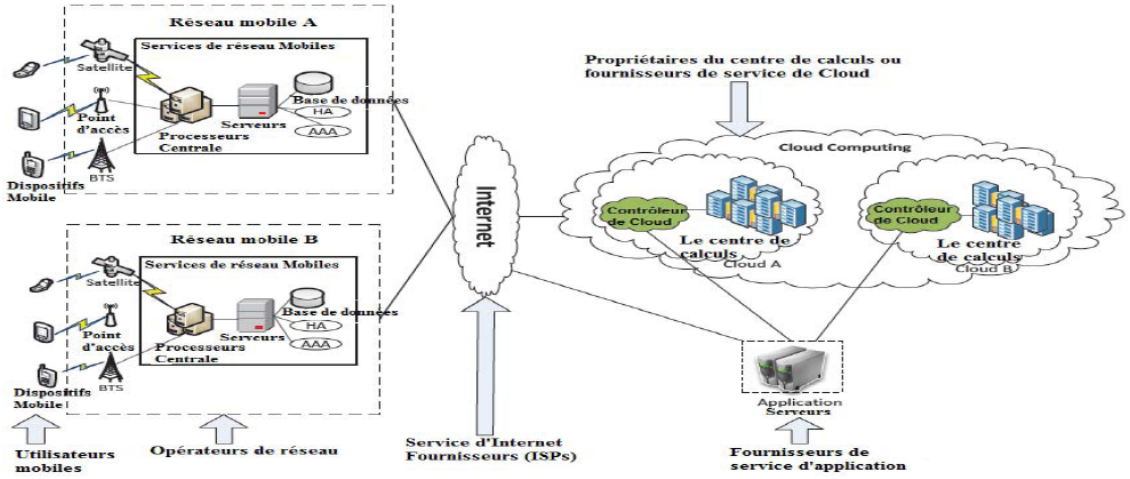


FIGURE 2.1 – Architecture du Cloud Computing Mobile.

Dans cette figure, les dispositifs mobiles sont connectés à des réseaux mobiles via les stations de bases qui permettent d'établir et de contrôler les connexions et les interfaces fonctionnelles entre les réseaux et les appareils mobiles. Les demandes des utilisateurs mobiles et les informations à l'instar de l'ID et l'emplacement sont transmises aux processeurs centraux qui sont reliés à des serveurs fournissant des services de téléphonie mobile. Les opérateurs des réseaux mobiles peuvent offrir des services tels qu'AAA (Authentication Authorization Accounting) à base de l'agent local HA (Home Agent) et les données des abonné stockées dans les bases de données. Après cela, les demandes des abonnés sont délivrées à un Cloud à travers l'internet.

Dans un Cloud, il y a toujours des contrôleurs du Cloud qui traitent les demandes de couvertures des utilisateurs mobiles avec les services du Cloud Computing correspondant. Ces services sont développés avec les concepts de l'informatique, la virtualisation et

l'architecture orientée services SOA (Service Oriented Architecture).

Afin de permettre une meilleure compréhension du modèle SOA nous allons présenter la figure 2.2 [18] :

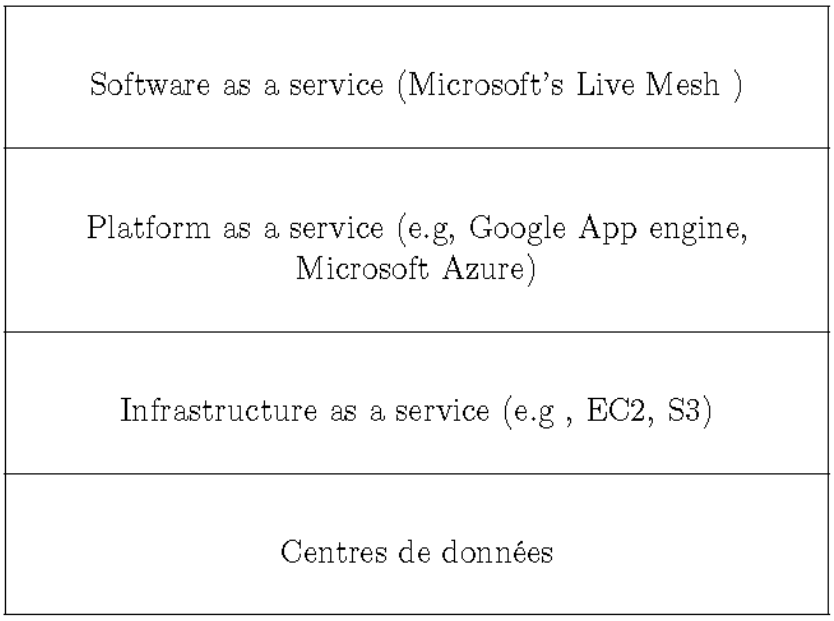


FIGURE 2.2 – L'architecture orientée services du Cloud Computing.

Sur la figure 2.2, les services du Cloud Computing sont généralement classés en fonction du concept de couche , on constate qu'il existe quatre couches dans ce paradigme les trois couches supérieures sont déjà présentées dans la section 1.3 . Par contre la couche centres de données permet d'offrir les possibilités matérielle et infrastructure au Cloud en plus elle contient un certain nombre de serveurs qui sont reliées à des réseaux de grande vitesse afin de fournir les différents services aux utilisateurs. Généralement les centres de traitement de données se situent dans des endroits avec une grande densité populaire.

## 2.4 Avantages du Cloud Computing Mobile

Le Cloud Computing Mobile est conçu pour être une solution prometteuse pour le traitement mobile, dans ce qui suit nous décrivons comment le Cloud peut être utilisé pour surmonter les obstacles du traitement mobile, en soulignant les avantages du Cloud Computing Mobile.

### 2.4.1 Extension de la durée de vie de la batterie

La batterie est l'une des principales préoccupations des utilisateurs des appareils mobiles. Pour cela plusieurs solutions ont été proposées afin d'améliorer les performances du processeur [19, 20] et gérer le disque et l'écran d'une manière intelligente [21, 22]afin de réduire la consommation d'énergie. Cependant ces solutions exigent des changements dans la structure des appareils mobiles ce qui n'est pas possible pour tous les dispositifs mobiles. La technique de calcul du déchargement a été proposée avec l'objectif de migrer les calculs importants et complexes des machines limités à l'instar des dispositifs mobiles (smartphone [5], Android [7], etc.) vers des machines ingénieuses autrement dit les serveurs au niveau du Cloud. Cela évite de prendre une application avec un long temps d'exécution sur un dispositif mobile qui se traduit par une grande quantité de consommation d'énergie.

### 2.4.2 Amélioration de la capacité de stockage et la puissance de calcul :

Le Cloud Computing Mobile est conçus pour permettre aux utilisateurs mobiles de stocker et accéder aux données dans le Cloud, le premiers exemple est AMAZON simple Storage devices (AMAZON S3) [9] qui supporte le service de stockage de fichiers, le deuxième exemple est le iCloud conçu par APPLE qui permet aussi à ses clients de stocker leurs fichiers [23]. Avec le Cloud, les utilisateurs peuvent bénéficier d'une quantité considérable d'espace sur leurs dispositifs mobiles vu que certains de leurs fichiers sont envoyés et traités au niveau du Cloud.

Facebook représente un exemple typique de l'utilisation du Cloud. Le Cloud Computing Mobile permet aussi de réduire le coût de fonctionnement des applications du calcul intensifs vu la migration du calcul des dispositifs mobiles vers le Cloud, ce dernier fait toujours appel à des machines virtuelles VMs [14] et des cloudlet [24].

### 2.4.3 Amélioration de la fiabilité

Le stockage de données ou l'exécution des applications sur Cloud est un moyen efficace pour améliorer la fiabilité, car les données et les applications sont stockées et sauvegardées sur plusieurs ordinateurs ce qui permet de réduire le risque de perte de

données et des applications sur les dispositifs mobiles .En outre le Cloud Computing Mobile est conçu pour être un modèle de sécurité de données pour les fournisseurs des services et aux utilisateurs. Par exemple le Cloud peut être utilisé pour protéger les droit d'auteur des contenus numérique [25]. En effet le Cloud peut offrir à distance pour les utilisateurs mobiles avec des services de sécurité, des antivirus, des détecteurs de code malveillant et l'authentification ce qui implique d'avoir un usage efficace et fiable.

## 2.5 Application du Cloud Computing Mobile :

Les applications mobiles gagnent de plus en plus une place dans le marché mobile en plus elles bénéficient des avantages du Cloud Computing Mobile. Dans cette section nous présentons quelques applications du Cloud Computing Mobile .

### 2.5.1 Commerce mobile

Le commerce mobile (m-commerce) est un modèle d'affaire pour le commerce en utilisant des appareils mobiles. Les applications du m-commerce remplient généralement certaines tâches qui requièrent la mobilité, par exemple les transactions de paiements mobiles, la messagerie mobile et la billetterie mobile en plus elles doivent faire face à des défis différents à l'instar de la bande passante ainsi que la sécurité etc. par conséquent, les applications du m-commerce sont intégrées dans un environnement du Cloud Computing afin d'aborder ces questions.[26] ont proposés une connexion de 3G e-commerce, une plateforme de e-commerce basée sur le Cloud Computing. Ce paradigme combine entre les avantages des réseaux 3G et le Cloud Computing dont l'objectif est d'augmenter la vitesse de traitement de données et le niveau de sécurité basé sur l'ICP ( Infrastructure à Clés Publiques) .Dans [27], une plateforme qui utilise la technologie du Cloud Computing afin de renforcer la sécurité pour les utilisateurs et améliorer la satisfaction des clients.

### 2.5.2 Mobile Learning

Mobile Learning (m-learning) est conçu à base de l'apprentissage électronique (e-learning) et la mobilité. Cependant, les applications traditionnellles du m-learning ont des limitations en termes du coût élevé des dispositifs mobiles, les réseaux, le taux de transmission faible et les ressources éducatives limitées [28, 29, 30]. Les applications Basées

sur le Cloud et le m-learning sont proposées pour résoudre ces limitations. [31] présente les avantages de la combinaison de m-learning et le Cloud Computing pour améliorer la qualité de communication entre les étudiants et les enseignants. Grâce à un site web construit sur Google Apps Engine, les élèves communiquent avec leurs enseignants à tout moment. En outre, un contexte m-learning basé sur la plateforme IMERA [32] montre qu'un système basé sur le Cloud m-learning aide les apprenants à accéder à des ressources d'apprentissage à distance.

Dans [33], un outil éducatif est développé sur la base du Cloud Computing pour créer des cours basés sur le traitement d'image / de vidéo via les téléphones mobiles ce qui implique que les apprenants peuvent comprendre et comparer les différents algorithmes utilisés dans les applications mobiles.

### 2.5.3 Services médicaux mobiles

Le but de l'application du Cloud Computing mobile dans les applications médicales est de minimiser la limitation du traitement médical traditionnel. Les services médicaux mobiles (m-healthcare) permettent aux utilisateurs mobiles d'accéder aux ressources (par exemple un dossier d'un patient) facilement et rapidement. En outre le m-healthcare offre aux hôpitaux et aux organisations de santé une variété de services à la demande sur le Cloud plutôt que de posséder des applications autonomes sur des serveurs locaux. [34] présente cinq principales application du m-healthcare :

- **Les services de surveillance de la santé** : qui permettent aux patients d'être suivis à tout moment et n'importe où grâce à des communications sans fil à large bande.
- **Un système intelligent de gestion des urgences** : qui permet de gérer et de coordonner la flotte des véhicules d'urgence efficacement et en même temps lors de la réception des appels d'accidents ou d'incidents.
- **Des dispositifs mobiles santé/conscient** :qui permettent de détecter le taux d'impulsion, la pression artérielle et le niveau d'alcool pour avertir les soins de santé d'un système d'urgence.
- **Un accès à l'information** : qui permet aux patients et aux fournisseurs de soins de santé d'accéder aux informations médicales courantes et anciennes.
- **Une gestion d'incitation** : est utilisée pour permettre aux patients de payer

les dépenses de soin et de gérer les autres frais.

### 2.5.4 Mobile Gaming

Les jeux mobiles (m-game) est un marché potentiel générant des revenus pour les fournisseurs de services. Le m-game peut décharger des jeux nécessitant des ressources informatiques importantes (par exemple le rendu graphique) aux serveurs du Cloud, ce qui permet aux joueurs d'interagir avec l'interface écran de leurs dispositifs mobiles. Les auteurs dans [35] démontrent que le déchargement (code multimédia) peut économiser de l'énergie pour les appareils mobiles. [36] proposent MAUI (Memory Arithmetic Unit and Interface) ce paradigme permet non seulement d'économiser de l'énergie pour les appareils mobiles mais aussi d'améliorer les performances des applications mobiles (taux de rafraîchissement du jeu augmente de 6 à 13 images par seconde). [37] présentent un m-jeu utilisant une technique d'adaptation du rendu pour ajuster dynamiquement le jeu et de configurer les paramètres en fonction des contraintes de communication et les exigences des joueurs. La technique d'adaptation du rendu repose principalement sur l'idée de réduire le nombre d'objets dans la liste d'affichage car tout objets crées par le moteur de jeu sont nécessaires pour jouer le jeu. L'objectif donc est de maximiser l'expérience de l'utilisateur en tenant compte des communications et le coût de calcul.

### 2.5.5 Autres applications pratiques

Le Cloud devient un outil utile pour aider les utilisateurs à partager des photos et des clips vidéo de manière efficace et de marquer leurs amis dans les réseaux sociaux populaires comme Twitter et Facebook. MeLog [38] est une application du Cloud Computing Mobile qui permet aux utilisateurs mobiles de partager une expérience en temps réel (par exemple, les voyages, le shopping, et l'événement) sur les Clouds à travers un blog. Les utilisateurs mobiles (par exemple, les voyageurs) sont pris en charge par les services du Cloud pour guider leur voyage en montrant des cartes, l'enregistrement de l'itinéraire, et en stockant des images et des vidéos.

[39] introduit un service mobile de localisation qui permet aux utilisateurs de capturer un clip vidéo des bâtiments environnants l'algorithme d'appariement qui fonctionne sur un Cloud peut utiliser une grande quantité d'informations afin de rechercher l'em-

placement de ces bâtiments. En outre, OneHourTranslation [6] fournit une traduction en ligne grâce à un service fonctionnant sur le Cloud de service Amazon Web. OneHour-Translation aide les utilisateurs mobiles en particulier les visiteurs étrangers à recevoir des informations traduites dans leur langue par le biais de leurs appareils mobiles. Ainsi, nous pouvons reconnaître que le Cloud Computing Mobile est probablement une tendance technologique qui prévaudra avec de nombreuses applications dans un avenir proche.

## 2.6 Conclusion

Le Cloud Computing Mobile fournit à l'utilisateur mobile une fonctionnalité transparente et riche, quel que soit les limites des appareils mobiles. Bien qu'encore à ses débuts, le Cloud Computing Mobile pourrait devenir le modèle dominant des applications mobiles à l'avenir. Apres avoir présenté les points importants du contexte de notre thème nous allons entamer la présentation des Clouds P2P dans le chapitre suivant.

Chapitre 3

# *Etat de l'art sur le Cloud Computing*

## 3.1 Introduction

Le Cloud Computing est devenu populaire dans le domaine de la recherche ainsi que celui de l'industrie. Alors que les systèmes Cloud sont généralement hébergés dans de grands centres de données et sont gérés de manière centralisé, d'autres types d'architectures du Cloud peuvent être imaginés, il s'agit du Cloud P2P.

Les dispositifs mobiles et les applications mobiles continuent à avoir une croissance extraordinaire. Cependant, les limites en terme de puissance de calcul et la durée de vie de la pile les rendent difficile pour que les utilisateurs qui exploitent entièrement leurs mobiles. Pour cela plusieurs recherches ont été faites et plusieurs approches ont été proposées mais l'objectif reste commun, surmonter les limitations des dispositifs mobiles en mettant en oeuvre des méthodes de débarquement de calcul au niveau du Cloud.

Dans ce chapitre nous décrivons une architecture décentralisé, il s'agit du Cloud P2P, mais commençons d'abord par une présentation des architectures de système P2P, ensuite nous étudions les méthodes de débarquement de calcul existantes.

## 3.2 Cloud computing P2P

### 3.2.1 Systèmes P2P

Les systèmes P2P se sont rendus célèbres vis-à-vis du grand public grâce aux systèmes de partage de fichiers sur l'Internet comme Gnutella [10] , FreeNet [11], mais aussi grâce à des systèmes qui utilisent les ressources inutilisées des ordinateurs comme Seti@home [12], Genome@home [13]. Les systèmes pair à pair sont motivés par :

- Un besoin croissant de ressources informatiques (espace disque, temps de calcul ...).
- Un besoin de faire du calcul massivement parallèle.

- Le profit qu'ils apportent aux entreprises (parc informatique souvent sous-utilisé).
- Vulnérabilité du modèle « client/serveur ».
- Travaux de collaboration entre personnes.

#### 3.2.1.1 Définition

Le terme P2P se rapporte à une classe de systèmes et d'applications qui utilisent des ressources distribuées afin d'accomplir une fonction critique de manière décentralisée [40]. Appelés aussi, systèmes d'égal-à-égal, tous les participants jouent un rôle identique. Par exemple dans le cas de l'échange de fichiers sur des réseaux pair à pair, les ordinateurs qui y participent sont tour à tour demandeur et fournisseur, client et serveur, ce sont des pairs.

#### 3.2.1.2 Architectures

Depuis leur émergence à la fin des années 90, les systèmes pair à pair ont beaucoup évolués et se sont diversifiés dans leur architecture. On peut classifier les réseaux pair à pair en trois générations :

– Première génération : architecture centralisée.
– Deuxième génération : architecture décentralisée.
– Troisième génération : architecture hybride.

**a. Première génération**

La première génération de réseaux pair à pair est l'architecture centralisée qui est très similaire à l'architecture client/serveur. Dans ce modèle, un serveur central stable indexe tous les pairs du système et stocke les informations sur leur contenu. Lors de la réception d'une requête d'un pair, le serveur central choisit un autre pair dans son répertoire qui assortit la demande. Ensuite, des communications sont exécutées directement entre les deux pairs. L'exemple de cette génération est Seti@home . La figure 3.1 illustre le schéma d'une architecture pair à pair de première génération.

**b. Deuxième génération**

La deuxième génération de réseaux P2P correspond à des architectures décentralisées qui ne s'appuient sur aucun serveur. Chaque pair a exactement les mêmes possibilités et

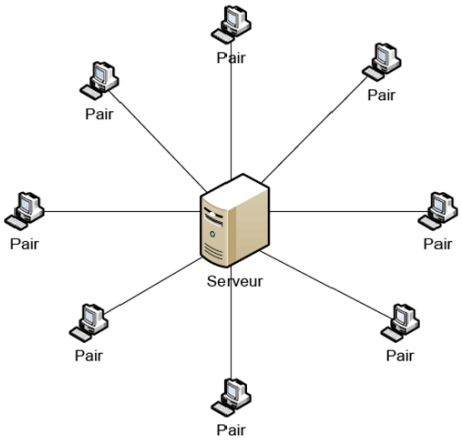


FIGURE 3.1 – Schéma d'un architecture pair à pair de première génération

peut se comporter en tant que client ou serveur en même temps. Cette génération peut être divisée en deux classes : non-structurée et structurée. Dans la première classe, la topologie logique est souvent aléatoire. Chaque pair indexe ses propres ressources partagées. Une requête d'un pair inonde (broadcast) directement les pairs voisins, qui à leur tour inondent leurs voisins. Cette action est effectuée jusqu'à ce que la demande soit obtenu la réponse ou qu'un nombre maximum d'étapes d'inondation soit atteint. On peut trouver dans cette classe le Gnutella 0.4 . La figure 3.2 illustre le schéma d'une architecture pair à pair décentralisée non-structurée

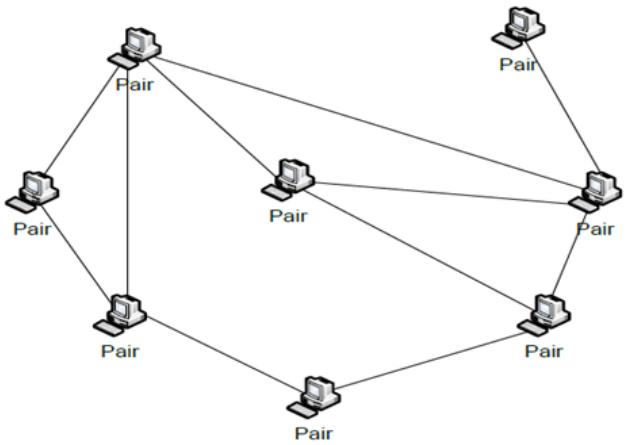


FIGURE 3.2 – Schéma d'une architecture pair à pair décentralisé non-structurée

Dans la deuxième classe, la topologie logique est structurée comme anneau (Chord [41]), d-dimension (CAN [42]) etc. Ces topologies structurées sont souvent construites en utilisant des tables de hachage distribuées (DHT). Chaque pair indexe une partie des

ressources partagées du réseau et possède une partie de la table de hachage du système. La requête est transmise selon la topologie structurée et est assuré du succès après un nombre déterministe d'étapes sous des conditions idéales. La figure 3.3 montre l'architecture du système Chord. La deuxième classe est plus robuste que la première classe et propose l'anonymat des pairs. Elle présente des aptitudes au changement d'échelle et propose la réduction du temps de recherche grâce à la table de hachage. Cependant, elle nécessite un protocole assez lourd pour la maintenance de la structure de topologie.

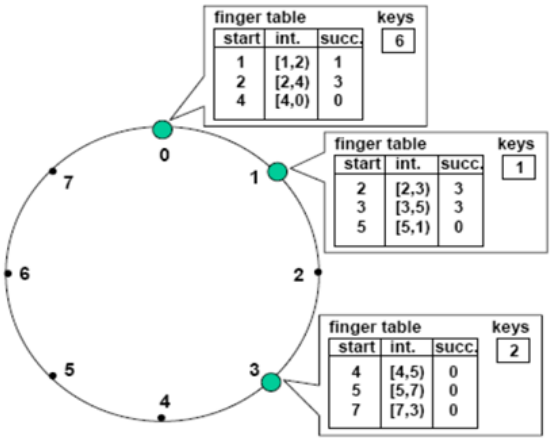


FIGURE 3.3 – Architecture du système Chord

### c. Troisième génération

La troisième génération de réseaux pair à pair correspond à des architectures hybrides qui associent à la fois l'architecture centralisée et décentralisé. Ce type d'architecture utilise plusieurs pairs (appelés super-pairs ou super-noeuds) qui ont la possibilité d'indexer et de contrôler un ensemble de pairs connectés au système. Un super-pair est connecté à d'autres super-pairs suivant le modèle de l'architecture décentralisée. Les super-pairs doivent rester suffisamment nombreux pour ne pas entraîner l'arrêt du système en cas de perte ou d'arrêt d'un super-pair. Par conséquent, si une recherche d'un pair n'est pas indexée par le super-pair auquel il est rattaché, celui-ci transmet alors la requête vers un autre super-pair. Le système KaZaA [8] est un exemple de réseau pair à pair de cette génération. La figure 3.4 illustre un schéma d'une architecture pair à pair de troisième génération. Il existe deux types d'architecture hybride : l'architecture hybride statique et dynamique. Dans le premier cas, une machine peut devenir un super-pair si l'utilisateur le souhaite. Dans le deuxième cas, le logiciel client permet de devenir automatiquement

un super-pair sous certaines conditions. Cette génération d'architecture présente les avantages des deux générations précédentes : la tolérance aux fautes et la réduction du trafic des requêtes et du temps de recherche ; mais, elle est plus complexe à mettre en oeuvre.

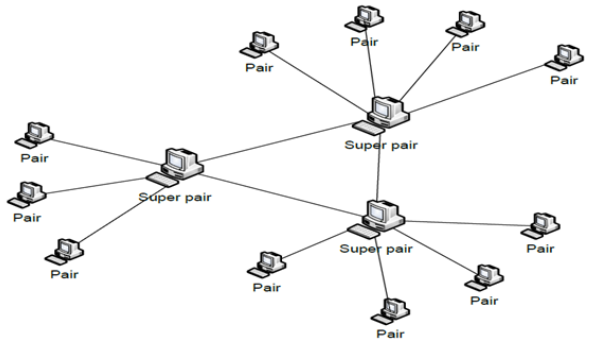


FIGURE 3.4 – Schéma d'une architecture pair à pair de troisième génération

## 3.2.2 Cloud P2P

### 3.2.2.1 Architectures de Cloud Computing

Dans[43], les auteurs décrivent les éventuelles architectures du Cloud Computing : centralisé, fédéré et P2P, cette description est illustré dans la figure 3.5 :

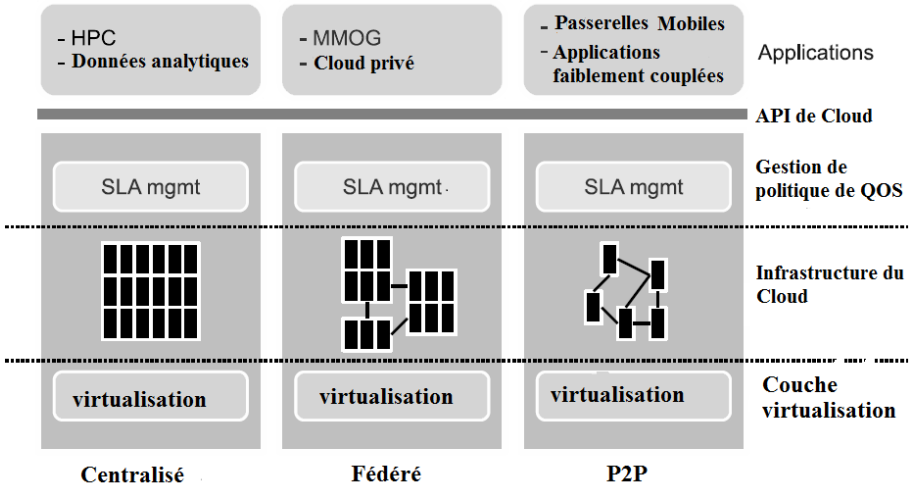


FIGURE 3.5 – Les dimensions du Cloud Computing

Les Clouds centralisés constituent les offre commerciales et les applications tel que : le calcul scientifique et le traitement de données, par contre les Clouds fédérés sont une évolution logique de l'approche centralisée, ils impliquent de multiples Clouds qui sont liés

pour construire un Cloud vaste. La fédération peut être utilisée afin d'améliorer la fiabilité à travers le partitionnement physique du pool de ressources, mais aussi pour résoudre les problèmes de la latence de communication par le client contraignants pour le centre de données le plus proche. En outre ce type de Cloud intéresse les entreprises qui hésitent à transférer leurs données à partir de l'entreprise à un fournisseur de service pour des raisons de sécurité.

Les Clouds P2P pourraient être construits pour un assemblage de pairs (individus) sans aucune surveillance centrale ou un composant de coordination. Les Clouds P2P permettent de provisionner les ressources à faible coût ou nul, les applications distribuées faiblement couplées les ou les emplacements physiques des noeuds est important. Le Cloud API fournit une interface pour la négociation des ressources, l'allocation et la surveillance.

#### 3.2.2.2 Pourquoi un Cloud P2P ?

Tandis que les Clouds P2P sont peu susceptibles pour fournir des dispositifs et les garanties QoS d'un Cloud centralisé ou fédéré, il existe des scénarios d'utilisation pour lesquels une architecture distribuée d'un Cloud peut être utile. Un Cloud P2P peut assembler à aucun coût les ressources existantes, donc de petits ou de même moyens organismes pourraient transformer les ressources en infrastructure de calcul qui peut être utilisé par des clients internes.

#### 3.2.2.3 P2P VS Volunteer computing (VC)

Volunteer Computing est un paradigme informatique bien connu, où les utilisateurs exécutent des applications tierses en installant une application spécifique sur leurs PC, ces derniers récupèrent et traitent les données d'entrées à partir d'un site central ensuite ils renvoient les résultats.

Les VCs sont principalement destinés aux scientifiques embrassant les applications parallèles par exemple le système de BOINC largement utilisé dans [44] sépare le programme client de la partie de l'application spécifique, ensuite les utilisateurs installent le client BOINC et sélectionnent les projets qu'ils supportent.

Les Clouds en général et les systèmes VCs ont quelques différences importantes vu la divergence de leurs objectifs, la table 3.1 nous aide à comprendre les différences majeures.

| IaaS Cloud | Volunter computing | P2P Cloud |
|---|---|---|
| Un seul fournisseurs de ressources | Plusieurs fournisseurs de ressources | Plusieurs fournisseurs de ressources |
| Envirenement virtualisé | Execution de l'application specifique | Envirenement virtualisé |
| Haute qualité de service | Qualité de service non garantie | Qualité de service non garantie |
| Centres de données ou distribué géographiquement | distribué géographiquement | Centres de données ou distribué géographiquement |
| Publique, privé ou hybride | Publique | Publique, privé ou hybride |

TABLE 3.1 – Comparaison des Clouds IaaS, du Volunteer computing et Clouds P2P .

D'après le tableau comparatif on constate que :

- Les ressources d'un Cloud appartiennent généralement a une seule entité généralement le fournisseur du Cloud, tandis que les VCs s'appuient sur les ressources fournies par des utilisateurs tiers.
- Un Cloud peut assurer un niveau élevé de qualité de service par contre un système VC ne peut fournir aucune garantie puisque les noeuds sont gérés par des utilisateurs individuels et peuvent être arrêtés à n'importe quel moment.
- Un Cloud est hébergé sur les grands centres de données tandis qu'un VC est distribué géographiquement, en plus de ça, un Cloud peut être publique, privé ou hybride par contre un VC est toujours publique vu que les ressources informatiques peuvent êtres exploitées par n'importe quel projet.

Et pour conclure cette comparaison on peut dire que les Cloud P2P empreintent des caractéristiques des Cloud IaaS est des VCs à la fois, un Cloud P2P diffère d'un VC parce qu'il n ya pas de coordination centrale ni référentiel central des tâches.

#### 3.2.2.4 Arechitecture P2P CS

Le P2P CS met en ouvre une collection de processus fonctionnant sur des hôtes distincts. Chaque processus est composé de plusieurs modules de logiciel organisés sous forme d'un modèle en couche. Un service d'échantillonnage de pairs PSS [45] vise a fournir à chaque noeud une liste de pairs pour échanger des messages entre eux, ce permet le maintien d'une superposition non structurée sur l'ensemble des pairs ensuite une nouvelle vue du système sera créee.

Le PSS peut garder la superposition même en présence de noeuds joignant et quittant

le système, cette caractéristique est fondamentale quand il s'agit d'un environnement dynamique ou les ressources sont gérées par des utilisateurs individuels. Afin de permettre une meilleure compréhension de l'architecture nous l'avons illustré sur la figure 3.6 :

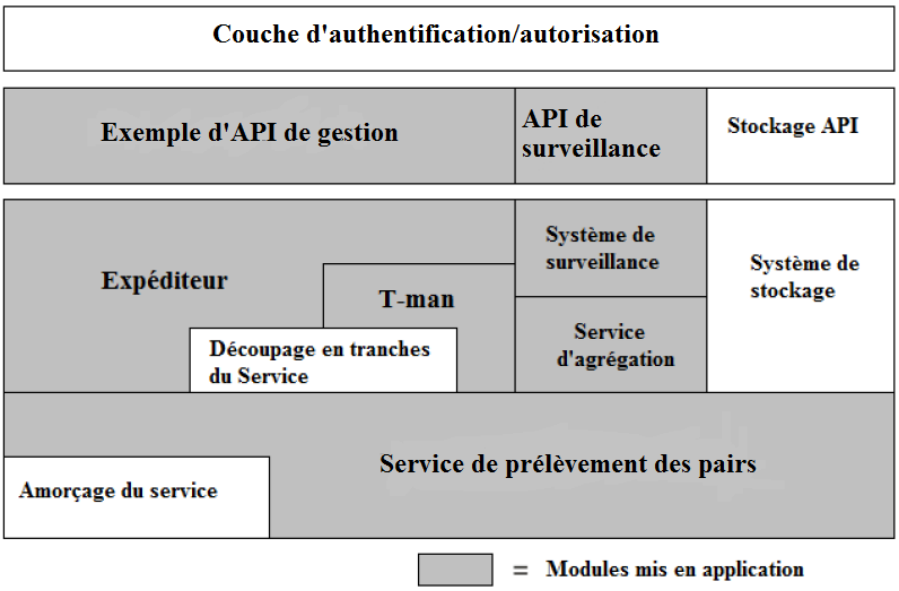


FIGURE 3.6 – Architecture en couches des P2P CS.

Le PSS utilise le service d'amorçage BS[46] sont rôle est de collecter l'ensemble des noeuds afin de commencer le transfert des messages, car au début chaque noeud ne connaît pas l'identité des autres dans le Cloud.

Le service de tranchage SS [47] est utilisé pour classer les noeuds selon certain critères, donc via ce service l'utilisateur peut demander des tranches de l'ensemble des pairs selon certains critères définis par lui-même.

Un autre service d'agrégation AS [48] est utilisé pour calculer des mesures globales en utilisant les échanges des messages locaux, donc l'AS permet à chaque noeud de connaitre les paramètres du système sans accéder à un registre à l'instar de la taille du réseau, la charge moyenne , nombre de partitions actives etc..

L'API de système de surveillance fournit deux opérations pour le démarrage et l'arrêt de l'affichage du temps exécution par contre L'API du système de gestion d'instance contient l'interface qui permet à un utilisateur de gérer les instances de ressources : création d'une nouvelle instance, la résiliation d'une instance active, l'énumération des ressources actuellement détenues et ainsi de suite. Elles sont similaires aux opérations EC2-run-instances, EC2-start-instances, EC2-stop-instances, EC2-terminer-instances et

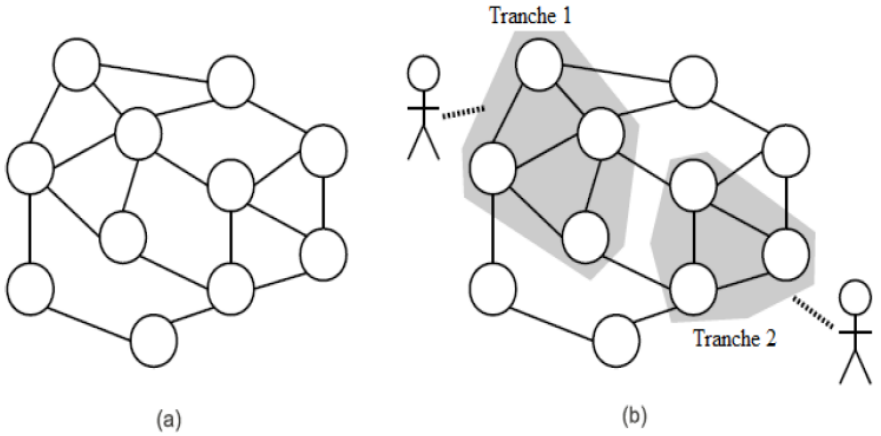


FIGURE 3.7 – Le découpage d'un nuage P2P en tranches.

ainsi de suite, fournies par le service Amazon EC2. Le répartiteur est chargé de traiter les demandes présentées par l'utilisateur via l'interface utilisateur de haut niveau, de les traduire ,ensuite les envoyer vers les autres noeuds.

T-Man [49] est un protocole basées sur la construction d'un réseau de recouvrement avec une topologie de données, P2P CS utilise T-Man pour lier l'ensemble des noeuds appartenant à la même couche.

## 3.3 Méthodes de débarquement du calcul

### 3.3.1 Communication client/serveur

Dans le processus de communication client/serveur, la communication est faite à travers un dispositif mobile, et un dispositif de substitution via des protocoles tel que des procédures d'appel à distance (RPC), l'invocation à distance (RMI) et les sockets. Aussi bien que les RPC et les RMI supportent les APIs et sont considérés stables par les développeurs, le déchargement à travers ces deux méthodes signifie que les services doivent être préinstallés sur les dispositifs participants. Ceci est un inconvénient lorsqu'on considère des réseaux ad hoc et la nature mobile du Cloud Mobile ainsi que les limites de la mobilité des utilisateurs en cas de proximité d'appareils qui ne supportent pas les services nécessaires[50].

Marinele [50] a présenté le Hyrax pour les applications androïd de Smartphones qui sont distribuées en terme de données et calcul basés sur Hadoop, la mise en commu-

nication à travers une plate forme android. Le Hyrax explore la possibilité d'employer un cluster de téléphones portables comme fournisseurs de ressources et montre la praticabilité d'un Cloud mobile. Ensuite il présente le HyraxTube qui représente une simple recherche multimédia mobile distribuée, son objectif est de permettre à des utilisateurs d'effectuer une recherche dans des dossiers multimédia en termes de temps, qualité et endroits. Spectra[51] et chroma[52] sont deux exemples des systèmes qui emploient des services préinstallés accessible via RPC afin de déparquer le calcul au niveau du Cloud. Les applications utilisent RPC afin de faire appel aux fonctionnalités dans les serveurs "Specrta". Quand le dispositif mobile doit débarquer une application, le client de spectra consulte une base de données qui stocke des informations sur les serveurs spectra tel que leurs disponibilités ainsi que la charge CPU etc. Ces serveurs sont préinstallés avec le code d'application agissant en tant que services. Les développeurs ont besoin de partitionner manuellement les applications en précisant les méthodes qui pourraient être candidates pour le débarquement, ensuite Spectra choisit lors de l'exécution et selon un pool de ressources les méthodes à utiliser parmi celles mentionnées ci-dessus.

Appache Hadoop est une implementation open Source de MapReduce[53] et fournit une interface virtualisée à un cluster d'ordinateurs mesurés aléatoirement. Dans Hyrax, un serveur central avec l'accès au dispositifs mobiles coordonne les données et le calcul, ce qui permet aux téléphones portables de communiquer.

Un autre Framework basé sur Hadoop est présenté par Huerta Hanepa et Lee [54] tel que pour un Cloud mobile virtuel concentrant sur les objectifs communs dans lesquels les dispositifs mobiles sont considérés comme fournisseurs de ressources. Ils arguent du fait que l'endroit des personnes joue un rôle important dans leurs activités. Le système de débarquement organise l'envoi et la réception des travaux des dispositifs mobiles ensuite il crée une machine virtuelle cette dernière s'occupe de l'exécution des tâches.

Dans [55], Deboosere et al proposent un modèle de grille, ou un dispositif mobile se relie à un serveur en tant que client au dessus d'un protocole classique VNC. Dans ce systeme, l'entrée de l'utilisateur est envoyée au serveur via un réseau sans fil, et après le traitement de l'entrée le serveur renvoie les résultats au Cloud. En particulier, cette recherche se concentre sur des algorithmes de choix de serveurs qui sont nécessaires quand l'endroit des dispositifs mobiles change. Afin de réduire le temps de réponse du serveur l'application peut être migrée à un serveur voisin.

### 3.3.2 Virtualisation

Un des premiers travaux dans cette direction Paranoid Android [56], décrit un prototype dans lequel les contrôles de sécurité sont appliqués à distance sur les clones du vrai dispositif dans le Cloud. Les expériences avec l'OS cloné dans les Cloud privés et publics montrent l'efficacité de l'approche. Plus tard, les auteurs dans [57] présentent un système qui permet à des utilisateurs de créer des images virtuelles de leurs Smartphones sur le Cloud. Leur prototype offre des applications fonctionnant sur le Cloud pour apparaître comme des applications locales sur l'utilisateur physique du dispositif qui s'interagit avec lui et commande le Smartphone virtuel à distance. Après l'exécution des commandes, l'appareil virtuel envoie de nouveau au vrai dispositif non seulement les données de sortie mais examine également les mis à jour. Leurs résultats expérimentaux sur le Cloud privé (serveurs locaux) prouvent que leur système est particulièrement proportionné pour les applications comportant de nombreux calculs qui sont exécutées à distance sur les Smartphones virtuels, dans lesquels seulement les résultats graphiques sont transmis au dispositif physique.

Dans [36] les auteurs décrivent MAUI, un système de débarquement qui exploite des modèles d'évaluation et des profileurs d'application pour que les applications débarquent en particulier des méthodes lourdes sur le Cloud. D'une part, l'approche dans [58] se fonde sur un clone de logiciel (le CloneCloud) sur le Cloud de l'OS d'un Smartphone. La méthodologie de débarquement est comme suit : le code binaire de l'application est divisé et une analyse autonome décide quelle partie du code binaire sera émigrée sur le CloneCloud. Leurs expériences avec des clones sur le Cloud privé fait de serveurs locaux révèlent à 20x moins de consommation d'énergie avec des applications telles que le balayage de virus, la recherche d'image et le profilage comportemental. Similaire à CloneCloud est Cloudlet [59], dans lequel l'image virtuelle réside sur une machine voisine (au vrai dispositif) et ses services sont employés au-dessus d'un LAN sans fil. Plus récemment ThinkAir [60] combine les approches de MAUI et de CloneCloud : L'OS du dispositif entier est cloné sur le Cloud et le débarquement est décidé au moyen d'un modèle d'évaluation en ligne. Il fournit une méthode efficace pour effectuer l'attribution de ressource sur demande et exploite le parallélisme sur le Cloud en créant, en reprenant et en détruisant dynamiquement les clones virtuels une fois nécessaire.

Le travail dans [61] présente les modèles d'énergie dont la différence de consomma-

tion d'énergie sur le dispositif mobile contre l'énergie a eu besoin pour envoyer les données au Cloud, pour chiffrer, etc. Les auteurs de [62] discutent les facteurs principaux qui affectent la consommation d'énergie des applications mobiles dans le Cloud Computing, et considèrent que les facteurs sont la charge de travail, les modèles de transmission de données, utilisation de WLAN et 3G. Dans les travaux plus récents de [63] les auteurs présentent une étude analytique pour trouver la politique optimale d'exécution. Ceci est identifié en configurant de façon optimale la fréquence du signal d'horloge dans le dispositif mobile pour réduire au minimum l'énergie utilisée pour le calcul et en programmant de façon optimale le débit au-dessus d'un canal sans fil stochastique pour réduire au minimum l'énergie pour la transmission de données. En formulant ces issues comme problème d'optimisation ils obtiennent les solutions qui donnent la perspicacité analytique en trouvant la décision de débarquement optimale.

Dans [64] les auteurs ont mis en application le Clone to Clone, une plateforme distribuée de P2P pour des Clone du Cloud qui associe un clone de logiciel à chaque dispositif mobile, C2C offre des services tel que le partage de contenant, la recherche et l'exécution distribuée. Sur C2C les auteurs ont construit CloneDoc un système de collaboration en temps réel pour les utilisateurs de dispositifs mobiles et CloneBox, un système de partage de fichiers et de stockage avec configuration.

### 3.3.3 Agent mobile

Scanvenger[65] est un Framework qui emploie une approche de code mobile pour diviser et distribuer le calcul, il présente également un programmateur pour l'évaluation de coût. Sa méthode d'évaluation de coût est basée sur la vitesse du serveur, il est possible que le dispositif débarque le calcul sur plusieurs serveurs afin de réduire le temps d'exécution.

### 3.3.4 Discussion

Excepté le hyrax, les travaux les plus récents ont employé la virtualisation et l'agent mobile pour débarquer le calcul et les tâches. Même les projets mentionnés ci-dessus sont basés sur de méthodes anciennes à l'instar de hadoop. Par conséquent, il est sûr de dire que les tendances dans ce secteur favorisent la virtualisation et l'agent mobile au dessus des systemes de communication client-serveur. Les avantages de ces deux méthodes par

rapport a la communication client-serveur est l'utilisation des RPC, en plus de sa les communications client-serveur ont des APIs et exigent des applications préinstallé.

Le tableau 3.2 comparatif permet d'avoir une vue globale sur l'ensemble des méthodes représentées ci-dessus.

| Nom et type | objectif | Surveillance de Resource |
|---|---|---|
| Spectra,Chroma : Client/serveur | utilisateur spécifiable : réduction de temps d'exécution, utilisation d'énergie et fidélité croissante | oui : CPU, réseau, la batterie, l'état de fichier caché, CPU à distance et l'état de fichier caché à distance |
| Scavenger : agent mobile | améliorer la performance et l'économie d'énergie | non |
| MAUI : virtualisation | Premier but est l'économie d'énergie, la vitesse d'exécution est également considérée. | Oui : CPU, largeur de bande de passante et la latence . |
| CloneCloud : virtualisation | utilisateur spécifiable : réduire le temps d'exécution l'utilisation d'énergie. | Oui : CPU, réseau et la mémoire. |
| C2C : virtualisation | utilisateur spécifiable : améliorer la vitesse d'exécution et l'économie d'énergie | Oui : CPU, largeur de bande de passante . |

TABLE 3.2 – Vue globale sur l'ensemble des méthodes de débarquement du calcul.

Nous constatons que l'objectif de ces méthodes est de réduire la consommation d'énergie, le temps d'exécution ainsi que l'augmentation des performances donc elles ne sont pas bien placées pour gérer la mobilité des dispositifs mobiles et d'assurer des communications fiables et sans coupure, ce qui nous ramène à étudier quelques mécanismes de gestion de mobilité.

## 3.4 Gestion de mobilité

Une des questions clés produites dans un Cloud est la conception des technique de gestion de mobilité qui soutiennent la mobilité d'un dispositif mobile tout en fournissant un service sans coupure. Bien que la recherche faite à propos de la mobilité se base sur la gestion d'endroit dans un réseau sans fil[66], ici nous nous concentrons sur des méthodes suivies pour contrôler et supporter la mobilité dans le Cloud Computing Mobile.

La détermination de l'endroit courant d'un dispositif mobile est utile pour évaluer ses déplacements dans un réseau. Le travail sur la localisation tombe principalement sur

les méthodes basées sur l'infrastructure telle que GSM (Global System for Mobile), wifi, ultrason avec RF (Radio Frequency ), GPS (Global Positioning System), RFID ( Radio Frequency IDentification) et l'IR ( Infra Rouge) , ou les méthodes basées sur les pairs.

Dans[67], la disponibilité d'un dispositif mobile est déterminée par la mobilité. Pour prévoir le statut de disponibilité des enregistrements des ressources mobiles qui sont employés en utilisant un modèle de châines de Markov. Par exemple les utilisateurs sont classifiés selon le degré de mobilité, on obtient donc des utilisateurs à mobilité faible, moyenne et élevée et ceci est utilisé pour calculer la probabilité de la mobilité.

Dans le service d'une communication fiable, le message confié à être délivré à l'instant "t" si l'expéditeur et le destinateur sont connectés physiquement pendant ce temps. Dans [68] les auteurs pensent que le calcul du délai de délivrance de message joue un rôle crucial dans la prestation de garantie de messages entre les noeuds avant toute partition du réseau ou défaillance de la couche physique se produit ce qui implique l'intégration du protocole de délivrance dans lequel il représente la fonction de l'état courant de transmission de gamme, l'emplacement, la vitesse et la direction des noeuds mobiles participant à la communication de groupe. Le temps de livraison doit être inferieur au temps pris pour les noeuds qu'ils veulent quitter la zone de communication.

## 3.5 Conclusion

Les Clouds P2P visent à Mettre en place des capacités de surveillance et de gestion totalement décentralisée, allocation des nœuds disponibles pour une tâche donnée en minimisant bien sur le nombre de noeuds , en plus de sa les P2P CS fournit des informations à propos du nombre de noeud occupés ainsi que le temps du CPU consommés par un utilisateurs. Après avoir présenté les points importants des P2P CS, quelques approches existantes à propos des méthodes de débarquement du calcul, ainsi que celle relatives à la gestion de mobilité, nous avons tiré quelques remarques qui seront utiles dans le chapitre suivant, lorsque nous essayons de proposer une solution qui répond aux questions posées de notre problématique.

Chapitre 4

# *Architecture pour le Cloud mobile en mode P2P centralisé*

## 4.1 Introduction

En dépit de l'utilisation croissante du Cloud mobile, ce paradigme reste toujours exposé aux problèmes tels que la rareté de ressources, déconnexion fréquente et la mobilité. Dans ce chapitre nous allons proposer une solution en se basant sur des approches étudiées et énumérer les exigences d'une plate forme Cloud ensuite nous essayions de valider cette solution en proposant un prototype qui va être simulé en JAVA.

### 4.1.1 Principe de la solution

Notre solution consiste à mettre en ouvre une architecture pour le Cloud mobile en mode P2P en s'inspirant de la plate forme clone to clone déjà présentée dans le chapitre précédant, ensuite nous proposons un mécanisme de collaboration en temps réel entre les acteurs de notre architecture en faisant appel à des approches étudiées tels que MAUI (Memory Arithmetic Unit and Interface) [36] et CloneCloud.

Puisque nous travaillons sur l'utilisation des systèmes P2P avec le Cloud mobile, donc nous rencontreons les dispositifs mobiles et plus précisément les Smartphones, iPHONE etc..., d'un côté et des services multimédias offerts à ces derniers d'un autre côté, nous avons décidé de mettre en oeuvre un mécanisme de partage de fichiers. Pour conclure nous intégrons dans les mécanismes précédant un protocole de garantie qui permet d'assurer la gestion de mobilité ce qui ramène à une solution qui réunie le débarquement du calcul et la gestion de mobilité ce qui n'est pas le cas dans les approches étudiées.

## 4.2 Conception d'une architecture pour le Cloud mobile en mode P2P

En premier lieu, notre modèle permet à chaque dispositif mobile voulant adhérer au Cloud de créer son propre clone. Par conséquent chaque clone a besoin de savoir comment se connecter aux autres clones. Pour cela nous allons mettre en ouvre un super clone qui comprend un service d'annuaire (tel que les adresses IP des clones crées, leurs VCPU et leurs états "libre" ou "occupé") dont l'objectif est de prendre en charge la création des clones ,ensuite une plate forme des clones sera crée à base du modèle P2P centralisé dont le serveur est le super clone et les hôtes sont les clones crées.

Afin de permettre une description explicite de notre architecture, nous allons montrer les différentes étapes de création d'un clone et comment il se relie à son dispositif mobile. Donc un utilisateur désireux de rejoindre la plateforme des clones a deux possibilités, soit il clone le dispositif mobile lui même ou il demande au super clone de lui cloner son dispositif en précisant également les informations d'accès des autres clones vers le sien. Dans le premier cas, après que l'utilisateur crée son propre clone, ce dernier l'annonce au super clone sinon le super clone s'occupe de la création du clone ainsi qu'une adresse IP qui sera livrée à l'utilisateur ainsi que son clone dont le but est de permettre une communication entre eux. Cependant dans les deux cas, l'utilisateur dispose d'un contrôle total du clone, y compris ce qu'il faut installer/mettre en publique/être accessible.

- **L'enregistrement auprès du super clone :**

  Apres la création d'un clone lui même s'inscrit auprès du super clone en lui envoyant une requête d'enregistrement ensuite le super clone mappe le dispositif mobile au clone. Si le clone est crée par l'utilisateur donc ce dernier détient une adresse IP publique qui sera affectée au clone via le super clone. Dans le cas contraire le super clone attribue une adresse IP au clone créé à partir de son pool d'adresse IP, cette partie est illustré sur la figure 4.1 étape 1.

- **Recherche de l'utilisateur :**

  Lorsque l'utilisateur doit se connecter à son clone (s'il n'est pas créé par l'utilisateur) il obtient l'adresse IP à travers une recherche auprès du super clone, ensuite il la mémorise dans son dispositif mobile, cette partie est illustrée sur la figure 4.1 étape 2.

- **Connexion utilisateur/clone :** Lors de la création du clone un VCPU et une VRAM lui sera affecté, l'utilisateur sera donc prêt à établir une communication avec son clone via son adresse IP et installe tout ce qu'il veut sur le clone, cette partie est illustrée sur la figure 4.1 étape 3.

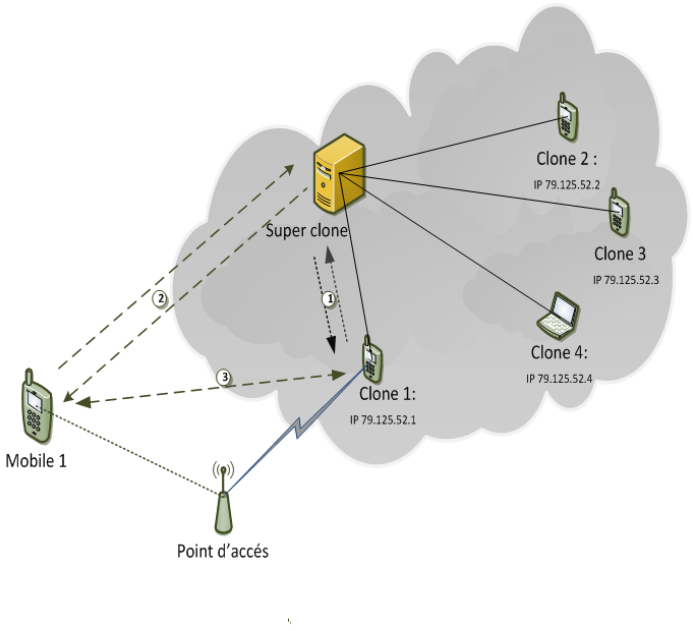


FIGURE 4.1 – Description des étapes de création d'un clone.

## 4.3 Mécanisme de collaboration en temps réel

### 4.3.1 Présentation du mécanisme

L'objectif de notre mécanisme est de permettre une collaboration en temps réel entre les acteurs de la plateforme, pour cela nous allons faire appel à des approches étudiées telles que MAUI ou CloneCloud.

Pour permettre une description explicite de ce mécanisme, nous allons décrire les déférentes étapes lorsque le dispositif mobile désire exécuter une application ou une partie d'elle dans le Cloud. Ceci est illustré comme suit :

Initialement, en utilisant les approches telles que MAUI ou CloneCloud, une analyse décide quelle partie de l'application sera migrée dans le Cloud.

- **Etape 1 :**l'utilisateur de dispositif mobile envoie la partie de l'application qui doit être exécutée dans le Cloud à son clone dans le Cloud.
- **Etape 2 :**le clone reçoit la partie de l'application :
  - o Si le clone est libre et le temps d'exécution est petit alors :
    - Le clone A envoie un message de changement d'état(ReqChEt) au le super clone afin de mettre son état à 'Occupé'.
    - Il exécute la partie de l'application, puis aller directement à l'étape 8.
  - o Sinon, si le clone A est libre et le temps d'exécution est long alors :
    - Le clone A va solliciter les autres clones de la plateforme des clones en envoyant un message(ReqC) au super clone pour demander la liste des adresses IP des clones qui sont à l'état "Libre " , en incluant dedans un message de changement d'état afin de rendre sont état "Occupé".
  - o Sinon, si le clone A est occupé alors :
    - Il va enfiler la partie de l'application dans une file d'attente afin de l'exécuter lorsque l'exécution des applications qui la précèdent s'achève.
- **Etape 3 :**le super clone va construire une liste L d'adresses IP ainsi que les performances des clones qui sont à l'état "Libre" afin de l'envoyer au clone sollicitant.
- **Etape 4 :** le super clone va mettre au courant les clones qui sont à l'état "occupé" en lui envoyant l'adresse IP du clone sollicitant , si l'un de ces clones (Occupé) termine l'exécution de ces tâches, il va rejoindre le clone sollicitant en le contactant directement(ReqR), si ce dernier n'a pas encore exécuté ses tâches et s'il reste des tâches à exécuter encore il va affecter une ou plus au clone désireux de joindre l'exécution , sinon il va lui envoyer un message de négation (ReqL) .
- **Etape 5 :** lorsque le clone sollicitant reçoit la liste L et si elle n'est pas vide, il va choisir un ou plusieurs clones les plus performants afin de leur affecter des tâches de l'application à exécuter via une communication P2P.
- **Etape 6 :** lors de la réception des tâches les clones choisis dans l'étape précédente vont envoyer un message de changement d'état au super clone afin de mettre leur l'état à 'Occupé'.
- **Etape 7 :**les clones choisis exécutent leurs tâches ensuite ils envoient les résultats au clone sollicitant.
- **Etape 8 :**le clone sollicitant, termine l'exécution après avoir récupèrer les résultats

puis il les envoie à son Mobile.

- **Etape 9 :** tous les clones qui ont terminé leurs tâches envoient un message de changement d'état (ReqChEt) au super clone afin de mettre leur état à 'Libre'.

**Exemple 1** :

Dans la figure 4.2 nous déroulons notre mécanisme de collaboration en temps réel quand le mobile A veut exécuter une application et débarquer une partie de cette dernière au Cloud (à son clone) :

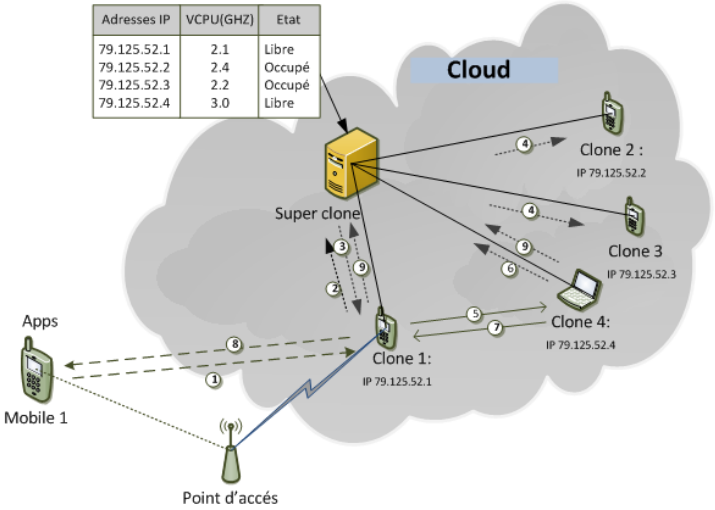


FIGURE 4.2 – Exemple illustratif d'une collaboration en temps réel entre les acteurs de C2C.

1- L'utilisateur du mobile 1 envoie la partie de l'application qui doit être exécuter dans le Cloud à son clone 1 .

2- Le clone 1 est libre et l'application à exécuter est grande alors ce dernier va solliciter les autres clones de la plateforme des clones en envoyant un message (ReqC) au super clone pour demander la liste d'adresses IP des clones qui sont à l'état "Libre".

3- Le super clone envoie l'adresse IP avec les performances du clone 4 au clone1.

4- Le super clone va mettre au courant les autres clones qui sont occupés (2 et 3) en leur envoyant l'adresse IP de clone 1.

5- Le clone 1 reçoit l'adresse IP du clone 4 et lui affecte une tâche de l'application via l'établissement d'une communication P2P.

6- Le clone 4 envoie un message de changement d'état au super clone afin de mettre

son état à 'Occupé'.

7- Le clone 4 exécute ses tâches ensuite il envoie les résultats au clone 1.

8- Le clone 1 termine l'exécution et récupère les résultats de toutes les tâches qui sont déléguées au clone 4, puis il les envoie au Mobile 1.

9- Les clones 1 et 4 ont terminé leurs tâches, donc ils envoient un message de changement d'état au super clone qui va mettre leur l'état à 'Libre'.

### 4.3.2 Description des algorithmes :

Apres avoir défini les étapes de notre mécanisme de collaboration du manière générale, maintenant nous allons les écrire sous forme de trois algorithmes à savoir :

- Algorithme $\mathbf{M}_i$ (mobile i).
- Algorithme $\mathbf{C}_i$ (mobile i).
- Algorithme SC (super clone).

#### 4.3.2.1 Algorithme du mobile $\mathbf{M}_i$

**Procédure** $M_i$ : (*Lorsque il veut executer une apps A*))
  (TL,TE) ← MAUI(A) ;
  envoyer (TE) à Ci ;
  traiter(TL) ;
**Fin**

Algorithme 1: Lorsque Mi veut executer une apps A.

#### 4.3.2.2 Algorithme du clone $\mathbf{C}_i$

Pour permettre une meilleure description de cet algorithme, nous allons le diviser en plusieurs procédures qui sont décrites ci-dessus :

**Declarations :**

**var** list_repense : ensemble des résultats **init** Φ ;
T : ensemble des tâches **init** Φ ;
n, m, y, k,nbr_rep : **entier** ;

```
Procédure C_i : (Lorsque il reçoit (TE) depuis M_i))
    Si (etat_i = "libre")Alors
        envoyer (ReqChEt) à SC ;
    Sinon
        Tant que (etat_i = "occupe")faire
            attente
        Fait
    Fin Si
    Si (TE « ) Alors
        R ← traiter (TE) ;
        envoyer(R) à M_i;
    Sinon
        envoyer (ReqC) à SC ;
        T ← MAUI(TE) ;
    Fin Si
    n ← Card(T) ;
Fin
```

Algorithme 2: Lorsque Ci reoit (TE) depuis Mi

```
Procédure C_i : (Lorsque il reçoit (list_IP) depuis SC)
    m ← Card(list_IP) ;
    y ← 1 ;
    k ← 1 ;
    Tant que ((k ≤ n) and (y ≤ m) ) faire
        envoyer (T[k]) à list_IP [k] ;
        y ← y+1 ;
        k ← k+1 ;
        nbr_rep ← nbr_rep +1 ;
    Fait
Fin
```

Algorithme 3: Lorsque Ci reçoit (list_IP) depuis SC.

```
Procédure Ci : (Lorsque il reçoit (IPj) depuis SC)
    Tant que (etat_i = "occupe")faire
        attente ;
    Fait
    envoyer (ReqR) à Cj ;
Fin
```

Algorithme 4: Lorsque Ci reçoit (IPj) depuis SC.

```
Procédure Ci : (Lorsque il reçoit (t) depuis Cj)
    r ← traiter(t) ;
    envoyer (r) à Cj ;
    envoyer (ReqChEt) à SC ;
Fin
```

Algorithme 5: Lorsque Ci reçoit(t) depuis Cj.

```
Procédure Ci : (Lorsque il reçoit (ReqR) depuis Cj)
    Si (k=n+1) Alors
        envoyer(ReqL) à Cj ;
    Sinon
        envoyer (t[k]) à Cj ;
        k← k+1 ;
        nbr_rep ← nbr_rep +1 ;
    Fin Si
Fin
```

Algorithme 6: Lorsque Ci reçoit(ReqR) depuis Cj.

```
Procédure Ci : (Lorsque il reçoit (ReqL) depuis Cj)
    envoyer (ReqChEt) à SC ;
Fin
```

Algorithme 7: Lorsque Ci reçoit(ReqL) depuis Cj.

```
Procédure Ci : (Lorsque il reçoit (r) depuis Cj)
    list_repense ← list_repense ∪{rj}; nbr_rep← nbr_rep-1 ;
    Si (k ≤ n ) Alors
        envoyer (tk) à Cj ;
        k ← k+1 ;
        nbr_rep ← nbr_rep+1 ;
    Sinon
        Si ( nbr_rep=0 ) Alors
            R ← traiter(list_repense) ;
            envoyer (R) à Mi ;
            terminer ;
        Fin Si

    Fin Si
Fin
```

Algorithme 8: Lorsque Ci reçoit(r) depuis Cj.

#### 4.3.2.3 Algorithme du SC

**Declarations :**

**Const** N : **entier**

**var** list_IP : ensemble des adresse IP **init** $\Phi$ ;

```
Procédure SC :(Lorsque il reoit (ReqChEt) depuis Ci)
    mettre etati← ¬etati;

Fin
```

Algorithme 9: Lorsque SC reçoit(ReqChEt) depuis Ci.

```
Procédure SC :(Lorsque il reoit (ReqC) depuis Ci)
    Pour j de 1 à N faire
        Si (etatj="libre") Alors
            list_IP ← list_IP U {IPj} ;
        Sinon
            envoyer (IPi) à Cj ;
        Fin Si
    Fin Pour
    envoyer (list_IP) à Ci ;
Fin
```

Algorithme 10: Lorsque SC reçoit (ReqC) depuis Ci

- **Exemple 2 :** Soit le mobile 1 (figure 4.3 ) désire exécuter une tache T (figure 4.4 ci-dessous), en utilisant des méthodes de MAUI, une partie de cette tache sera exécutée dans le Cloud. Dans ce qui suit, on présente les étapes de déroulement des Algorithmes ci-dessus et la figure 4.5 illustre les différentes requêtes et messages échangés entre les acteurs de la plateforme C2C.

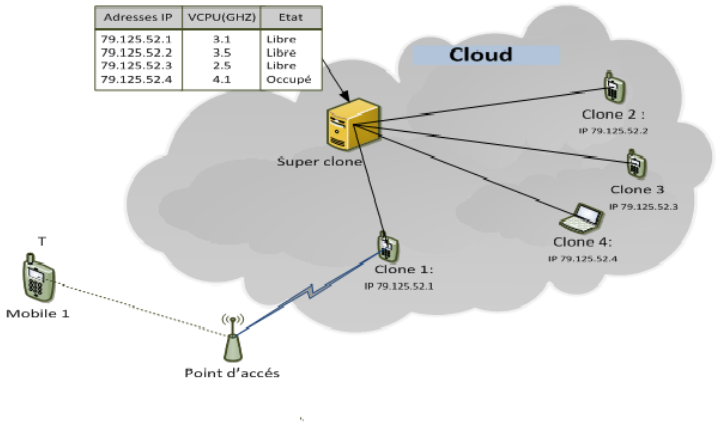


FIGURE 4.3 – Etat de la plateforme.

**Déroulement des algorithmes ci-dessus**

1- M1 veut exécuter T :

- En utilisant MAUI TL=T1 et TE=T2.
- envoyer(T2) à C1.
- traiter(T1).

2- C1 reçoit T2 depuis M1 :

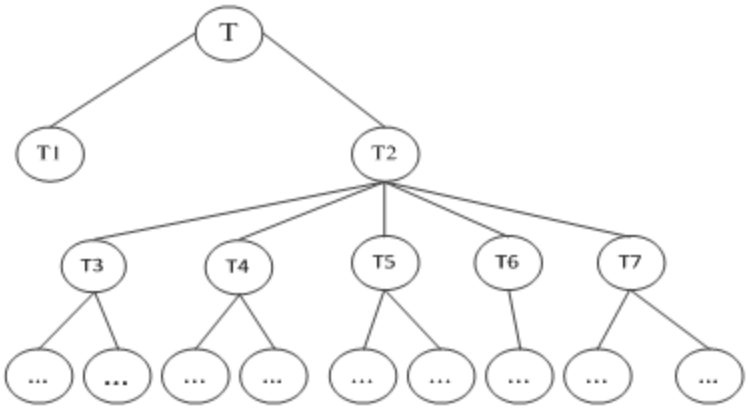


FIGURE 4.4 – Arbre de la tâche T.

- envoyer ReqChEt à SC.
- T2 » alors :
  - envoyer(ReqC) à SC.
  - diviser T2 ,tl= {T5},te={ T3, T4,T6,T7 },n=4.
  - traiter(T5) .

3- SC :

- SC reçoit ReqChEt depuis C1 :
  - mettre etat1=occupé .
- SC reçoit (ReqC) depuis C1 :
  - construire (list_IP), list_IP={(C2,3.5)(C3, 2.5)} .
  - envoyer(list_IP) à C1.
  - envoyer(IP1) à C4.

4- C1 et C4 :

- C1 reçoit list_IP depuis SC :
  - m=2, nbr_rep=0 .
  - envoyer (T3) à C2 .
  - y=2, k=2, nbr_rep=1 .
  - envoyer(T4) à C3.
  - y=3, k=3, nbr_rep=2.
- C4 reçoit IP1 depuis SC
  - attente .

5- C2, C3 et C4 :

- C2 reçoit T3 depuis C1 :
  - R3 = traitement (T3).
  - envoyer (R3) à C1 .
- C3 reçoit T4 depuis C1 :
  - R4 = traitement (T4) ;
  - envoyer (R4) à C1 .
- C4 termine ces taches :
  - envoyer (ReqR) à C1 .

6- C1 :

- C1 reçoit ReqR depuis C4 :
  - envoyer (T6) à C4 .
  - k=4, nbr_rep=3 .
- C1 reçoit R3 depuis C2 :
  - nbr_rep=2,list_repense={R3,T5}.
  - envoyer(T7)à C2, k=5, nbr_rep=3 .
- C1 reçoit R4 depuis C2 :
  - nbr_rep=2,list_repense={R3,R4,T5}.

7- C4 et C2 :

- C4 reçoit T6 depuis C1 :
  - R6 = traitement (T6) .
  - envoyer (R6) à C1 .
- C2 reçoit T7 depuis C1 :
  - R7 = traitement (T7).
  - envoyer (R7) à C1 .

8- C1 :

- C1 reçoit R6 depuis C4
  - nbr_rep=2.
  - list_repense={R3,R4,T5,R6}.
- C1 reçoit R7 depuis C2
  - nbr_rep=2.
  - list_repense={R3,R4,T5,R6,R7}.
  - R2= traitement de (list_repense) .

  - envoyer(R2) à M1 .
- C1, C2, C3, C4 envoient (ReqChEt) à SC

9- SC et M1 :

- SC reçoit (ReqChEt) depuis Ci :
  - Mettre etat de Ci à libre .
- M1 reçoit R2 depuis C1
  - Terminer.

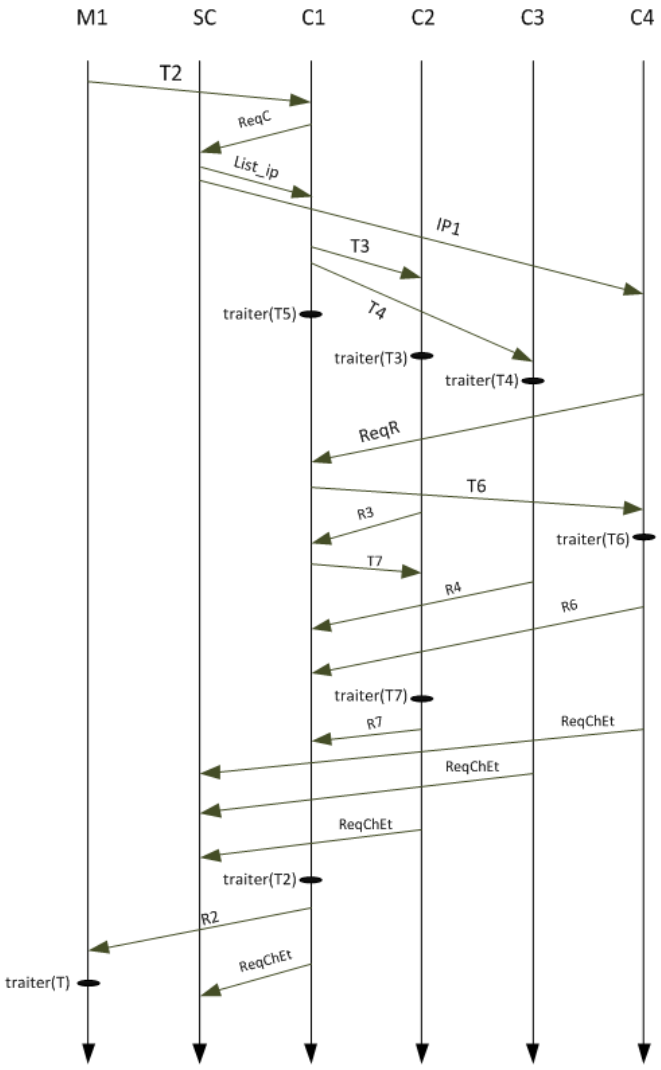


FIGURE 4.5 – Scénario de déroulement des algorithmes de mécanisme de collaboration en temps reel sur l'exemple précédant.

## 4.4 Système de partage de fichiers

### 4.4.1 Presentation du système

Notre mécanisme permet un partage de fichier entre les utilisateurs de la plateforme des clones, et pour que l'utilisateur du dispositif mobile A veut bénéficier d'un fichier dans le Cloud, il n'a qu'à exécuter les étapes suivantes :

- **Etape 1 :** le mobile A envoie le nom de fichier à son clone (A) dans le Cloud.
- **Etape 2 :**si le clone A possède ce fichier alors il va l'envoyer pour son vrai mobile, sinon il va solliciter les autre clone dans la plateforme en envoyant un message au super clone, dont il va demander la liste des adresses IP des clones qui possèdent ce fichier.
- **Etape 3 :** le super clone à son tour va diffuser le nom de fichier à tous les autres clones.
- **Etape 4 :**les clones qui possèdent ce fichier vont répondre par message positif(ReqP) et les autres par un message négatif(ReqN).
- **Etape 5 :** le super clone va envoyer la liste des adresses IP des clones qui ont répondu par un message positif au clone A.
- **Etape 6 :**le clone A demande le fichier au clone le plus performant en le contactant directement à son adresse IP (il envoie(ReqD, nom de fichier)).
- **Etape 7 :**le clone sollicité va répondre par l'envoi de fichier au clone A.
- **Etape 8 :** le clone A à son tour va envoyer le fichier reçu à son vrai mobile A.

**Exemple 1** :

Dans la figure 4.6 nous allons montrer ce qui se passe lorsque le mobile 1 a besoin du fichier "c.txt" :

1- Le mobile 1 envoie le nom de fichier qui l'a besoin ("c.txt") à son clone (1) dans le Cloud.

2- Le fichier n'existe pas dans la base de fichiers du clone 1, donc il envoie un message au super clone pour demander la liste des adresses IP des clones qui possèdent ce fichier.

3- Le super clone diffuse le nom de fichier à tous les autres clones.

4- Le clone 4 est le seul qui possède ce fichier donc, il répond par message positif(ReqP) et les autres par un message négatif(ReqN).

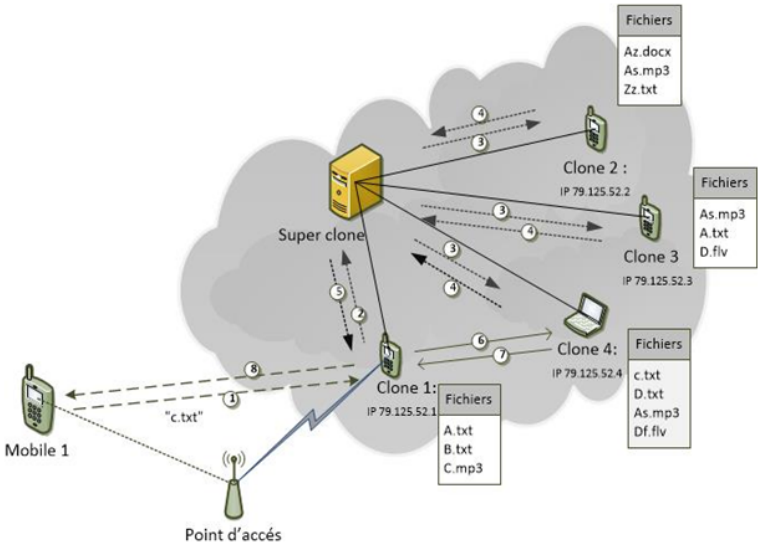


FIGURE 4.6 – Exemple illustratif d'un scénario de partage de fichier.

5- Le super clone va envoyer l'adresse IP de clone 4 au clone 1.

6- Le clone 1 demande le fichier au clone 4 via l'établissement d'une communication P2P

7- Le clone 4 va répondre par l'envoi de fichier au clone 1.

8- Le clone 1 à son tour va envoyer le fichier reçu à son vrai mobile 1.

### 4.4.2 Description des algorithmes

Apres avoir défini les étapes de notre système de partage de fichier d'une manière générale, maintenant nous allons les écrire sous forme de trois algorithmes à savoir :

- Algorithme $\mathbf{M}_i$ (mobile i).
- Algorithme $\mathbf{C}_i$ (clone i).
- Algorithme SC (super clone).

#### 4.4.2.1 Algorithme $M_i$ (mobile i).

```
Procédure Mi :(Lorsque il cherche un fichier (f) )
    Si (f existe dans la mémoire locale ) Alors
        Charger(f) dans la mémoire ;
    Sinon
        envoyer (nom_f) à Ci ;
    Fin Si

Fin
```

Algorithme 11: Lorsque Mi cherche un fichier (f) .

#### 4.4.2.2 Algorithme $C_i$ (clone i).

Pour permettre à une meilleure description de cet algorithme , nous allons le diviser en plusieurs procédures qui sont décrites ci-dessus :

```
Procédure Ci :(Lorsque il reçoit (nom_f) depuis mi )
    Si (f existe dans la mémoire locale ) Alors
        envoyer(f) à Mi ;
    Sinon
        envoyer (nom_f) à SC ;
    Fin Si

Fin
```

Algorithme 12: Lorsque Ci reçoit (nom_f) depuis mi.

```
Procédure Ci :(Lorsque il reçoit (nom_f) depuis SC )
    Si (f existe dans la mémoire locale ) Alors
        envoyer(ReqP) à SC ;
    Sinon
        envoyer (ReqN) à SC ;
    Fin Si

Fin
```

Algorithme 13: Lorsque Ci reçoit (nom_f) depuis SC.

```
Procédure Ci :(Lorsque il reçoit (list_IP) depuis SC )
    | envoyer (ReqD, nom_f) à l'un des adresses IP ;
    | reçoit(f) ;
    | envoyer (f) à Cj ;
    | SendReq (f) à Mi
Fin
```

Algorithme 14: Lorsque Ci reçoit (list_IP) depuis SC.

#### 4.4.2.3 Algorithme SC (super clone).

**Les déclarations :**

**var** N : **entier** // **init** nombre des clones dans C2C

list_IP : ensemble des adresse IP **init** $\Phi$ ;

```
Procédure SC :(Lorsque il reçoit (nom_f) depuis Ci )
    | diffuser (nom_f) ;
Fin
```

Algorithme 15: Lorsque SC reçoit (nom_f) depuis Ci.

```
Procédure SC :(Lorsque il reçoit (X)  depuis  C_i)
    | Si (X=ReqP ) Alors
    |     | N ← N-1 ;
    |     | list_IP ← list_IP U {IPj} ;
    | Sinon
    |     | Si ( X=ReqN ) Alors
    |     |     | N ← N-1 ;
    |     | Fin Si
    |     |
    | Fin Si
    | Si (N=0 ) Alors
    |     | envoyer (list_ IP) à Ci ;
    | Fin Si
    |
Fin
```

Algorithme 16: Lorsque SC reçoit (X)  depuis  C_i

La figure 4.7 représente l'interprétation de déroulement des algorithmes ci-dessus :

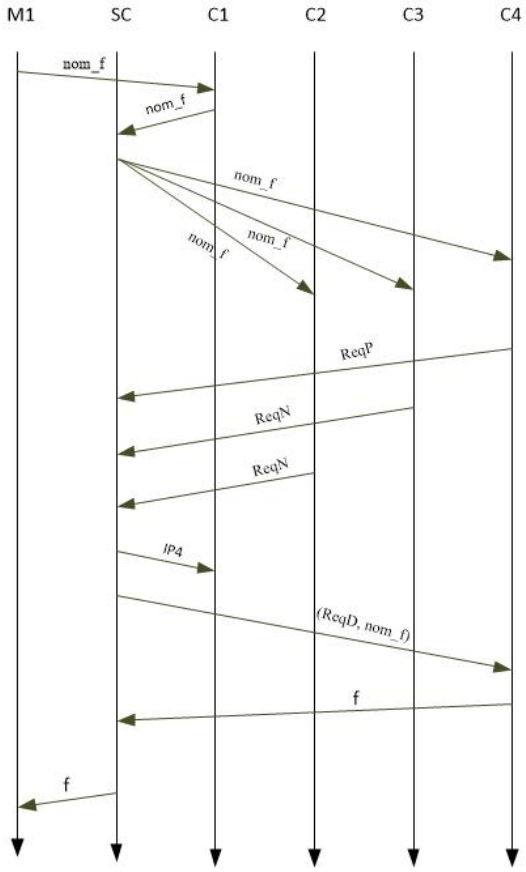


FIGURE 4.7 – Scénario de déroulement des algorithmes de mécanisme de partage de fichier.

## 4.5 Gestion de mobilité

Vue la mobilité des dispositifs mobile, des problèmes lors de la communication de mobile/clone peuvent être rencontrés, cependant afin de garantir une communication fiable et sans coupure nous allons mettre en ouvre un protocole de gestion de mobilité en s'inspirant de l'approche présentée dans la section 4.2 du chapitre précédant.

– **Algorithme de garantie :**

Pour permettre une meilleure description de notre algorithme nous allons représenter la zone de couverture ZC du réseau sous forme d'un cercle qui est illustré dans la figure 4.8 tel que C est le centre avec les coordonnées (x1,y1) et B représente le point le plus éloigné de C et qui appartient à la ZC avec ces coordonnées (x2,y2). Le mobile A qui désire se connecter au réseau nous lui associons les coordonnées (x3, y3).

**R :** rayon de ZC il représente la distance entre C et B R=[(x1,y1)-(x2,y2)]

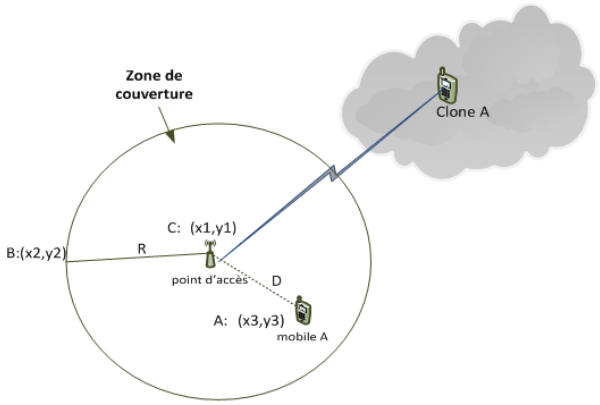


FIGURE 4.8 – gestion de mobilité.

**D :** distance entre A et C D =[(x3,y3)-(x1,y1)].

**V :** la vitesse du mobile A.

**TD :**TD : le temps de délivrance de données du clone au mobile.

```
Procédure SendReq( )
    Si ((A se déplace vers C) ou (D< < < <) ) Alors
        calculer le rapport D/V ;
        Si (TD<(D/V)) Alors
            livraison possible de données ;
            envoyer(données) au Mobile ;
        Fin Si
    Sinon
        Tant que (D<R) faire
            calculer le rapport D/V ;
            Si (TD<(D/V)) Alors
                livraison possible de données ;
                envoyer(données) au Mobile ;
            Sinon
                livraison non garantie de données ;
            Fin Si
        Fait
    Fin Si
Fin
```

Algorithme 17: SendReq pour la gestion de mobilité.

L'algorithme SendRec permet au super clone d'envoyer une requête au clone afin de lui donner l'autorisation pour commencer l'envoi de données.

## 4.6 Discussion de la solution proposée

Dans cette section nous allons essayer de citer les avantages et les inconvénients de notre solution.

### 4.6.1 Avantages

Dans les services de Cloud, s'il arrive rapidement à maturité, des perspectives considérables sont ouvertes : le principal intérêt réside dans le fait d'utiliser des smartphones, ou des netbooks (puisqu'Android motorise aussi les netbooks), des applications qui n'auraient pas pu être proposées sur un terminal mobile, car trop gourmandes en ressource de calcul. Mais il offre d'autres avantages comme une consommation moindre de la batterie vu le débarquement du calcul au niveau du Cloud ce qui permet au mobile de compter sur son clone ce qui ramené à une déconnexion permanente de ce mobile en attendant la livraison des résultats, d'où une meilleure autonomie du dispositif mobile, ou encore une utilisation plus confortable de celui-ci grâce à une meilleure rapidité d'exécution vu les capacités du clone à l'instar du VCPU et VRAM ainsi que des performances très élevés.

Pour conclure, cette solution permet à la résolution de l'un des problèmes majeurs du Cloud mobile qui est la mobilité des dispositifs mobile grâce à la mise en œuvre d'un mécanisme de gestion de mobilité.

### 4.6.2 Inconvénients

Cette solution reste toujours ouverte aux problèmes, à l'instar de la sécurité et l'intégrité des données puisque en regroupant des ressources sur Internet on peut perdre une partie du contrôle sur celles-ci. Dès lors que des données transitent sur Internet, le risque de piratage est bien plus présent que sur une utilisation locale, en plus de ça nous pouvons dire que l'absence d'un mécanisme de tolérance aux pannes peut créer des problèmes puisque la communication entre les clones s'effectue à l'intermédiaire du super clone, donc en cas de panne de ce dernier il sera impossible d'établir ces communications. En revanche l'application de ce genre de solution nécessite la disponibilité d'un nombre

important de ressources, puisque à chaque fois qu'un nouveau dispositif mobile désire débarquer le calcul au Cloud un clone doit être créé.

## 4.7 Conclusion

Après avoir présenté notre solution en mettant en œuvre une architecture pour le Cloud mobile en mode P2P et en incluant à l'intérieur trois mécanismes différents , à l'instar de la collaboration en temps réel, le partage de fichier ainsi que la gestion de mobilité, nous présentons dans le chapitre suivant l'implémentation du premier mécanisme et d'expliquer les différentes étapes de cette dernière.

Chapitre 5

# *Implementation*

## 5.1 Introduction

Après avoir présenté notre solution précédemment qui consiste à la mise en œuvre d'une architecture pour le CloudComputing en mode P2P ainsi que les mécanismes de fonctionnement, nous passons dans ce chapitre à l'implémentation en programment un prototype sous JAVA en faisant appel à la procédure de l'invocation à distance RMI (Remote Method Invocation). Ce chapitre s'articule autour de deux axes principaux. Dans la première partie, nous détaillons les étapes et la mise en place du RMI. Quant à la deuxième partie, nous intégrons le mécanisme de collaboration en temps réel présenté dans le chapitre précédant.

## 5.2 Description de RMI

RMI permet de créer un modèle objet distribué java qui s'intègre naturellement au langage java et au modèle objet local. Ce systèmeétend la sécurité et la robustesse de java au monde applicatif distribué. Avec le RMI, les méthodes de certains objets (appelés objets distants)peuvent être invoquées depuis des JVM différentes (espaces d'adressages distincts) de celles où se trouvent ces objets, y compris sur des machines différentes via le réseau. En effet, RMI assure la communication entre le serveur et le client via TCP/IP et ce de manière transparente pour le développeur. Il utilise des mécanismes et des protocoles définis et standardisés tels que les sockets et RMP (Remote Method Protocol). L'architecture RMI définit la manière dont ce comportent les objets, comment et quand des exceptions peuvent se produire, comment gérer la mémoire et comment les méthodes appelées passent et reçoivent les paramètres. RMI préserve la sécurité inhérente à l'environnement java notamment grâce à la classe RMISecurityManager et au DGC (DistributedGabageCollector).

### 5.2.1 Utilisation de RMI

Nous allons voir les principales étapes nécessaires à la mise en place d'un service de type RMI.

#### 5.2.1.1 L'interface

La première étape consiste à créer une interface distante qui décrit les méthodes que le client pourra invoquer à distance. Pour que ses méthodes soient accessibles par le client cette interface doit hériter de l'interface Remote. Toutes les méthodes utilisables a distances doivent pouvoir lever les exceptions de type RemoteException qui sont spécifiques à l'appel distant. Cette interface devra être placée sur les deux machines (serveur et client).

#### 5.2.1.2 L'implémentation

Il faut maintenant implémenter cette interface distante dans une classe. Par convention, le nom de cette classe aura pour suffixe Impl. La classe doit hériter de la classe java.rmi.server.RemoteObject ou de l'une de ses sous classes la plus facile d'utilisation étant java.rmi.server.UnicastRemoteObject. C'est dans cette classe que nous allons définir le corps des méthodes distantes que pourront utiliser nos clients .Evidement , il est possible d'ajouter d'autre méthodes mais les clients ne pourront y accéder et donc ne pourront les utiliser .

#### 5.2.1.3 Stubs et Skeletons

Lorsque le client fera appel à une méthode distante, cet appel sera transféré au stub .Le stub est donc un relais du coté client. Il devra donc être place sur la machine cliente. C'est le représentant local de l'objet distant. Il « marshalise » (emballe) les arguments de la méthode distante et les envoie dans un flux de données. D'autre part, il « démarchais »(emballe) la valeur ou l'objet de retour de la méthode distante. Il communique avec l'objet distant par l'intermédiaire du skeleton. Le skeleton est lui aussi un relais mais du côté serveur .il devra être place sur la machine servant de serveur. Il « démarshalise » les paramètres de la méthode distante, les transmet à l'objet local et « marshalise » les valeurs de retours à envoyer au client.

Les stubs et les skeletons sont donc des intermédiaires entre le client et le serveur qui gèrent le transfert distant des données. On utilise le compilateur RMI pour la génération des stubs et des skeletons. C'est un utilitaire fourni avec le JDK, depuis la version 2 de java,skeletons n'existe plus. Seul le stub est nécessaire du côté client mais aussi du côté serveur

#### 5.2.1.4 Le registre RMI

Les clients trouvent les services distants en utilisant un service d'annuaire activé sur un hôte connue avec un numéro de port connu. RMI peut utiliser plusieurs services d'annuaire, y compris java Naming and Directory Interface (JNDI). Il inclut lui-même un service simple appelé Registry (rmiregistry). Le registre est exécuté sur chaque machine qui héberge des objets distants (les serveurs) et accepte les requêtes pour ces services, par défaut sur le port 1099. Un serveur crée un service distant en créant d'abord un objet local qui implémente ce service. Ensuite il exporte cet objet vers RMI. Quand l'objet est exporté ; RMI crée un service d'écoute qui attend qu'un client se connecte et envoie des requêtes au service. Après l'exportation, le serveur enregistre l'objet dans le registre de RMI sous un nom public qui devient accessible de l'extérieur. Le client peut alors consulter le registre distant pour obtenir des références à des objets distants.

#### 5.2.1.5 Le serveur

Notre serveur doit enregistrer auprès du registre RMI l'objet local dont les méthodes seront disponibles à distance. Cela se fait grâce à l'utilisation de la méthode statique bind() de la classe Naming. Cette méthode permet d'associer (enregistrer) l'objet local avec un synonyme dans le registre. L'objet devient alors disponible par le client.

#### 5.2.1.6 Le client

Le client peut obtenir une référence à un objet distant par l'utilisation de la méthode statique lookup() de la classe Naming. Il peut ensuite invoquer des méthodes distantes sur cet objet. La méthode lookup() sert au client pour interroger un registre et récupérer un objet distant. Elle prend comme paramètre une URL qui spécifie le nom d'hôte du serveur et le nom du service désiré. Elle retourne une référence à l'objet distant. La valeur retournée est de type Remote. Il est donc nécessaire de caster cet objet en l'interface dis-

tante implémentée par l'objet distant. En effet, le client travaillle avec l'interface distante et non avec l'objet distant lui-même.

#### 5.2.1.7 Lancement

Il est maintenant possible d'exécuter l'application. Cela va nécessiter l'utilisation de trois consoles. La première sera utilisée pour activer le registre rmiregistry. Dans une deuxième console, nous exécutons le serveur. Celui-ci va charger l'implémentation en mémoire. Répartitions des classes dans un cas concret d'application client/serveur la répartition doit se faire de la manière suivante :

- Le client doit avoir un accès aux interfaces distantes et aux stubs de leurs implémentations.
- Le serveur doit avoir un accès aux interfaces distantes, aux implémentations et aux stubs.

#### 5.2.1.8 Schéma récapitulatif

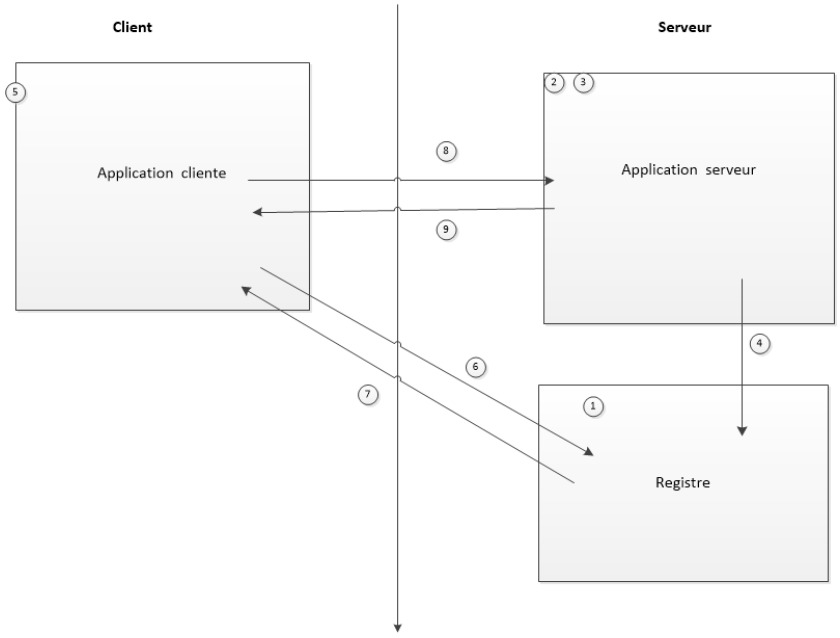


FIGURE 5.1 – Schéma récapitulatif du RMI.

1- On exécute rmiregistry.

2- On exécute le serveur.

3- On crée l'objet distant.

4- On l'enregistre dans l'annuaire avec la méthode Naming.bind().

5- On exécute le client.

6- On recherche un objet distant avec la méthode naming.lookup().

7- L'annuaire renvoie une référence de l'objet distant.

8- On invoque une méthode distante.

9- On reçoit une valeur de retour.

## 5.3 Intégration du mécanisme de collaboration en temps réel

L'intégration de notre mécanisme de collaboration en temps réel s'appuie sur trois axes principaux, afin de permettre une meilleure description nous allons les présenter comme suit :

- Communication mobile/clone.
- Communication clone/super clone.
- Communication clone/clone.

### 5.3.1 Communication mobile/clone

Cette étape consiste à l'établissement d'une communication entre le mobile et son clone en suivant les étapes décrites ci-dessus, dont le but est d'utiliser l'appel d'une méthode distante (Factorielle () comme exemple) avec un serveur clone celui-ci permet au mobile de calculer le Factorielle. La figure 5.2 permet de décrire explicite de cette étape :

1- On exécute rmiregistry.

2- On exécute le clone.

3- On crée l'objet distant.

4- On l'enregistre dans l'annuaire avec la méthode Naming.bind().

5- On exécute le mobile.

6- On recherche un objet distant avec la méthode naming.lookup().

7- L'annuaire renvoie une référence de l'objet distant.

8- On invoque une méthode distante Factorielle().

9- On reçoit une valeur de retour.

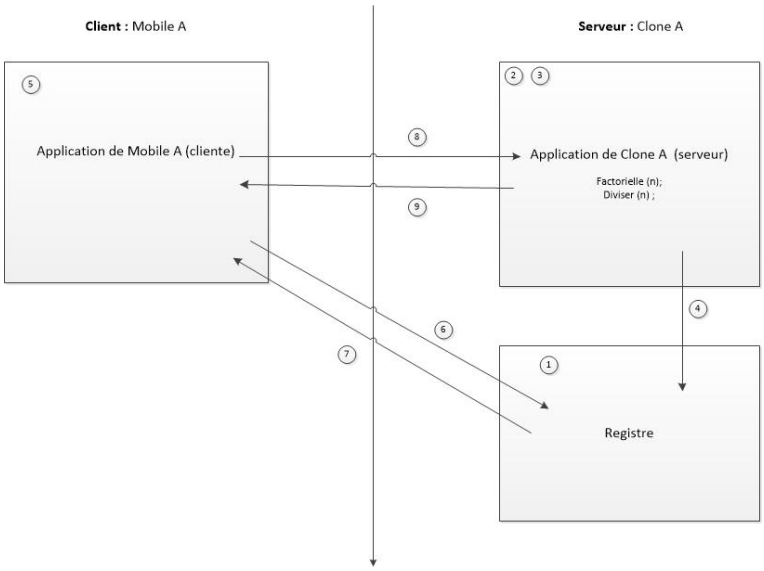


FIGURE 5.2 – Communication mobile/clone.

### 5.3.2 Communication clone/super clone

Cette étape représente l'objectif majeur du calcul distribué en permettant au clone de solliciter l'autre clone de la plateforme via le super clone qui contient un service d'annuaire représenté par une base de données sous MySQL. La figure 5.3 est la meilleure interprétation de cette étape :

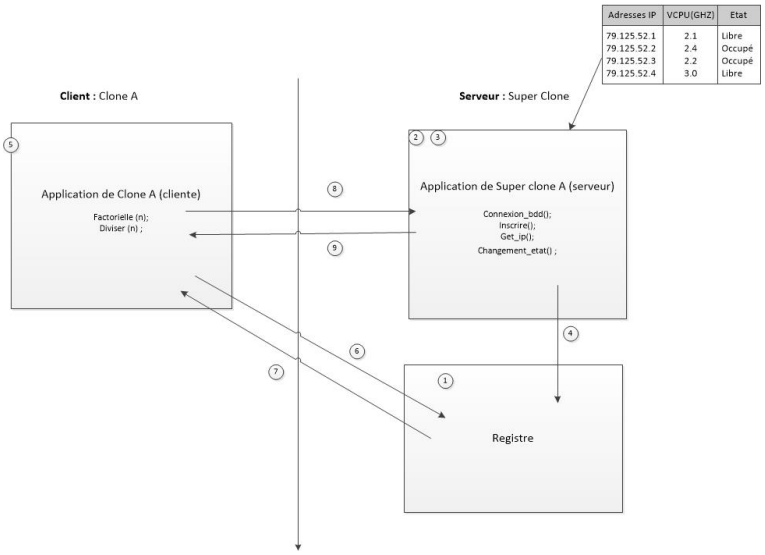


FIGURE 5.3 – Communication clone/super clone.

1- On exécute rmiregistry.

2- On exécute le super clone.

3- On crée l'objet distant.

4- On l'enregistre dans l'annuaire avec la méthode Naming.bind().

5- On exécute le clone.

6- On recherche un objet distant avec la méthode naming.lookup().

7- L'annuaire renvoie une référence de l'objet distant.

8- On invoque une des méthode distante.

9- On reçoit une valeur de retour.

### 5.3.3 Les méthodes invoquées à distance

- Connexion_bdd() permet au super clone de se connecter au service d'annuaire.
- Get_ip() permet au clone de recevoir les adresses IP des clone libre de la part du super clone.
- Inscrire() permet au clone de s'inscrire au service d'annuaire.
- Changement_etat permet au super clone de Changer l'état du clone.

### 5.3.4 Communication clone/clone

Cette étape permet à une communication entre le clone sollicitant et les clone de la plateforme en mode P2P afin de livrer à ces derniers les taches à exécuter. La figure 5.4 décrit cette étape :

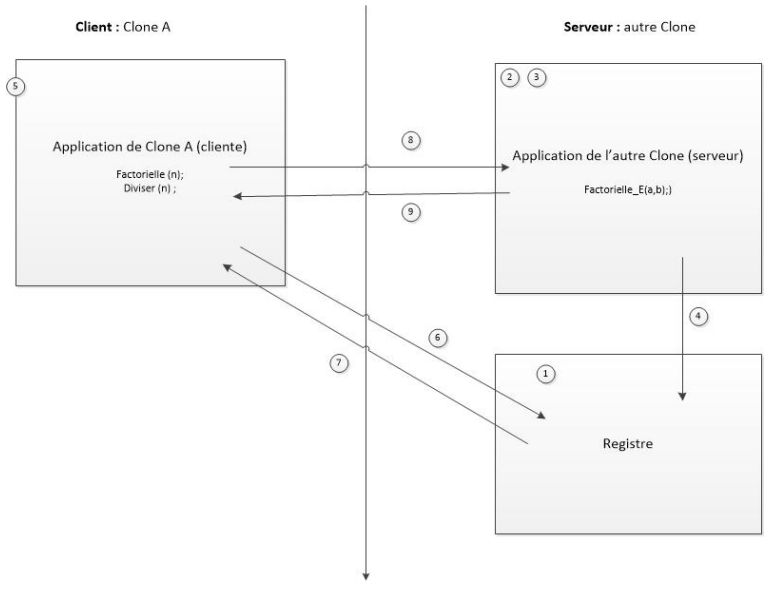


FIGURE 5.4 – Communication clone/clone.

1- On exécute rmiregistry.

2- On exécute le clone.

3- On crée l'objet distant.

4- On l'enregistre dans l'annuaire avec la méthode Naming.bind().

5- On exécute les clones sollicités.

6- On recherche un objet distant avec la méthode naming.lookup().

7- L'annuaire renvoie une référence de l'objet distant.

8- On invoque l'une des méthodes distantes Factorielle ().

9- On reçoit une valeur de retour.

### 5.3.5 Schéma récapitulatif

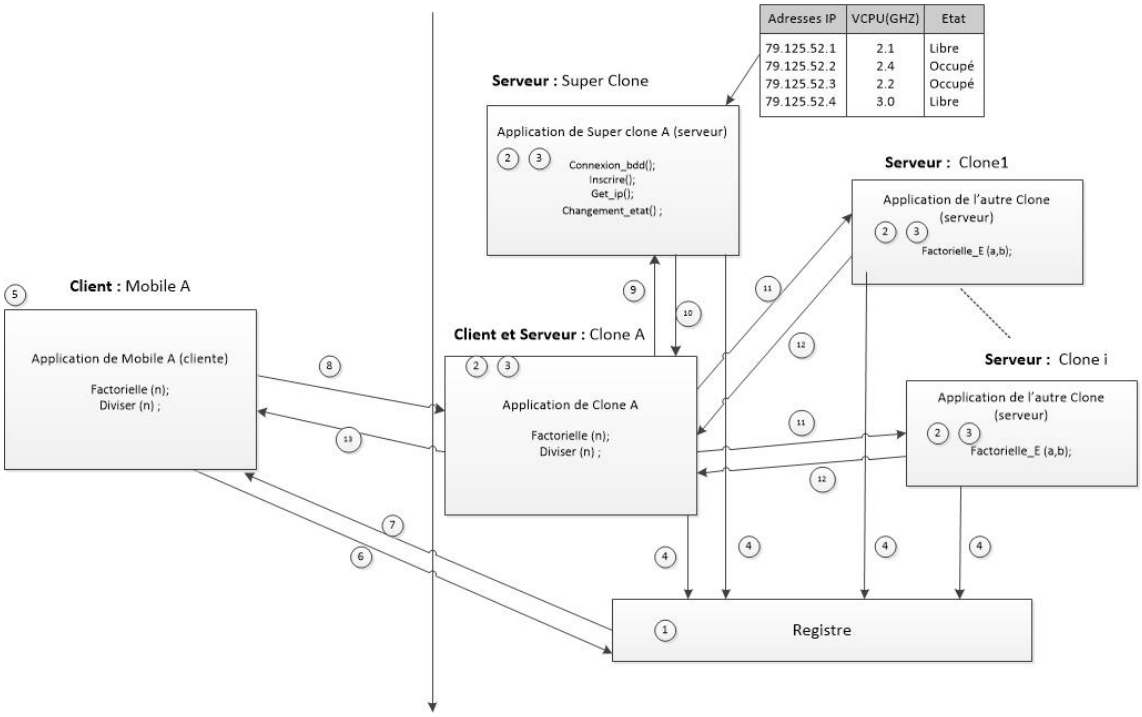


FIGURE 5.5 – Schéma récapitulatif.

1- On exécute rmiregistry.

2- On exécute les serveurs.

3- On crée les objets distants.

4- On les enregistre dans l'annuaire avec la méthode Naming.bind().

5- On exécute le mobile.

6- On recherche un objet distant avec la méthode naming.lookup().

7- L'annuaire renvoie une référence de l'objet distant.

8- On invoque la méthode distante Factorielle ().

9- On invoque la méthode distante Get_ip ().

10- On reçoit la list_ip.

11- On invoque la méthode distante Factorielle_E ().

12- On reçoit la valeur de retour.

13- On envoie la valeur finale au mobile.

## 5.4 Interfaces graphiques

### 5.4.1 Interface du Mobile

La figure 5.6 représente l'interface qui est visible par l'utilisateur du Mobile :

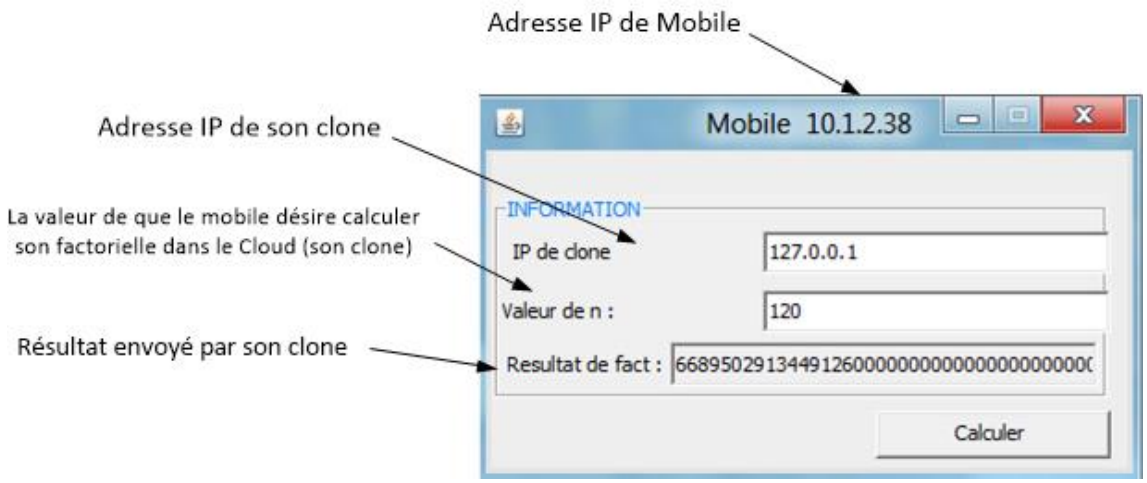


FIGURE 5.6 – Interface du Mobile.

### 5.4.2 Interface du Clone

La figure 5.7 représente l'interface du clone :

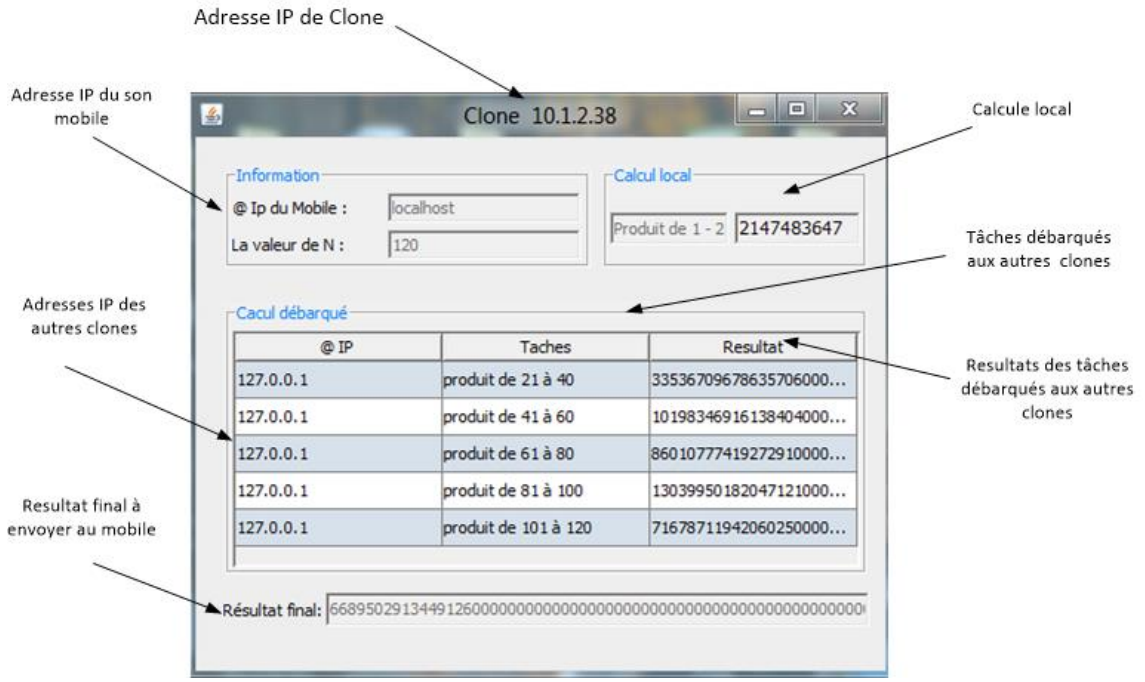


FIGURE 5.7 – Interface du Clone .

## 5.5 Conclusion

Dans ce chapitre nous avons implémenté le mécanisme de la collaboration en temps réel entre les acteurs de notre architecture, en permettant à un calcul distribué, pour cela nous avons fait appel à la procédure d'invocation à distance RMI après avoir montré les étapes nécessaires à la mise en place d'un service de type RMI.

# *Conclusion et perspectives*

Le Cloud Computing est une technologie en plein essor qui permet aux entreprises de disposer d'infrastructures et de progiciels directement en ligne sur Internet. Nous distinguons les diférents types de Cloud possibles avec l'IaaS pour les infrastructures techniques, le PaaS pour les infrastructures habillées avec des outils de middleware comme les bases de données par exemple, et le SaaS pour les services logiciels. Ces trois types peuvent se déployer sous quatre formes de topologies différentes : le Cloud public pour déporter en ligne, le Cloud privé pour l'utilisation des concepts du Cloud en interne à l'organisation, le Cloud hybride pour l'utilisation commune du public et du privé, et enfin le mode communautaire pour des entreprises géographiquement proches ou avec un intérêt communs.

Avec le fort développement des terminaux mobiles (Smartphones et tablettes) et leur adoption progressive dans le monde professionnel d'un côté leurs limites en terme de puissance de calcul, stockage et traitement de données d'un autre coté le Cloud Computing intègre le Mobile Computing afin de répondre aux demandes des utilisateurs.

Dans ce mémoire, nous nous sommes intéressés à la mise en œuvre d'une architecture pour le Cloud mobile en mode P2P qui réalise le débarquement du calcul des dispositifs mobiles vers leurs clones dans le Cloud. En permettant à une collaboration en temps réel entre les acteurs de l'architecture ainsi qu'un mécanisme de partage de fichiers, ceci en intégrant bien évidement un mécanisme de gestion de mobilité. Par conséquent des problèmes à l'instar de la durée de vie de la batterie, la latence, le temps d'exécution et la mobilité peuvent être résolus.

Bien que, par manque de temps ou par manque de ressource,les aspects qui vont suivirent ne sont pas développés d'avantages dans ce document, nous pensons qu'ils constituent des points de recherche intéressants pour notre travaille :

– La possibilité de rendre notre architecture totalement distribuées.

– Renforcer la sécurité du Cloud : L'un des problèmes majeur du Cloud Compu-

ting est la sécurité des données pour cela nous pensons que l'une des possibilités suivantes prouvent être utiles :

- Intégrité : Chaque utilisateur mobile du Cloud doit assurer l'intégrité de son information stockée sur le Cloud. Chaque accès qu'il fait doit être authentifié et vérifié. On propose de différentes approches par exemple chaque information stockée par chaque individu ou entreprise dans le Cloud est étiquetée ou initialisée à eux où ils sont les seules pour avoir accès (mouvement, mettre à jour ou suppression).
- Authentification : l'utilisation des mots de passe .. etc.

– Le déploiement de notre solution sur une vraie plateforme de Cloud comme Amazon et l'exploitation des machines virtuelles fournies par Iaas.

# Bibliographie


[1] *http ://www.yankeegroup.com/.*

[2] *http ://en.wikipedia.org/wiki/Cloud_ computing/.*

[3] *fr.wikipedia.org/wiki/International_ Business_ Machines/*

[4] *http ://www.techopedia.com/definition/26679/mobile-cloud-computing-mcc/.*

[5] *http ://www.cnetfrance.fr/special/smartphone-4000002598.htm/.*

[6] *http ://www.onehourtranslation.com/.*

[7] *http ://www.android.com/.*

[8] *http ://www.kazaa.com/.*

[9] *http ://aws.amazon.com/s3/.*

[10] *http ://rfc-gnutella.sourceforge.net/ .*

[11] *http ://freenet.sourceforge.net/.*

[12] *http ://setiathome.berkeley.edu/.*

[13] *http ://genomeathome.stanford.edu/.*

[14] *fr.wikipedia.org/wiki/Machine_ virtuelle/.*

[15] Audrey Coutty, *"Thierry Vonfelt L'évolution maitrisée vers le IaaS/PaaS"*. Livre blanc produit par EuroCloud France, Novembre 2011.

[16] Alexandra Carpen-Amarie, *"BlobSeer as a data-storage facility for Clouds : self-adaptation, integration, evaluation"*.THÈSE / ENS CACHAN – BRETAGNE sous le sceau de l'Université européenne de Bretagne pour obtenir le titre de DOCTEUR DE L'ÉCOLE NORMALE SUPÉRIEURE DE CACHAN Mention : Informatique École doctorale MATISSE, version 2.0, le 27 février 2012.

[17] Hoang T. Dinh, Chonho Lee, Dusit Niyato, and Ping Wang , *"Survey of Mobile Cloud Computing : Architecture, Applications, and Approaches"* OI : 10.1002/wcm.1203 2011.

[18] W. Tsai, X. Sun, and J. Balasooriya, *"Service-Oriented Cloud Computing Architecture"*.In Proceedings of the 7th International Conference on Information Technology : New Generations (ITNG), pp. 684-689, July 2010.

[19] JR. Kakerow, *"Low power design methodologies for mobile communication"*.In Proceedings of IEEE International Conference on Computer Design : VLSI in Computers and Processors, pp. 8, January 2003.

[20] Paulson,*"Low-Power Chips for High-Powered Handhelds"*.In Computer (Volume :36 , Issue : 1 ) ,2003 .

[21] J. W. Davis , *"EPower benchmark strategy for systems employing power management"*.In Proceedings of the IEEE International Symposium on Electronics and the Environment, pp. 117, August 2002.

[22] R. N. Mayo and P. Ranganathan , *"Energy Consumption in Mobile Devices : Why Future Systems Need RequirementsAware Energy Scale-Down"*.In Proceedings of the Workshop on Power-Aware Computing Systems, October 2003.

[23] J. Douceur and J. S. Donath , *"The sybil attack"*. In Proceedings for the 1st International Workshop on Peer-to- Peer Systems, pages 251-260, Cambridge, MA, USA, Mar. 2002.

[24] Mahadev Satyanarayanan ,Paramvir Bahl ,Ramón Cáceres And Nigel Davies , *"The Case for VM-Based Cloudlets in Mobile Computing"* ,In Pervasive Computing, IEEE (Volume :8 , Issue : 4 )2009 .

[25] P. Zou, C. Wang, Z. Liu, and D. Bao, *"Phosphor : A Cloud Based DRM Scheme with Sim Card"*.," In Proceedings of the 12th International Asia-Pacific on Web Conference (APWEB), pp. 459, June 2010.

[26] Yang, T. Pan, and J. Shen, *"On 3G Mobile E-commerce Platform Based on Cloud Computing"*.In Proceedings of the 3rd IEEE International Conference on Ubi-Media Computing (U-Media), pp. 198 - 201, August 2010.

[27] Z. Leina, P. Tiejun, and Y. Guoqing , *"PResearch of Mobile Security Solution for Fourth Party Logistics"*.In Proceedings of the 6th International Conference on Semantics Knowledge and Grid (SKG), pp. 383 - 386, January 2011

[28] X.Chen, J.Liu, J.Han and H.Xu, *"Primary Exploration of Mobile Learning Mode under a Cloud Computing Environmen"*.In Proceedings of the International Conference

on E-Health Networking, Digital Ecosystems and Technologies (EDT), vol. 2, pp. 484 -487, June 2010.

[29] H. Gao and Y. Zhai , *"Design of Cloud Computing Based on Mobile Learning"*.In Proceedings of the 3rd International Symposium on Knowledge Acquisition and Modeling (KAM), pp. 293 - 242, November 2010.

[30] Jian Li *"Study on the Development of Mobile Learning Promoted by Cloud Computing"*.In Proceedings of the 2nd International Conference on Information Engineering and Computer Science (ICIECS), pp. 1, December 2010.

[31] W. Zhao, Y. Sun, and L. Dai , *"Improving computer basis teaching through mobile communication and cloud computing technology"*.In Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), vol. 1, pp. 452 - 454, September 2010.

[32] C. Yin, B. David, and R. Chalon , *"Use your mobile computing devices to learn - Contextual mobile learning system design and case studies"*.In Proceedings of the 2nd IEEE International Conference on Computer Science and Information Technology (ICCSIT), pp. 440,September 2009.

[33] R. Ferzli and I. Khalife , *"Mobile cloud computing educational tool for image/video processing algorithms"*.In Digital Signal Processing Workshop and IEEE Signal Processing Education Workshop (DSP/SPE), pp. 529, March 2011.

[34] U. Varshney, *"Pervasive healthcare and wireless health monitoring"*.Journal on Mobile Networks and Applications, vol. 12, no. 2-3,pp. 113 - 127, March 2007.

[35] Z. Li, C. Wang, and R. Xu, *"Computation offloading to save energy on handheld devices : a partition scheme"*.In Proceedings of the 2001 international conference on Compilers, architecture, and synthesis for embedded systems (CASES), pp. 238 - 246, November 2001.

[36] Cuervo, A. Balasubramanian, Dae-ki Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl , *"MAUI : Making Smartphones Last Longer with Code offload"*.In Proceedings of the 8th International Conference on Mobile systems, applications, and services, pp. 49-62, June 2010.

[37] S. Wang and S. Dey , *"Rendering Adaptation to Address Communication and Computation Constraints In Cloud Mobile Gaming"*.in IEEE Global Telecommunications Conference (GLOBECOM), pp. 1-6, January 2011.

[38] H. Li and X-S. Hua , *"Melog : mobile experience sharing through automatic multimedia blogging"*.In Proceedings of the 2010 ACM multimedia workshop on Mobile cloud media computing (MCMC), pp. 19-24, 2010.

[39] Z. Ye, X. Chen, and Z. Li, *"Video based mobile location search with large set of SIFT points in cloud"*.In Proceedings of the 2010 ACM multimedia workshop on Mobile cloud media computing (MCMC), pp. 25-30, 2010.

[40] D. S. Milojicic, V. Kalogeraki and R. Lukose , *"Peer to Peer Computing"* .IN HP Laboratories, 2002.

[41] I. Stoica, R. Morris, F. Kaashoek, H. Balakrishnan *"Chord : A Scalable Peer-to-peer Lookup Service for Internet Applications. "* In Proceedings of ACM SIGCOMM, 2001.

[42] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker, *"A Scalable Content-Addressable Network"*In Proceedings of ACM SIGCOMM, 2001 .

[43] F. Panzieri, O. Babaoglu, V. Ghini, S. Ferretti, and M. Marzolla,*" Distributed computing"* .in the 21st century : Some aspects of cloud computing. Technical Report UBLCS-2011-03,Department of Computer Science, University of Bologna, Italy, May 2011.

[44] D. P. Anderson, *"Boinc : A system for public-resource computing and storage"* . In Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, GRID '04, pages 4–10,Washington, DC, USA, 2004. IEEE Computer Society.

[45] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen, *" Gossip-based peer sampling"* . ACM Trans. Comput. Syst., 25(3), 2007.

[46] M. Jelasity, A. Montresor, and O. Babaoglu, *"The bootstrapping service."* In Proceedings of the 26th IEEE International Conference/Workshops on Distributed Computing Systems, ICDCSW'06, pages 11–,Washington, DC, USA, 2006. IEEE Computer Society.

[47] M. Jelasity and A.-M. Kermarrec, *"Ordered slicing of very large-scale overlay networks"* . In Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing, pages 117–124,Washington, DC, USA, 2006. IEEE Computer Society.

[48] M. Jelasity, A. Montresor, and O¨ . Babaoglu *"Gossip-based aggregation in large dynamic networks"* . ACM Trans. Comput. Syst., 23(3) :219–252, 2005.

[49] M.Jelasity and O.Babaoglu, *"T-Man : Gossip-based Overlay Topology Managemen"* .In 3rd Int. Workshop on Engineering Self-Organising Applications (ESOA'05) ,2005.

[50] E. Marinelli, *" Hyrax :cloud computing on mobile devices using MapReduce", Masters Thesis, Carnegie Mellon University. 2009* .

*[51] J. Flinn, S. Park and M. Satyanarayanan,* "Balancing performance, energy, and quality in pervasive computing" *In Proceedings of the 22nd International Conference on Distributed Computing Systems"*. IN IEEE, 2002, pp. 217-226. .

[52] R Balan, M. Satyanarayanan, S. Park and T. Okoshi *"Tactics-based remote execution for mobile computing"*. In Proceedings of the 1st International Conference on Mobile Systems, Applications and Services, ACM, 2003, pp. 273-286 .

[53] J. Dean and S. Ghemawat, *"MapReduce : simplified data processing on large clusters"*.IN Communications of the ACM 51 (2008) 107-113.

[54] G. Huerta-Canepa and D. Lee , *"A virtual cloud computing provider for mobile devices"*. In Proceedings of the 1st ACM Workshop on Mobile Cloud Computing Services : Social Networks and Beyond, MCS'10, ACM, New York, NY, USA, 2010, pp. 6 :1-6 :5.

[55] L. Deboosere, P. Simoens, J.D. Wachter, B. Vankeirsbilck, F.D. Turck,B. Dhoedt and P. Demeester, *"Grid design for mobile thin client computing, Future Generation Computer Systems"*, 27 (2011) pp 681-693 .

[56] G. Portokalidis, P. Homburg, K. Anagnostakis and H.Bos, *"Paranoid android : versatile protection for smartphones"* .In Proc of ACSAC '10, 2010.

[57] E. Chen and M. Itoh *"Virtual smartphone over ip"* In Proc. of IEEE WoWMoM '10, 2010 .

[58] B. G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, *"CloneCloud : elastic execution between mobile device and cloud"* . In Proc of EuroSys '11,2011..

[59] M.Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, *"The case for vm-based cloudlets in mobile computing"* .Pervasive Computing, IEEE,8(4) :14 –23, oct.-dec. 2009.

[60] S.Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang , *"Thinkair : Dynamic resource allocation and parallel execution in the cloud for mobile code offloading "*. In Prod. of IEEE INFOCOM 2012, 2012.

[61] K. Kumar and Y. H. Lu, *"Cloud computing for mobile users : Can offloading computation save energy ?"* .In IEEE Computer, 43(4) :51–56, April 2010.

[62] A. Miettinen and J. Nurminen, *"Energy efin Aciency of mobile clients in cloud computing"* In Proc of HotCloud 2010.

[63] Y. Wen, W. Zhang, and H. Luo, *"Energy-optimal mobile application execution : Taming resource-poor mobile devices with cloud clones"*. In Proc of IEEE INFOCOM 2012, 2012.

[64] S.Kosta, V.Perta,J.Stefa,P.Hui and A.Mei, *"Clone2Clone (C2C) : Enable Peer-to-PeerNetworking of Smartphones on the Cloud"*. TR-SK032012AM ,Mars 2012.

[65] M. Kristensen,*"Scavenger : transparent development of efficient cyber foraging applications"* In Proceedings of the IEEE International Conference on Pervasive Computing and Communications, PerCom.

[66] I. Akyildiz, J. McNair, J. Ho, H. Uzunalioglu and W. Wang, *"Mobility management in next-generation wireless systems"*.In Proceedings of the IEEE 87 (1999) 1347-1384.

[67] I. Constandache, X. Bao, M. Azizyan and RR Choudhury ,*"Did you see bob ? : human localization using mobile phones"*. In Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking, MobiCom'10, ACM, New York, NY, USA, 2010, pp. 149-160.

[68] M.Chitra and RSD.Wahida Banu ,*"Towards robust and reliable groupe communication in ad hoc mobile environment "* .In Journal of Theoretical and Applied Information Technology 2005.

**Résumé**

Un environnement typique de Cloud Computing se caractérise par la mise en réseau de plusieurs dizaines de milliers de serveurs. Une fois déployés, ces centres de données connaissent un environnement de matériel et de réseau relativement stable. Les applications mobiles comme les Smartphones offrent des ressources de détection, de stockage de la mémoire et des traitements considérables, qui sont portables. La vision pour le Cloud Computing pourrait favoriser des modèles plus souples, permettant aux utilisateurs généraux, y compris les applications mobiles, de participer dans le Cloud comme des fournisseurs et des consommateurs de ressources. Si le Cloud est pour le traitement ce qu'est l'Internet pour les données, la possibilité pour les ordinateurs portables de participer à la prestation de services doit être étudiée. Par conséquent, ces concepts soulèvent un certain nombre de problématiques sur les clients et les fournisseurs que nous soulèverons : La manière dont elle le fournisseur garanti une exploitation des services à des performances élevées ? Les problèmes liés à la mobilité lors de l'utilisation des dispositifs qu'un client peut rencontrer et comment y remédier ? Comment réduire le temps d'exécution lorsque l'utilisateur désire débarquer le calcul au niveau de Cloud, ainsi que la consommation d'énergie ? Nous décrirons enfin les solutions qu'il est possible de mettre en place dans un futur proche.

**Mots clés :**Cloud Computing, Mobilité .

**Summary**

A typical cloud computing environment is characterized by networking of several tens of thousands of servers. Once they are deployed, these data centers know an environment of hardware and network relatively stable. The mobiles applications as smartphones offer resources detecting, storage of memory and considerable treatments, which are portable. The vision for the Cloud Computing could lead to more flexible models for general users, including mobile applications, participate in the Cloud as suppliers and consumers of resources. If the cloud is for treat then Internet is for data, the possibility for laptops to participate in the services should be investigated. This concept raises a number of client and supplier issues that we will raise : How providers guarantee services operating with high performance ? The problems related to the mobility when using devices that can meet customer and how to fix it ? How to reduce the execution time when the user needs to debark calculating in the finally, we describe the solutions that can be implemented in the near future.

**Keywords :** Cloud Computing, Mobility