**République Algérienne Démocratique et Populaire**
**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**

**Université A/Mira de Béjaïa**
**Faculté des Sciences Exactes**
**Département d'Informatique**


*Mémoire de fin de cycle*
*en vue d'obtention du diplôme de Master en informatique*
*Option : Réseaux et Systèmes Distribués*

**THEME**

---

# Optimisation des Architectures P2P-SIP pour la Téléphonie IP.

---

**Présenté par :**

$M^{lle}$ ***OUAKKOUCHE*** **Amanda**

**Devant le jury composé de :**

| | | |
|---|---|---|
| **Président** | $D^r$. ***A.ALOUI*** | **Maitre de conférences, U.A.M.Béjaia** |
| **Examinateur** | $M$. ***A.AKILAL*** | **Maitre assistant, U.A.M.Béjaia** |
| **Examinatrice** | $M^{me}$. ***N.HALFOUNE*** | **Maitre assistante, U.A.M.Béjaia** |
| **Encadreur** | $D^r$. ***M.AMAD*** | **Maitre de conférences, U.A.M.Béjaia** |

Promotion 2015/2016

# Dédicaces

# Remerciements

*En préambule à ce travail, je remercie Dieu tout puissant et miséricordieux, qui m'a donné la force et la patience d'accomplir ce modeste travail.*

*Je souhaite adresser mes remerciements les plus sincères à mes parents qui tout au long de ce travail, m'ont apporté leur précieux soutien ainsi que leur encouragements.*

*Je tiens à remercier sincèrement mon encadrant* ***M.AMAD Mourad*** *qui, s'est toujours montré à l'écoute et disponible tout au long de la réalisation de ce mémoire, ainsi pour l'inspiration, l'aide et le temps qu'il a bien voulu me consacrer.*

*Mes vifs remerciements vont également aux membres du jury qui ont accepté d'examiner mon travail et de l'enrichir par leurs suggestions.*

*Mes sincères reconnaissances à tous mes enseignants pour les efforts fournis durant tout notre cursus.*

*Enfin, je tiens également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.*

# Table des matières

# Acronymes

**IP** : **I**nternet **P**rotocol

**P2P** : **P**eer **T**o **P**eer

**DHT** : **D**istributed **H**ash **T**able

**RON** : **R**esilient **O**verlay **N**etwork

**OSI** : **O**pen **S**ystems **I**nterconnection

**TTL** : **T**ime **T**o **L**ive

**ID** : **ID**entifier

**SHA** : **S**ecure **H**ash **A**lgorithm

**VoIP** : **V**oice **o**ver **I**nternet **P**rotocol

**ToIP** : **T**elephony **o**ver**I**nternet **P**rotocol

**SIP** : **S**ession **I**nitiation **P**rotocol

**MGCP** : **M**edia **G**ateway **C**ontrol **P**rotocol

**SCCP** : **S**kinny **C**lient **C**ontrol **P**rotocol

**PSTN** : **P**ublic **S**witched **T**elephone **N**etwork

**ITU** : **I**nternational **T**elecommunications **U**nion

**TCP** : **T**ransmission **C**ontrol **P**rotocol

**UDP** : **U**ser **D**atagram **P**rotocol

**RTP** : **R**eal-time **T**ransfer **P**rotocol

**RTCP** : **R**eal-time **T**ransfer **Control** **P**rotocol

**SDP** : **S**ession **D**escription **P**rotocol

**UAS** : **U**ser **A**gent **S**erver

**UAC** : **U**ser **A**gent **C**lient

**BNF** : **B**ackus **N**aur **F**orm

**TU** : **T**ranssaction **U**ser

**dSIP** : **d**istributed **S**ession **I**nitiation **P**rotocol

**QoS** : **Q**uality **o**f **S**ervice

**FIFO** : **F**irst **I**n **F**irst **O**ut

**MATLAB** : **MAT**rix **LAB**oratory

**IETF** : **I**nternet **E**ngineering **T**ask **F**orce

# Table des figures

# Liste des tableaux

# Liste des algorithmes

# Résumé

Les architectures P2P sont des systèmes complètement décentralisés, robustes, scalables et tolérants aux panes. Nous nous intéressant aux architectures P2P-SIP à base de réseaux P2P structurés, particulièrement, Chord car elles sont plus performantes en terme de coût de recherche.

Les chercheurs ont proposé le protocole SIP étant un protocole de signalisation cohérent aux architectures P2P grâce à sa simplicité, son hétérogénéité, sa flexibilité, son extensibilité mais aussi parce qu'il est standardisé. SIP est un élément très important dans les architectures P2P-SIP pour la VoIP.

Dans ce projet, nous proposons une amélioration en terme lors de cout de localisation tout en considérant la priorité des appels d'urgence dans les architectures P2P-SIP structurées. Nous avons modélisé le réseau Chord en se basant sur un modèle mathématique qu'est les files d'attente M/M/1. Nous avons évalué les performances en utilisant le langage Matlab. L'évaluation de performance montre que les résultats sont globalement satisfaisants.


**Mot-Clés** : Réseaux P2P, Protocole Chord,SIP, P2P-SIP, M/M/1, routage avec priorité.


# Abstract

The P2P architectures are a decentralized, robustess, scalable and fault tolerant systems. We are interessted by a P2P-SIP architectures based on structured P2P networks, particullary, Chord networks because they are more performant in terme of the cost lookup.

The reaserchers have proposed a SIP protocol as a coherent signalisation protocol to a P2P architectures thanks to it smplicity, heterogeniety, flexibility, extensiability, but also it's standardized. SIP is a fundamental element for VoIP based application such as P2P-SIP architecture.

In this work, we propose an amélioartion in terme of cost lookup by considering emergency call priority in a structured P2P-SIP. We have modelized a Chord network by drawing on mathematic model which is M/M/1 queue. We have evaluate the performances by using the Matlab language. Performance evaluation show that resultats are globally satisfactory.


**Keywords** : Networking P2P, Chord protocol, SIP, P2P-SIP, M/M/1, Priority routing.

# Introduction générale

L'internet a révolutionné le monde des ordinateurs et de la communication. Il a poussé les informaticiens à développer de différentes applications en mode client/serveur durant plusieurs années. Une application particulière de par son architecture est apparue à la fin des années 90 adoptant le mode P2P. Celui-ci a donné naissance à plusieurs applications qui se basent sur de nouvelles architectures qui en dérivent, telles que : Imesh, Bi torrent...

L'internet a non seulement incité les développeurs à diversifier les architectures et applications mais aussi élaborer de multiples protocoles de communication : messageries instantanés, partage de fichiers et la transmission de la voix sur IP.

La téléphonie sur IP (*Voice over IP*) faisait partie des "fantasmes" des premiers internautes. En effet, elle peut, à la fois, transporté la voix mais aussi proposer tous les services classiques de la téléphonie aux clients.

Skype c'est l'une des applications les plus utilisées dans le monde, elle constitue l'un des meilleurs exemples de la VoIP, c'est une application qui se base sur une architecture complètement décentralisée en utilisant ses propres protocoles de signalisation et VoIP.

Le protocole SIP est un protocole de signalisation standardisé de la couche application, il joue les rôles d'un client et d'un serveur. Il est facile à implémenter et c'est surtout un protocole indépendant des protocoles des couches inférieures du modèle TCP/IP, c'est ce qui explique son importance et son utilisation aujourd'hui dans des applications informatiques.

L'efficacité de la VoIP dans les architectures P2P et la standardisation du protocole SIP ont poussé les chercheurs de L'IETF à combiner les architectures P2P avec le protocole SIP ce qui à donner naissance à des architectures P2P-SIP.

Plusieurs travaux et architectures (*SoSimple, dSIP...*) ont été proposés dans le but de pouvoir les utiliser pour développer des applications fonctionnelles, optimales, moins couteuses et sécurisées qui satisferont les clients et internautes.

Malgré que plusieurs études ont été élaborées, les systèmes P2P-SIP rencontrent quelques limites et lacunes telles que : la tolérance aux pannes, la surcharge de la bande passante et des nœuds, la sécurité et enfin la qualité de service.

Dans le présent travail, nous nous intéressons aux problèmes liées à la qualité de service à travers un paramètre que l'on juge primordial qu'est le temps (*délai*).

Certes, les architectures P2P-SIP structurées optimisent les coûts de la recherche, réduisent le nombre de sauts lors de la localisation d'un utilisateur mais dans le cas où un utilisateur effectue un appel d'urgence, cet appel sera traité exactement comme tous les autres appels alors qu'il devrait passer en premier.

L'objectif de notre projet est de traiter la priorité des appels d'urgence dans le cas des architectures P2P-SIP structurées afin d'optimiser les délais de transmission. Pour se faire, nous avons organisé notre mémoire en cinq chapitres :

**Le premier chapitre** « **Architectures P2P** »est consacré à l'état de l'art sur les architectures P2P qui sera exploité au cours de notre étude.

**Le deuxième chapitre** « **Protocole SIP** » est une présentation détaillée du protocole de signalisation SIP et de son fonctionnement pour appuyer son adoption dans les architectures P2P-SIP.

**Le troisième chapitre** « **Architectures P2P-SIP** »» nous a permis de classifier les architectures proposées, de les détailler et de déterminer leurs inconvénients et limites.

**Le quatrième chapitre** « **Routage prioritaire dans les architectures P2P-SIP structurées**» est consacré à l'exhibition de la solution proposée, le fonctionnement, la modélisation et enfin les différents algorithmes de notre solution.

**Le cinquième chapitre** « **Aspects techniques et mise-en-œuvre** »est réservé pour l'évaluation des performances de notre solution en utilisant la simulation avec Matlab, et c'est ainsi que nous allons soumettre les résultats de la simulation obtenus.

Chapitre 1

# Architecture P2P

## 1.1 Introduction

Avec la continuité de son développement, l'internet a donné naissance à une nouvelle architecture appelée « Client/serveur » où les données sont centralisées et accessibles par les clients. Néanmoins, à la fin des années 90 l'architecture « Peer to Peer » s'annonce comme rivale, avec succès, et fait son explosion à partir de 2000.

Ce chapitre est un état de l'art sur les réseaux Peer to Peer.

## 1.2 Notion de Client/serveur

C'est un réseau composé de postes (*clients*) qui se connectent au serveur distribuant les applications et les fichiers. Cette architecture est utilisée en grandes et moyennes entreprises. Les particuliers et les petites entreprises peuvent se satisfaire d'une architecture de type poste à poste, moins lourde à mettre en œuvre, mais dans laquelle la sécurité est moindre [**1**].

La figure ci-après est un exemple du modèle client/serveur.

Figure 1.1 – *Aperçu d'une architecture client/serveur.*

### 1.2.1 Avantages et inconvénients du modèle client/serveur

#### 1.2.1.1 Avantages

L'architecture client/serveur est particulièrement recommandée pour des réseaux nécessitant un grand niveau de fiabilité, ses principaux atouts sont :

- des ressources centralisées :étant donné que le serveur est au centre du réseau, il peut gérer des ressources communes à tous les utilisateurs, comme par exemple une base de données centralisée, afin d'éviter les problèmes de redondance et de contradiction.
- une meilleure sécurité :car le nombre de points d'entrée permettant l'accès aux données est moins important.
- une administration au niveau serveur : les clients ayant peu d'importance dans ce modèle, ont moins besoin d'être administrés.
- un réseau évolutif :grâce à cette architecture, il est possible de supprimer ou rajouter des clients sans perturber le fonctionnement global du réseau et sans modification majeure.

#### 1.2.1.2 Inconvénients

L'architecture client/serveur a tout de même quelques lacunes parmi lesquelles :

- un coût élevé dû à la technicité du serveur ;
- un maillon faible :le serveur est le seul maillon faible du réseau client/serveur, étant donné que tout le réseau est architecturé autour de lui, le serveur a une grande tolérance aux pannes.

## 1.3 Notion de Peer to Peer

Le modèle pair à pair appelé également «égal à égal» est un modèle de communication décentralisé dans lequel chaque partie possède les mêmes capacités et peut initier une session de communication. Contrairement au modèle client/serveur, dans lequel le client effectue une demande de service et le serveur répond à la requête, le modèle de réseau P2P permet à chaque nœud de fonctionner à la fois comme un client et un serveur [**13**].

La figure 1.2 est une illustartion d'une architecture "Peer to Peer".

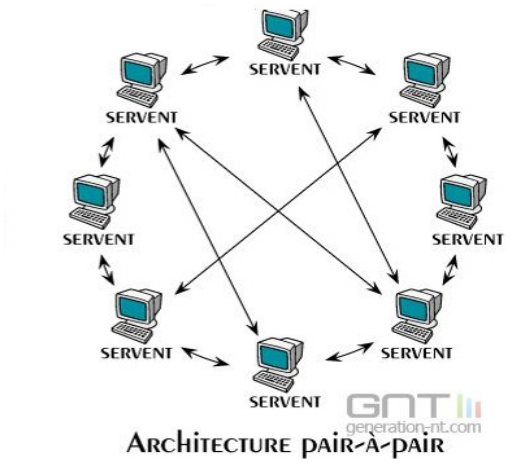


FIGURE 1.2 – *Architecture P2P.*

## 1.4 Avantages de l'architecture Peer to Peer

- **Anonymat** :le modèle égale à égale préserve la notion de l'anonymat (*tenir compte des opérations non-identifiées*) ;
- **Dynamisme** :la topologie peut s'accroître, comme elle peut bien se réduire (*des nœuds peuvent rejoindre ou quitter le réseau sans aucune énigme*) ;
- **Scalabilité** :le passage à l'échelle est supporté ;
- **Décentralisation** :les systèmes « paire à paire » ne dépendent pas des serveurs centrales. En d'autres termes, les nœuds s'échangent les informations sans passer pour autant par des serveurs centrales.
- **Tolérance aux pannes** :lorsqu'une panne (*logicielle ou matérielle*) se présente, le système reste toujours fiable et fonctionnel et ceci est grâce à la décentralisation.

## 1.5 Réseaux Overlay

Une méthode de routage de niveau applicatif entre les nœuds d'une application distribuée. Les paquets voyagent d'un nœud à un autre routés par la couche réseau (*OSI niveau 3*), mais la décision de routage (*i.e., quel nœud est la destination à chaque saut*) est faite explicitement au niveau applicatif de chaque nœud. C'est la raison pour laquelle les réseaux overlays sont aussi appelés Réseau Applicatifs. Un exemple solide est le Résilient Overlay Network (*RON, Réseau Overlay Robuste*) : un groupe de nœuds multi-home [1] assurent la continuité de la communication même en présence des pannes dans le réseau, en utilisant les chemins sur l'overlay de niveau applicatif pour éviter les réseaux ou les liens en panne. Les réseaux overlay sont représentés d'habitude par un graphe dirigé $(X, U)$, où $X$ est l'ensemble des nœuds terminaux dans l'overlay et $U$ est l'ensemble des liens de niveau applicatif entre les nœuds dans $X$. Le graphe $(X, U)$ doit être connexe [**20**].

## 1.6 Classification des architectures P2P

Dans l'architecture P2P, on distingue deux grandes classes :

### 1.6.1 Réseaux P2P non-structurés

Dans un réseau P2P non-structuré, il n'y a pas de définition stricte de la façon de construire la topologie du réseau. L'exemple d'applications des réseaux non-structurés de base est Gnutella [25]. Dans le modèle non-structuré, quand un pair veut trouver une information particulière, il n'a aucune idée de l'emplacement des données désirées. Par conséquent, une demande doit être inondée à tous les pairs de l'overlay. La demande a un temps limite, dit le time-to- live (TTL). Elle peut arriver au nœud correspondant comme elle peut rester dans le réseau jusqu'à ce que son délai expire [**8**].

L'avantage de ce modèle est qu'il est moins complexe que le modèle structuré. Néanmoins, il entraîne des retards de recherche et le réseau consomme inutilement la bande passante. En outre, il ne garantit pas de réponses positives à une recherche donnée.

1. consiste, pour un réseau informatique, à être connecté à plusieurs fournisseurs d'accès à Internet afin d'améliorer la fiabilité de la connexion à Internet.

Dans ce type de réseaux, on distingue trois sous-classes :

#### 1.6.1.1 Architecture centralisée

Cette architecture se compose d'un serveur unique, dont le but est de recenser les fichiers proposés par les clients. Contrairement à l'architecture client/serveur, ce serveur ne dispose pas des fichiers. En revanche, il dispose principalement de deux informations : celles des fichiers (*nom, taille...*) et celles des utilisateurs (*@IP...*). Le réseau Peer to Peer le plus connu du grand public est sans doute Napster [2]. Son excentricité réside dans le fait d'utiliser l'architecture centralisée.

la figure 1.3 est une illustration d'une architecture centralisée :

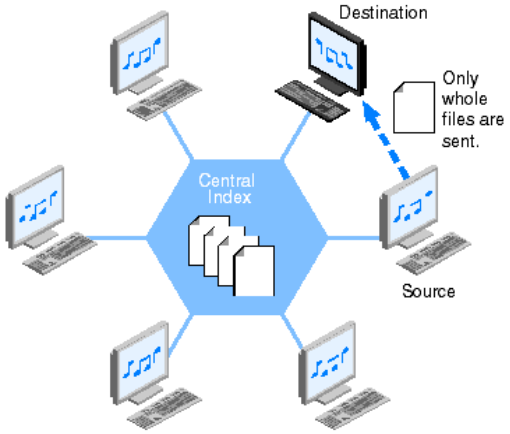


FIGURE 1.3 – *Architecture centralisée (NAPSTER).*

Lorsqu'un utilisateur souhaite télécharger un fichier, il effectue sa recherche comme dans un moteur de recherche classique : il lance une requête en écrivant un mot-clé, le serveur lui renvoie la liste des utilisateurs connectés qui partagent ce fichier. Par conséquent, il suffit de cliquer sur un des titres pour se connecter avec la machine correspondante et entamer le transfert.

a. **Avantages**
   - Facile à administrer, contrôler et mettre en œuvre ;
   - réduit les coûts de la recherche ;
   - tolérance aux fautes.

b. **Limites**
   - Le réseau reste dépendant du serveur ;
   - saturation de la bande passante ;
   - problème de sécurité et de robustesse ;
   - ne garantis pas l'anonymat.

#### 1.6.1.2 Architecture décentralisée

Contrairement à l'architecture centralisée, celle-ci préserve l'anonymat des pairs. Dès lors, pour avoir accès à une information ; il faut connaitre la topologie du réseau (*ceci se fait avec des*

2. www.Napster.com

*broadcasts*), chercher l'information sur tous les nœuds du réseau pour enfin avoir une réponse des nœuds correspondants aux critères.

Parmi d'autres exemples, le protocole GNUTELLA **citeGnutella** est un aperçu de l'architecture décentralisée.

La figure 1.4 un exemple concret d'un modèle décentralisé :

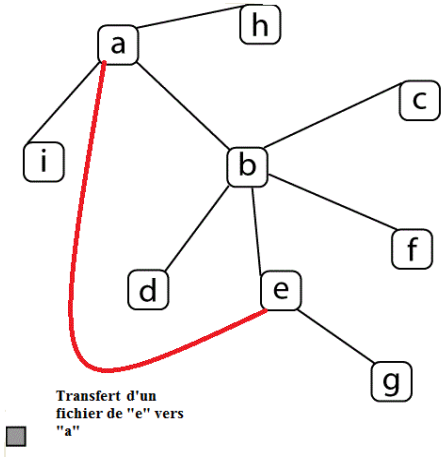


FIGURE 1.4 – *Architecture décentralisée (GNUTELLA).*

Le nœud « a » se connecte sur le réseau et il doit connaitre la topologie du réseau, pour se faire il va broadcaster une demande d'identification aux nœuds voisins (*i, h et b*) qui feront de même à leur tour (*b vers d, e, f, c et ainsi de suite*). Lorsque la demande est bien reçue, les nœuds vont envoyer leurs identifiants à « a », ainsi ce dernier crée un annuaire. Pour chercher un fichier, « a » effectue une demande ; l'envoie à ses voisins qui vont à leur tour l'envoyer à leurs voisins et ainsi de suite. Lorsqu'un nœud donné « e », par exemple, répond aux critères de la demande, il lui envoie une liste des fichiers correspondants, un index des fichiers sera alors crée sur « a », ce dernier choisit le fichier souhaité qui sera transféré de « e » vers « a » directement.

a. **Avantages**
   - Dynamisme : le volume du réseau peut se réduire ou s'accroître ;
   - garantit l'anonymat ;
   - disponibilité du réseau.

b. **Limites**
   - Pollution du réseau à cause des séries de broadcasts.

#### 1.6.1.3 Architecture hybride (*super-noeud*)

Cette architecture est une combinaison des deux précédentes (*centralisée et décentralisée*). Dans ce cas de figure, les machines clientes sont reliées à des serveurs selon l'architecture centralisée, or les serveurs sont interconnectés suivant l'architecture décentralisée.

Emule [3] est l'un des sites utilisant l'architecture hybride.

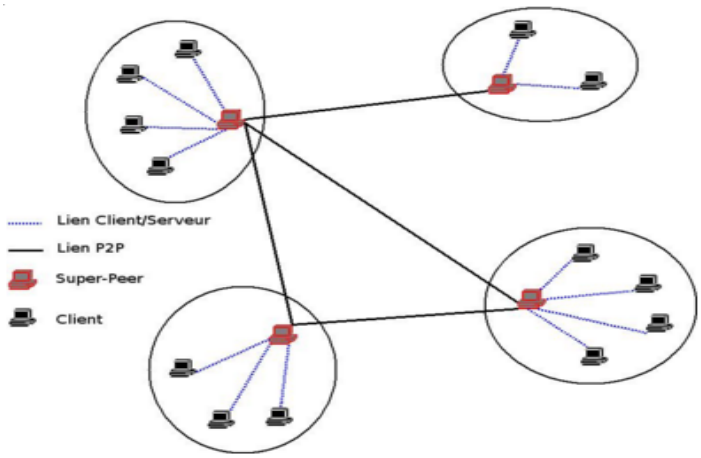


FIGURE 1.5 – *Architecture hybride (Emule).*

Lorsqu'un client souhaite retrouver un fichier, la recherche s'effectue comme dans l'architecture centralisée (*il la demande auprès du serveur auquel il est connecté*). Mais dans le cas où le fichier ne se trouve pas chez les clients du même serveur, ce dernier exécute les mêmes opérations qui s'effectuent dans l'architecture décentralisée auprès des serveurs voisins du même réseau.

a. **Avantages**
   - Pas de problèmes de bande passante.

b. **Limites**
   - L'anonymat n'est plus assuré ;
   - difficile à mettre en œuvre.

### 1.6.2 Réseaux P2P structurés

Un réseau P2P structuré est un réseau overlay auto-organisé avec une fonction de routage sophistiquée. Plusieurs réseaux P2P structurés utilisent un algorithme de routage basé sur la table de hachage distribuée (*DHT*) pour la gestion des ressources distribuées à travers le réseau overlay. Les DHTs offrent de simples interfaces" PUT/GET " pour stocker et récupérer des données.

3. www.emule.com

Lorsqu'un nœud rejoint l'overlay, un identifiant unique (*Node-ID*) lui sera attribué. De même, lorsque l'utilisateur souhaite stocker des données, un numéro unique (*ID-Key*) sera alors attribué à ces données bien spécifiques. Ce ID-Key détermine à quel nœud les ressources doivent être stockées. En d'autres termes, les données seront stockées au niveau du nœud dont l'identifiant est plus proche de l'ID-Key. De plus, les données peuvent également être répliquées aux nœuds voisins pour augmenter la fiabilité. Pour être en mesure d'effectuer une recherche, chaque nœud doit maintenir une table de routage (*table de hachage*) qui contient une plage d'ID-Keys qui sera alors transmise aux nœuds responsables. Lorsqu'un nœud reçoit une demande de recherche, il va rechercher l'ID-Key dans sa table de hachage. Si le nœud ne dispose pas d'informations sur cette demande de clé spécifique, il fera parvenir plus de nœuds dans sa table de hachage [**8**].

Plusieurs recherches sur les algorithmes DHT sont en cours. De différentes techniques de recherches et de gestions sont utilisées dans le but de développer des systèmes distribués robustes et performants. Parmi les algorithmes les plus connus, on peut citer CHORD [**18**], PASTERY [**4**] et TAPESTRY [**6**] .

### Protocole TAPESTRY

Tapestry est un réseau overlay qui permet de situer des objets (*par un chemin court*) dans le réseau de manière distribuée. Un réseau Tapestry permet de router des messages d'un nœud vers n'importe quel autre de manière distribuée. Le routage utilise, dans une version plus évoluée, le principe du hashed-suffixrouting présenté originallement par Plaxton [4], Rajaraman [5] et Richa [6]. Une caractéristique intéressante d'un réseau overlay du type Tapestry est la capacité de trouver un objet dans le réseau de manière efficace (*en terme de nombre de sauts de routage*). Il a été montré, de plus, que dans le cas d'un objet répliqué, une requête était menée au réplica le plus proche de l'émetteur de cette requête.

Tapestry apparaît donc comme un système très intéressant (*par sa nature distribuée*) pour les échanges des messages d'updates entre serveurs ou les requêtes des clients.

### Protocole CHORD

1. **Définition**

   C'est un système de recherche décentralisé et robuste pour les grands systèmes de P2P.

   Le protocole Chord supporte une seule opération :l'attribution des clés, il fait correspondre une clé à un nœud. Un nœud peut être responsable de stocker une valeur associée à une clé. Chord utilise une variante d'un hachage conforme pour assigner des clés à des nœuds Chord. Le hachage à tendance d'équilibrer la charge, depuis chaque nœud reçoit à peu près le même nombre de clés, et comporte relativement peu de mouvement en terme de clés lorsque des nœuds rejoignent et quittent le système [**18**].

4. www.Plaxton.com
5. www.Rajaraman.com
6. www.Richa.com

Pour un réseau Chord de $N$ nœuds, les recherches peuvent être résolues avec certitude en O(log $(N)$) messages et chaque noeud ne doit maintenir O(log $(N)$) des informations de routage à d'autres nœuds. Par ailleurs, il effectue des recherches performantes basées sur une clé et il résout la recherche en retournant l'adresse IP du noeud contenant la clé [**4**].

2. **Propriétés du protocol Chord**
   - **Décentralisation** :dans un système "peer to peer" utilisant Chord , il n'existe pas de serveur central. Chaque noeud est de même importance que tout autre noeud. Par conséquent, le système est très robuste, car il ne dispose pas d'un point de défaillance unique.
   - **Disponibilité** :les fonctions du protocole sont très bonnes, même si le système est dans un état de changement continu : Malgré les défaillances majeures du réseau sous-adjacent et malgré l'adhésion d'un grand nombre de nœuds, le nœud responsable d'une clé peut être toujours trouvé.
   - **Scalabilité** :le coût d'une recherche Chord n'augmente qu'en logarithmique du nombre de noeuds dans le système, donc Chord peut être utilisé pour de très grands systèmes.
   - **Charge équilibrée** :chord utilise une fonction de hachage conforme afin d'assigner des clés aux nœuds. Par conséquent, les clés sont réparties uniformément sur les nœuds.
   - **Nommage flexible** :chord impose : pas de contraintes sur la structure des clés, afin que l'utilisateur bénéficie d'une grande quantité de flexibilité dans la dénomination des données.
3. **Fonctionnement du protocole Chord**

Le protocole Chord utilise principalement la topologie en anneau et spécifie comment les clés sont attribuées aux nœuds, comment les localiser et comment un nœud peut rejoindre le réseau ? Nous allons découvrir ceci dans ce qui suit :

a) **Hachage cohérent**

Le hachage cohérent attribue à chaque noeud et clé un identifiant de $m$-bit en utilisant une fonction de hachage de base telle que SHA-1 [**3**]. L'identificateur d'un noeud est choisi en hachant son adresse IP, tandis que l'identificateur d'une clé est produit en hachant seulement cette dérnière [**18**].

b) **Attribution des clés aux noeuds**

L'attribution des clés se fait grâce au hachage cohérent. Ce dernier assigne des clés aux nœuds comme une suite. Les identifiants sont ordonnés dans un cercle appelé cercle des identifiants.
Une clé $k$ est assigné au 1$^{er}$ nœud dont l'identifiant est égal ou suit(succède) $k$ dans l'espace des identifiants. Ce nœud est appelé nœud successeur de la clé $k$, noté successeur($k$) [**18**].

La figure 1.6 nous présente un exemple de cette opération :

Supposons que le cercle des identifiants est de $m$ =3. Soient $n0$,$n1$ et $n3$ trois noeuds. Soient $k1$,$k2$ et $k6$ trois clés.

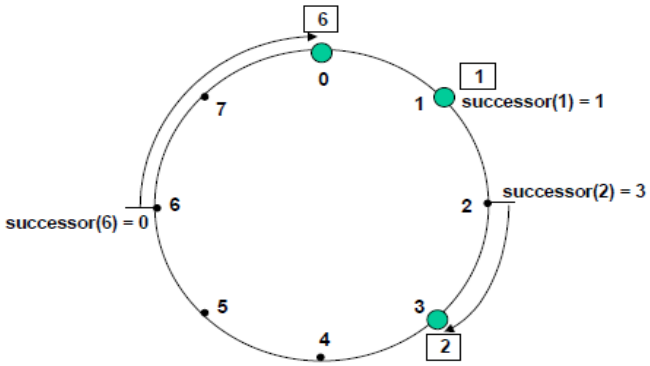


FIGURE 1.6 – *Exemple d'un cercle des identifiants.*

Successeur (1)=1, on remarque alors que la clé $k1$ est au noeud $n1$.
Successeur (2)=2, dans ce cas de figure l'identifiant $id2$ ne dispose pas de nœud donc on prend $n3$ : le noeud qui succède $id2$, et on lui attribue $k2$.
Successeur (6)=1, d'où $k6$ est à $n0$.

### c) Localisation des clés

La localisation des clés se fait grâce à une table appelée table de raccourcis (*finger table*).La table de raccourcis se crée comme suit :

supposons que n est un nœud qui maintient une table de routage de $m$ entrées appelée table de raccourcis. La $i^e$ entrée dans la table du nœud n contient l'identifiant du $1^{er}$ nœud, s, qui succède le nœud n dans le cercle des identifiants, i.e., s= successeur$(n+2^{i-1})$.
On appelle le nœud s le $i^e$ raccourcis du nœud n, noté : n.finger[i].node.

- Une entrée de la table de raccourcis inclut l'identifiant Chord et l'adresse IP attribuée au nœud.
- Le $1^{er}$ raccourcis de n est un successeur immédiat dans le cercle des identifiants.

Le schéma 1.7 illustre la table de raccourcis et son utilisation pour la localisation des clés :
La table de raccourcis de n1 se pointe sur les nœuds successeur de l'identifiant : $(1+2^0)\text{mod } 2^3 = 2$, $(1+2^1)\text{mod } 2^3 = 3$, $(1+2^2)\text{mod } 2^3 = 5$ réspectivement.
le successeur de $id2$ est n3 (n3 est le $1^{er}$ noeud après $id2$),le successeur de $id3$ est n3 et enfin le successeurs de $id5$ est n0.

Supposons que le noeud n1 souhaite trouver l'emplacement de la clé $k6$. La recherche s'effectue comme suite :

- n1 effectue la requête : quer($k6$).
- n1 cherche dans sa table de raccourcis la valeur la plus proche de $k$=6
- Nous remarquons que la valeur la plus proche de $k6$ est $id5$ dans la même ligne et dans la colonne "succ" nous trouvons 0, qui signifie n0, et dans ce cas on déduit que $k6$ se trouve dans le noeud n0.

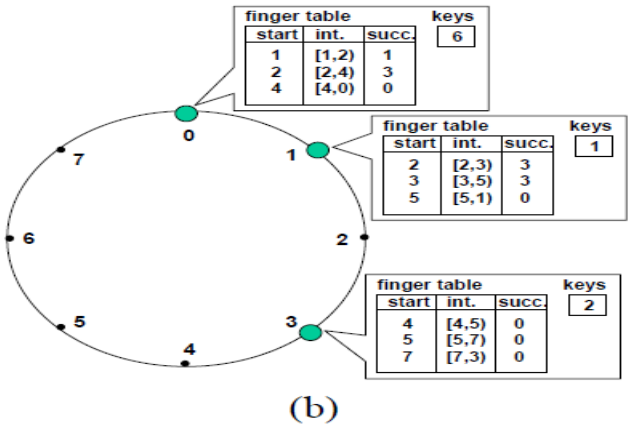


FIGURE 1.7 – *Exemple d'une table de raccoucis.*

d) **Reoindre le réseaux**

Dans un réseau dynamique, les nœuds rejoignent et quittent le réseau en permanence. Pour que le fonctionnement soit correcte Chord doit préserver deux propriétés :

- Chaque successeur d'un nœud doit être maintenu correctement ;
- pour chaque clé $K$, le nœud successeur($k$) doit être responsable de $k$.

Afin de préserver ce qui est cité précédemment, Chord doit assurer trois tâches :

– Initialiser le prédécesseur et les raccourcis du nœud n ;
– mettre à jour les raccourcis et prédécesseurs des nœuds existants ;
– informer la couche supérieure de sorte qu'il peut transférer l'état (*ex, les valeurs*) associé aux clés dont le nœud n est résponsable.

## 1.7 Limites des architectures Peer to Peer

L'architecture pair à pair ne dispose pas seulement d'avantages mais aussi de quelques lacunes :

- **virus** :une très grande quantité de virus circule dans l'architecture peer to peer car elle est difficile à contrôler ;
- **freeloading** :les freeloaders prennent sans donner, ils bénéficient des ressources partagées sans partager les leurs. Une grande quantité de bande passante est dépensée alors pour acheminer les ressources jusqu'à eux sans qu'il y ait de contribution au reste du réseau en retour ;
- **atteinte à la vie privée** :les adresses IP des utilisateurs peuvent être récupérées lorsque le logiciel est centralisé ;
- **pollution des réseaux** :une poignée de sociétés indépendantes développe en effet de nouvelles technologies destinées à polluer les réseaux.

## 1.8 Conclusion

Dans ce chapitre introductif, nous avons présenté un panorama général de ce qu'une architecture Peer to Peer, ses caractéristiques ainsi que ses avantages.

Nous avons ensuite rappelé les classes de l'architecture P2P dont la première consiste en réseau non-structuré qui se répartie également en trois sous classes : centralisée, décentralisée et hybride. La seconde se résume en réseau structuré : nous avons invoqué dans ce cas les protocoles Chord et Tapestry.

Il s'avère que P2P offre divers services dans le domaine de la communication et le partage des ressources (*logicielle ou matérielle*) mais il dispose de quelques lacunes qui incitent les chercheurs à effectuer de nombreuses recherches, afin de l'améliorer.

Le chapitre qui suit est un état des connaissances sur le protocole SIP.

Chapitre 2

# Protocole SIP

## 2.1 Introduction

L'apparition de la nouvelle technologie la voix sur Ip (VoIp [**24**]) revient à la combinaison de deux anciennes technologies : la téléphonie et l'internet.

La mise en œuvre de la Voix sur IP nécessite la conception et la création de nouveaux protocoles tels que les protocoles de transport et de signalisation. Parmi les protocoles de transport on distingue : les protocoles RTP et RTCP. Parmi les protocoles de signalisation on cite : le protocole H.323 [**21**], MGCP [**21**], SCCP [**21**] et enfin SIP [**21**]. Ce dernier sera alors l'objet d'étude de ce chapitre.

## 2.2 VoIP

La communication vocale effectuée en utilisant le protocole Internet (*IP*) pour le transport est connue comme Voice over Internet Protocol (*VoIP*). Les réseaux téléphoniques traditionnels, connus sous le nom Réseaux téléphoniques publics commutés (*PSTN1*) utilisent la commutation de circuits. Dans la commutation de circuits, les ressources sont réservées le long de la voie de communication pour l'ensemble de la durée d'appel. En revanche, le protocole Internet (*IP*) utilise la commutation de paquets. Dans la commutation par paquet, les informations sont transmises numériquement dans un ou plusieurs paquets.
Les paquets connaissent leur destination, et peuvent y arriver par des chemins différents [**24**].

## 2.3 Protocole H.323

Le standard H.323 [**21**] est normalisé en 1996, il fournit une base pour la communication utilisant de l'audio, vidéo et les données à travers le réseau IP. En se conformant au standard H.323, les produits et les applications multimédias peuvent interopérer. H.323 est une recommandation venant de l'ITU (*International Télécommunications Union*).
La norme H.323 est la principale concurrente de SIP pour la transmission du son, de données et de la vidéo en temps réel sur le réseau IP [**22**].

## 2.4 Protocole SIP

Session Initiation Protocol (*SIP*) est un protocole de signalisation en concert avec d'autres applications multimédia telles que la téléphonie vocale/vidéo (*VoIP*), la messagerie instantanée, et les jeux en ligne.

SIP permet aux utilisateurs terminaux (*appelés agents utilisateurs*) de négocier les caractérisations de la session partagée. Il leur permet d'établir, de modifier, et de terminer en temps réel des sessions multimédias entre les participants individuels ou multiples.

Le concept de SIP .est de soutenir la configuration des sessions entre des agents utilisateurs sur toutes les plates-formes, n'importe où sur Internet. SIP est un protocole de contrôle de la couche application qui fonctionne indépendamment des protocoles sous-jacents. En particulier, SIP peut être utilisé au-dessus de n'importe quel protocole de la couche transport comme TCP [**7**] ou UDP [**7**]. Pour répondre à l'utilisateur et la mobilité des services, SIP crée un autre espace de noms qui mappe entre l'identifiant de l'utilisateur et son emplacement actuel. L'avantage de séparer l'identifiant et l'emplacement du contact est que les utilisateurs n'a besoin de se rappeler des identifiants indépendamment de leur emplacement sur le réseau, quelques soit les terminaux qu'ils utilisent [**5**].

## 2.5 Comparaison entre SIP et H.323

Nous résumons les différences entre les deux protocoles de signalisation SIP et H.323 dans le tableau suivant :

| **Protocole H.323** | **Protocole SIP** |
|---|---|
| Signalisation sur les réseaux RNIS et RTC | Communication sur SIP |
| Approche télécommunication | Approche ToIP |
| Supporte les conférences vidéo et de données | Pas de restriction sur le média transporté |
| Messages binaires | messages encodés |
| Intéraction avec un grand nombre de protocoles | Intéraction avec un petit nombre de protocoles |

TABLE 2.1 – *Comparaison des deux protocoles SIP et H.323*

Le protocole H.323 était largement implémenté. En revanche, vu que les atouts majeurs de SIP sont : sa simplicité et sa grande flexibilité c'est ce qui a motivé l'implémentation de SIP.

## 2.6 Entités du protocole SIP

Le protocole SIP dispose des entités qui intéragissent entre elles afin de garantir les services SIP, on distingue :

**Agent utilisateur**

Un user agent est un élément final de l'architecture SIP. Il est constitué d'un user Agent client (*UAC*) et d'un user agent server(*UAS*) [**19**].

- **UAC** : est une application cliente qui envoie des requêtes SIP.
- **UAS** : est une application serveur qui dialogue avec l'utilisateur quand une requête SIP est reçue. Elle renvoie la réponse de l'utilisateur.

#### Serveur proxy

Les serveurs proxy sont chargés du routage d'une session SIP. Quand un serveur proxy reçoit une requête SIP, il l'a transmet au prochain serveur proxy sur le chemin qui mène vers l'user agent concerné par la requête si celui-ci est directement connecté au serveur proxy [**19**].

#### Enregistreur

Il s'agit d'un serveur qui accepte les requêtes SIP « REGISTER.SIP », il dispose de la fonction d'enregistrement d'utilisateurs. L'utilisateur indique par un message REGISTER émis au Registrar, l'adresse où il est joignable (*ex. adresse IP*). L'enregistreur met alors à jour une base de données de localisation.
L'enregistreur est une fonction associée à un Proxy server ou à un Redirect server. Un utilisateur peut s'enregistrer sur différents UAs SIP ; dans ce cas, l'appel lui sera délivré sur l'ensemble de ces UAs [**19**].

#### Serveur de redirection

Il s'agit d'un serveur qui accepte des requêtes SIP, traduit l'adresse SIP de destination en une ou plusieurs adresses réseau et les retourne au client. Contrairement au Proxy server, le serveur de redirection n'achemine pas de requêtes SIP. Dans le cas d'un renvoi d'appel, le Proxy server a la capacité de traduire le numéro de l'appelé dans le message SIP reçu en un numéro de renvoi d'appel et d'acheminer ensuite l'appel à cette nouvelle destination, et ce, de façon transparente pour le client d'origine. Pour le même service, le Redirect server retourne le nouveau numéro (*numéro de renvoi*) au client d'origine qui se charge d'établir un appel vers cette nouvelle destination [**19**].

## 2.7 Adressage et nommage

SIP choisit l'Email comme adresse de la forme : "user@domain", " user@IP adresse", "numéro telephone@gateway". Le nom de domaine peut être le nom du hôte sur lequel l'utilisateur est logé. L'adresse de la forme "numéro telephone@gateway" désigne le numéro du téléphone PSTN [**22**]. A l'inverse des numéros de téléphone, les adresses SIP désignent une entité plutôt qu'un terminal. Cette différence fondamentale permet à celles-ci de représenter un individu, quelque soit sa localisation physique et également quelque soit le terminal à sa disposition [**22**].

## 2.8 Structure du protocole SIP

SIP est structuré comme un protocole en couches, ce qui signifie que son comportement est décrit en termes d'ensembles de stades de traitements très indépendants avec seulement un couplage lâche entre chaque stade. Le comportement du protocole est décrit en couches pour

les besoins de la présentation, permettant la description de fonctions communes à travers des éléments dans une seule section [**19**].Les quatre couches du protocole SIP sont :

### Couche Codage et Syntaxe

C'est la couche SIP la plus basic, dans ce cas le codage est spécifié en utilisant une grammaire Backus-Naur Form (*BNF*) augmentée.

### Couche Transport

Elle définit comment un client envoie des demandes et reçoit des réponses et comment un serveur reçoit des demandes et envoie des réponses sur le réseau.

### Couche Transaction

Les transactions constituent un composant fondamental de SIP. Une transaction est une demande envoyée par un client de transaction (utilisant la couche transport) à un serveur de transaction, avec toutes les réponses à cette demande renvoyées depuis le serveur de transaction au client. La couche de transaction traite les retransmissions de couche application, appariant les réponses aux demandes, et les temporisations de couche application. Toutes les tâches qu'accomplit un client d'agent d'utilisateur (*UAC, user agent client*) ont lieu en utilisant une série de transactions.

### Couche Utilisateur de Transaction

Chacune des entités de SIP, excepte les mandataires sans état, est un utilisateur de transaction. Lorsqu'un ($\underline{TU}$) souhaite envoyer une demande, il crée une instance de transaction de client et lui passe la demande avec l'adresse IP de destination, le port, et le transport, auxquels envoyer la demande. Un TU qui crée une transaction de client peut aussi l'annuler.

La figure 2.1 présente les quatre couches du protocole SIP :

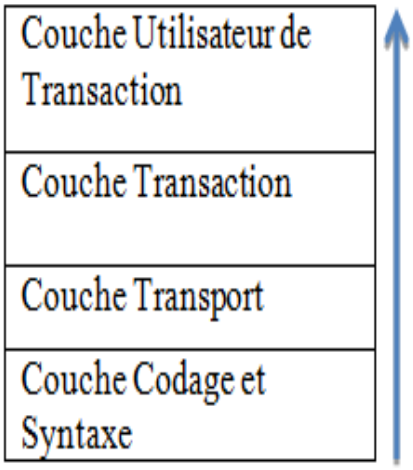


FIGURE 2.1 – *Les différentes couches du protocole SIP*

## 2.9 Pile du protocole SIP

La figure 2.2 présente les protocoles les plus utilisés en parallèle et en collaboration avec SIP pour la téléphonie sur IP. SIP est indépendant et peut être utilisé avec d'autres protocoles.

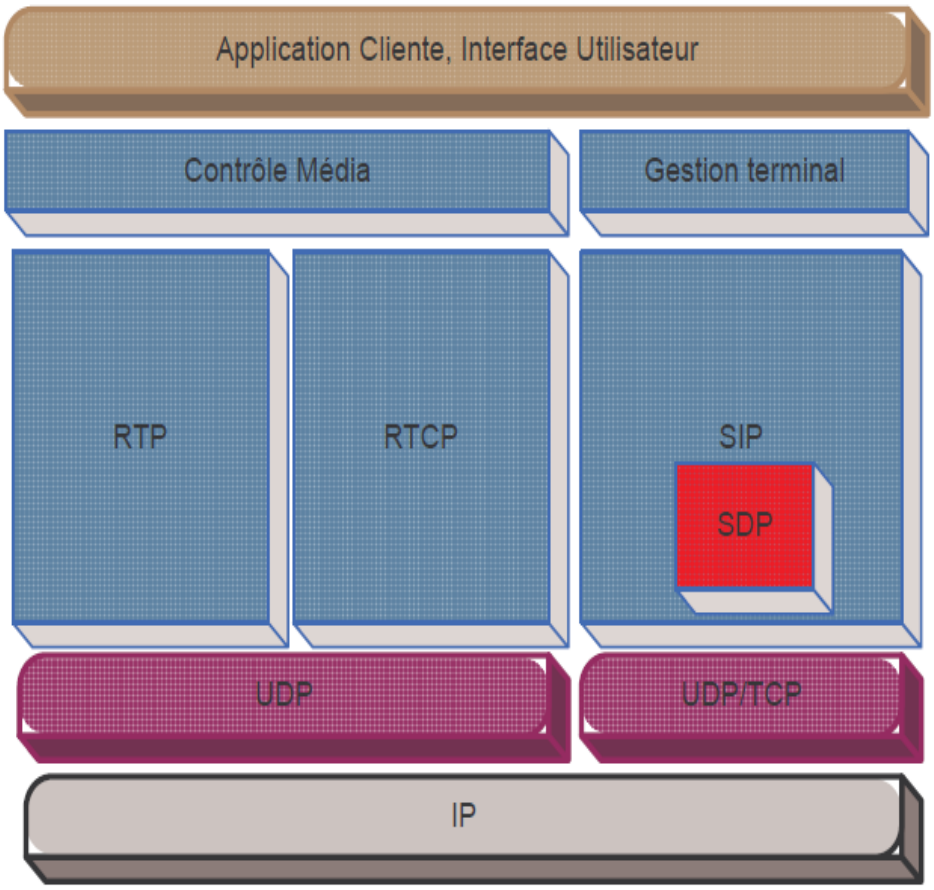


FIGURE 2.2 – *Pile du protocole SIP*

- **SIP** (*Session Initial Protocole*) : ce protocole permet de créer et de gérer les sessions entre différents participants pour s'échanger les données. Il est indépendant de la nature des données et des protocoles de transport. Ce protocole sert à établir une conversation téléphonique ou des conférences [**14**].
- **SDP** (*Session Description Protocol*) : le protocole SDP [7] est utilisé pour la négociation des caractéristiques d'une session. Il permet aux différents participants d'être compatibles entre eux. Grâce à SDP, les participants peuvent décider quel codec serra utilisé pour la voix et quel protocole de transport qui acheminera les médias (*RTP, par exemple*) [**14**].
- **RTP** (*Real-time Ttransfer Protocol*) : le protocole RTP [7] fournit des fonctions de transport utilisables par des applications utilisant des données en temps réel (*la voix, par exemple*) sur des réseaux IP [**7**]. RIP [**7**] ne gère pas la réservation des ressources et ne garantit pas la qualité du service.
- **RTCP** (*Real-time Tranfer Protocol*) : c'est un protocole RTP amélioré. En plus des fonctionnalités d' RTP , RTCP surveille le transfert et offre des fonctions de contrôle et d'identification. RTCP se base sur l'envoie périodique des paquets de contrôle à tous les participants d'une session, en utilisant le même mécanisme de transfert de données [14].

## 2.10 Aspects du protocole SIP

SIP fournit des primitives qui peuvent être utilisées pour mettre en œuvre différents services :
- **Localisation de l'utilisateur** :il permet de déterminer le système terminal qui doit être utilisé pour la communication.
- **Etablissement de l'appel** :il permet d'établir les paramètres de la communication entre l'appelant et l'appelé.
- **Disponibilité de l'utilisateur** : ce service permet de vérifier si le destinataire veut participer à la communication.
- **Moyen d'utilisation** : il permet de déterminer le média ayant les caractéristiques correspondantes.
- **Gestion de l'appel** : il permet le transfert et la terminaison de l'appel.

## 2.11 Services de sécurité

La vie privée et la sécurité sont des exigences obligatoires pour n'importe quel système de téléphonie. Le standard SIP définit des mécanismes de sécurité à utiliser pour protéger les messages de signalisation SIP et opérations médiatiques. Le protocole SIP fait appel aux mécanismes usuels de monde IP que de définir de nouveaux mécanisme, on distingue :

1. **confidentialité** : le chiffrement est necessaire afin d'assurer la confidentialité de données. Les méthodes de sécurité usuelles utilisées dans l'IP peuvent être utilisées dans la ToIP telles que : TLS, IPsec et SRTP.
2. **Authentification** : SIP fournit un mécanisme d'authentification simple basé sur l 'HTTP Digest directement intégrer dans l'en tête des messages SIP. A chaque requête d'un usager, le serveur SIP peut demander une authentification (*sauf pour ACK*) [**26**].

## 2.12 Requêtes et réponses du protocole SIP

SIP se fonde sur un modèle de transaction de demande/réponse du style http. Chaque transaction consiste en une demande qui implique une méthode ou fonction particulière sur le serveur, et au moins une réponse [**19**].

**Requêtes**

| **Requêtes** | **Définitions** |
|---|---|
| **INVITE** | Etablir une session entre les UAs. |
| **ACK** | Confirmation de la réception d'une réponse finale de « INVITE ». |
| **BYE** | Libération d'une session préalablement établie. |
| **REGISTER** | Indiquer au « Registrar » la correspondance entre l'adresse SIP et l'adresse de contact de l'UA. |
| **CANCEL** | Demande l'abandon d'un appel en cours mais n'a aucun effet sur un appel déjà accepté. |
| **OPTIONS** | Interroger les capacités et l'état d'un User agent ou d'un serveur. |

TABLE 2.2 – *Requêtes du protocole SIP*

Le protocole SIP a été étendu, quelques méthodes ont été rajoutées afin que son utilisation soit meilleure et efficace. Nous citons dans le tableau ci-dessous les méthodes rajoutées :

| **Requêtes** | **Définitions** |
|---|---|
| **SUBSCRIBE** | Demander une notification d'un événement. |
| **NOTIFY** | Transport de la notification de l'événement souscrit. |
| **PUBLISH** | Téléchargez l'état de présence à un serveur. |
| **MESSAGE** | Transport d'un corps de message instantané. |
| **AUPDATE** | Mettre à jour les informations de session. |
| **INFO** | Mi- appel de transport de signalisation. |
| **REFER** | Transfert d'un utilisateur à une URI. |

TABLE 2.3 – *Requêtes lors de l'extension de SIP*

L'exemple suivant est un bloc d'instructions que l'on effectue lors de l'utilisation de l'une des commandes précédentes. Nous prenons alors la requête « invite » pour illustrer cet exemple :

| INVITE sip :bob@biloxi.com SIP/2.0 |
|---|
| Via : SIP/2.0/UDP pc33.atlanta.com ;branch=z9hG4bK776asdhds |
| Max-Forwards : 70 |
| To : Bob ¡sip :bob@biloxi.com¿ |
| From : Alice ¡sip :alice@atlanta.com¿ ;tag=1928301774 |
| Call-ID : a84b4c76e66710@pc33.atlanta.com |
| CSeq : 314159 INVITE Contact : ¡sip :alice@pc33.atlanta.com¿ |
| Content-Type : application/sdp |
| Content-Length :142 |

TABLE 2.4 – *Exemple de la commande INVITE*

**Réponses**

Les auteurs de [11] ont récapitulé les réponses existantes dans le protocole SIP dans le tableau qui suit : L'exemple suivant est un bloc d'instructions que l'on exécute lors de l'utilisation

| **Réponses** | **Définitions** |
|---|---|
| **Classe 1xx** | Information, la requête a été reçue, et est en cours de traitement. |
| **Classe 2xx** | Succès, la requête a été reçue, comprise et acceptée. |
| **Classe 3xx** | Redirection, l'appel requiert d'autres traitements avant de pouvoir déterminer s'il peut être réalisé. |
| **Classe 4xx** | Erreur requête client, la requête ne peut pas être interprétée ou servie par le serveur. La requête doit être modifiée avant d'être renvoyée. |
| **Classe 5xx** | Erreur serveur, le serveur échoue dans le traitement d'une requête apparemment valide. |
| **Classe 6xx** | Echec global, la requête ne peut être traitée par aucun serveur. |

TABLE 2.5 – *Réponses du protocole SIP aux requêtes*

de l'une des réponses. Parmis elles on choisit la classe "1XX" :

| SIP/2.0 200 OK |
|---|
| Via : SIP/2.0/UDP proxy1.tn.com :5060 ; branch=z9hG4bK7745221 received = 192.190.132.21 |
| Via : SIP/2.0/UDP station1.tn.com :5060 ; branch=z9hG4bK776asdrgs |
| To : Mark Rich ; tag 22454 |
| From : Mary Taylor ;tag=21272 Call-Id : j1sv235bh8965ws |
| Cseq : 1 INVITE |
| Contact : sip :mark.rich@192.190.23.16 |

TABLE 2.6 – *Exemple d'une réponse*

## 2.13 Headers SIP

Nous remarquons dans l'exemple de la section précédente (2.4) qu'en plus des commandes question/ réponses nous trouvons des champs en-têtes tels que :

| En-tête SIP | Description |
|---|---|
| **VIA** | Il contient la version SIP et le protocole de transport qui sont en cours d'utilisation, ainsi que le nom d'hôte (ou adresse IP) et le port d'origine du message.<br>Il y a aussi un paramètre " branch ", qui est utilisé en tant qu'identificateur unique pour cette transaction. |
| **TO** | Il contient le nom et l'URI SIP du destinataire. |
| **FROM** | Il permet d'affiche le nom de l'expéditeur et l'URI SIP. Il existe également un paramètre appelé " tag" qui contient un pseudo-aléatoire pour chaque boîte de dialogue. |
| **CALL-ID** | C'est un identifiant unique d'une session. Toutes les transactions d'une même session utilisent un même CALL-ID. |
| **Max-FORWARDS** | Il indique le nombre maximal de sauts. Il sera décrémenté par chaque proxy qui transmet une demande. |
| **CSEC** | Il contient un numéro de séquence qui sera incrémenté après chaque requête successive. L'appelant et l'appelé maintiennent leurs propres numéros de séquence séparément. |
| **CONTACTE** | Contient un URI SIP ou d'autres informations telles que le courriel ou numéro de téléphone à partir duquel l'utilisateur souhaiterait être accessible. |
| **CONTENT-TYPE** | Dispose d'une description du corps du message. |
| **CONTENT-LENGTH** | Possède la taille du corps du message en octet. Si la valeur est à zéro, cela indique qu'il n'y a aucun corps de message pour cette demande. |

TABLE 2.7 – *En-tête du protocole SIP*

## 2.14 Fonctionnement du protocole SIP

Pour mieux comprendre le fonctionnement du protocole SIP et les interactions du système, nous donnons quelques exemples simples et concrets.
Dans ces exemples, nous prenons Amanda et Amel comme Users Agent.

### 2.14.1 Création d'une session

Amanda envoie la requête « Register » au serveur proxy (*l'enregistreur est inclut*) celui-ci la traite puis l'achemine à l'enregistreur du domaine «mult.dz ». Celui-ci effectue un lien entre l'adresse SIP d'Amanda et sa localisation actuelle (*l'adresse IP*) et envoie via le serveur proxy les mises à jour effectuées au service de localisation pour que ce dernier sauvegarde le contacte Amanda dans sa base de données pour qu'elle soit joignable par d'autres utilisateurs agent du réseau.
La figure ci-après illustre la création d'une session dans le protocole SIP :

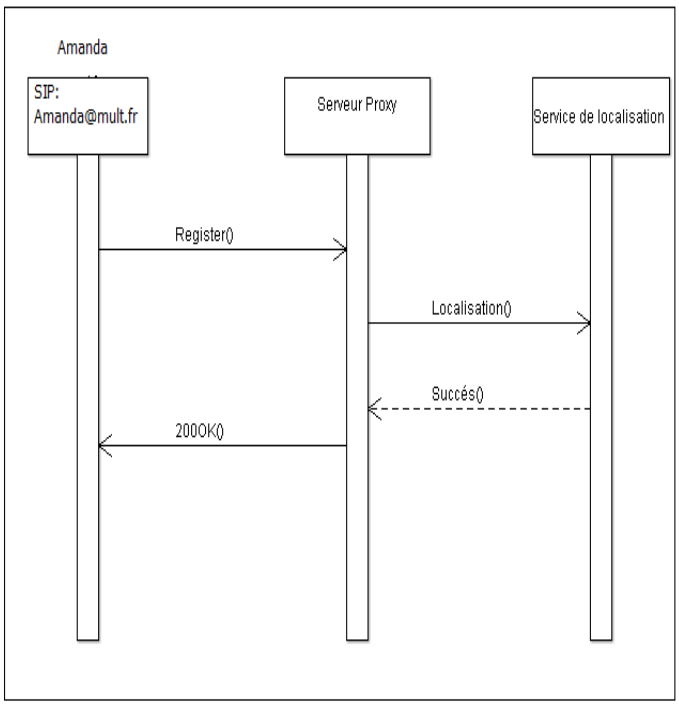


FIGURE 2.3 – *Création d'une session dans le protocole SIP*

### 2.14.2 Ouverture et fermeture d'une session

Lorsque Amel souhaite communiquer avec Amanda, elle lui envoie alors la requête « Invite » sachant que cette dernière est traitée et acheminée par les deux serveurs proxy d'Amel et d'Amanda respectivement. Une fois qu'Amanda reçoit cette requête, elle renvoie une réponse de classe 1xx comme quoi la requête est bien reçue puis envoie une autre réponse de la même classe pour lui confirmer que la requête d'Amel est en cours de traitement pour enfin lui répondre avec une réponse de classe 2xx pour lui confirmer que la requête a été acceptée.
Amel lui envoie à son une réponse finale « ACK », la session est alors ouverte et les deux Users Agent peuvent commencer à parler.
Amel souhaite fermer la session, elle envoie à Amanda la requête « BYE », à sa réception Amanda lui répond par un message 200 ok pour lui confirmer que la requête a été reçue, comprise est acceptée. Et donc la session est fermée. La figure ci-dessous est un schéma illustratif de l'ouverture et fermeture d'une session SIP :

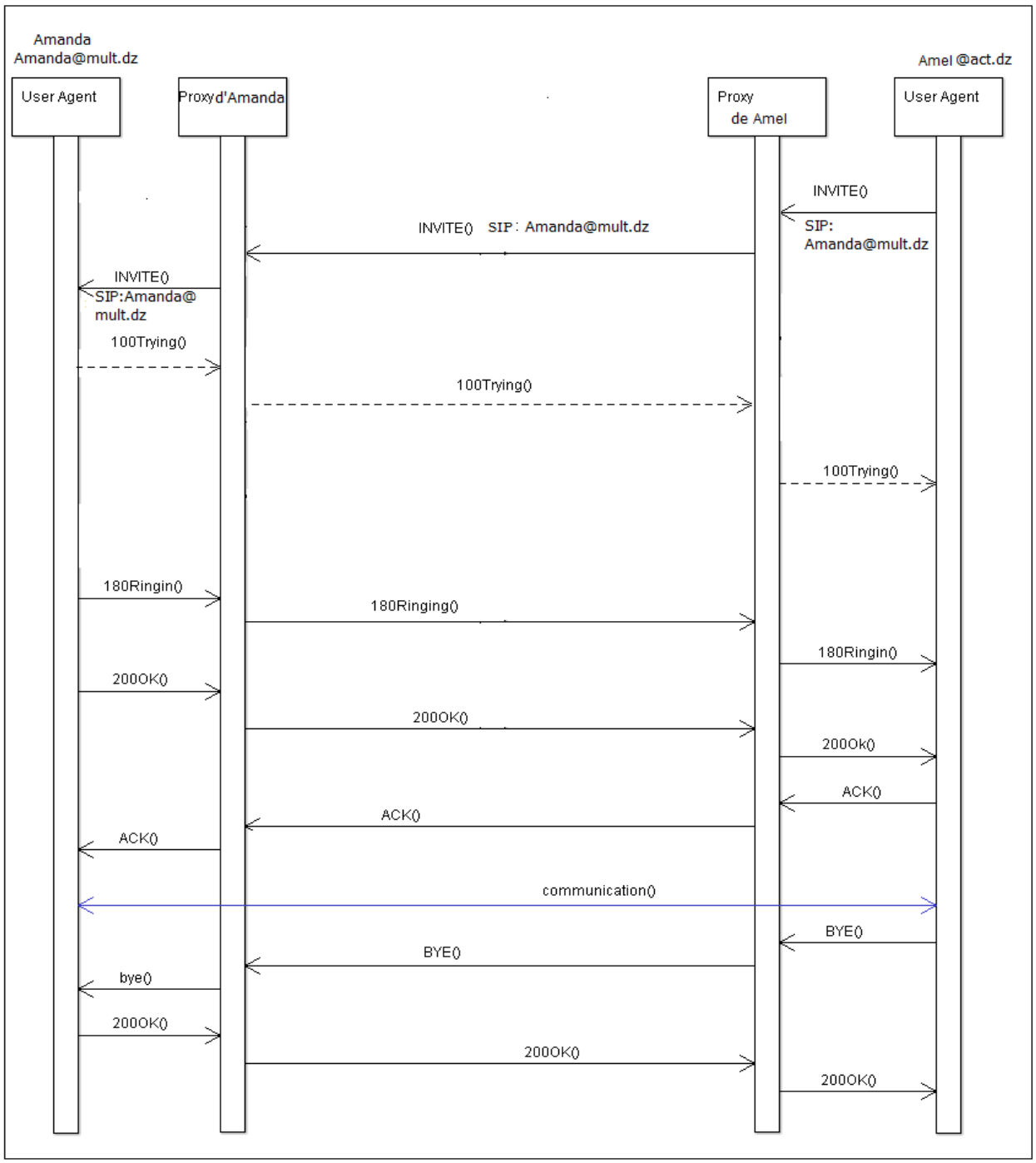


FIGURE 2.4 – *Ouverture et fermeture d'une session.*

### 2.14.3 Exemple lors d'une erreur

Amanda souhaite ouvrir une session de communication avec amel, elle envoie alors la requête « Invite ». Lors de son traitement, le serveur SIP lui renvoie une réponse de classe 4xx ce qui signifie que la requête d'Amanda est erronée (*problème d'authentification par exemple*) et elle devrait la modifier puis la renvoyer encore une autre fois.
La figure suivante est une représentation de cette erreur :

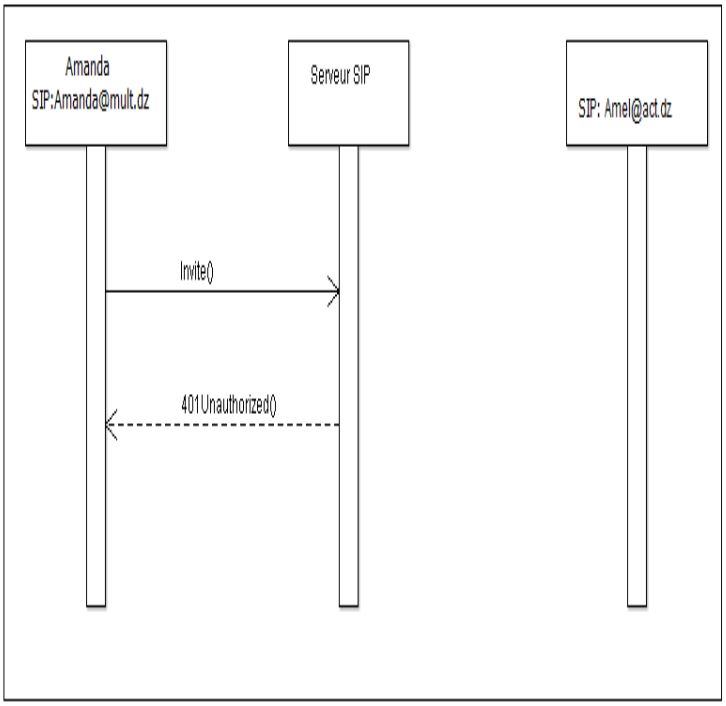


FIGURE 2.5 – *Exemple d'une erreur lors d'une ouverture d'une session*

### 2.14.4 Exemple avec la commande « Cancel »

Amanda souhaite communiquer avec Amel, elle lui envoie alors la requête « Invite ». amel lui répond alors avec deux réponses de classe 1xx : la première indiaue que le message a été bien reçu et la deuxième mentionne que le message a été bien compris et il est en cours de traitement.
Amel ne souhaite pas ouvrir cette session, elle annule l'appel en cours avec la commande « Cancel ».
La session n'a pas été alors ouverte, la figure qui suit est un exemple illustrant la réaction du système lors de l'utilisation de la méthode « Cancel» :

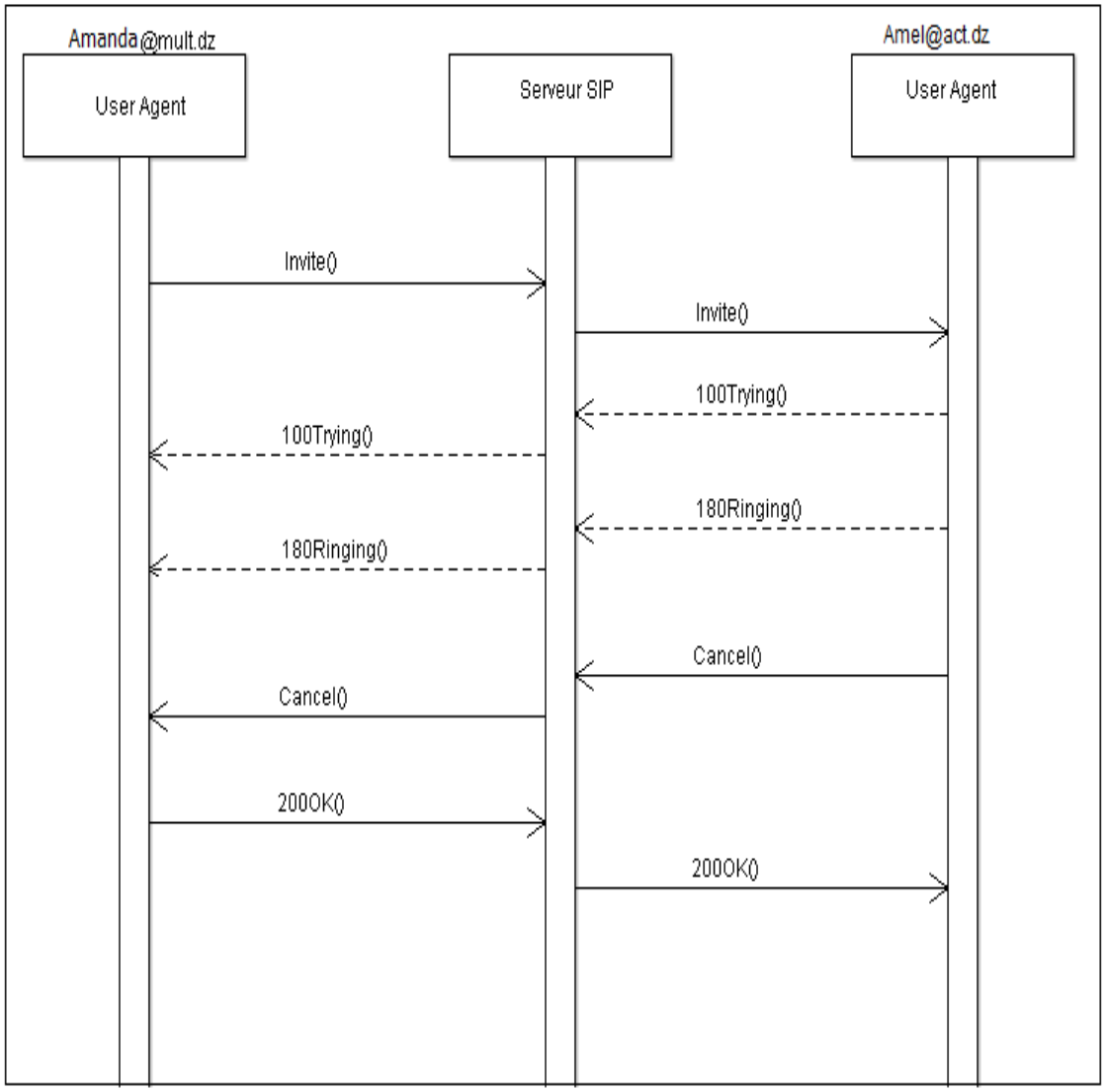


FIGURE 2.6 – *Exemple avec la méthode cancel.*

## 2.15 Conclusion

Dans ce chapitre, nous avons découvert en premier lieu le protocole SIP, ses composants ainsi que sa structure.
Nous avons par la suite, cité les protocoles les plus fréquemment utilisés avec le protocole SIP. Par ailleurs, nous avons présenté quelques aspects et services de sécurité qu'il offre.
Vu que SIP est un poids-léger en terme de questions/réponses, nous avons alors résumé dans des tableaux les différentes requêtes utilisées, les champs d'en-tête utilisés par ses requêtes ainsi que les différentes réponses retournées.
Pour conclure, nous avons donné des exemples concrets et très simples qui permettront de comprendre au mieux le fonctionnement du protocole SIP.

Dans le chapitre suivant, nous allons présenter les architectures P2P-SIP d'une manière détaillée, ainsi que les différentes recherches et travaux proposés.

Chapitre 3

# Architecture P2P-SIP pour les Applications VoIP

## 3.1 Introduction

L'une des plus grandes motivations des architectures P2P-SIP est sans doute Skype [1], c'est une application qui se base sur les architectures P2P non-structurelles totalement distribuées en utilisant le protocole KAZAA [**23**] pour la gestion des nœuds de cette architecture. Cependant, Skype utilise ses propres protocoles de signalisation et VoIP.

Les chercheurs ont alors pensé à exploiter les architectures P2P pour la téléphonie IP pour leur robustesse et fiabilité en se reposant sur le protocole de signalisation SIP pour sa flexibilité et son indépendance par rapport aux autres protocoles. Nous allons alors donner dans ce chapitre quelques notions de bases sur les architectures P2P-SIP, la classification de ses architectures et pour finir on va soulever les limites de ce type d'architectures.

## 3.2 Présentation des architectures P2P-SIP

P2P-SIP est la nomination donnée par le groupe de travail (*WG*) d'Internet Engineering Task Force (*IETF*). L'objectif de P2P-SIP c'est de bénéficier de la combinaison des avantages des architectures P2P (*ex. la décentralisation*) et ceux du protocole SIP (*ex. sa flexibilité*).

### 3.2.1 Systèmes P2P-SIP

C'est un ensemble de protocoles qui dérive du protocole SIP en utilisant les techniques P2P afin d'atteindre les objectifs des requêtes SIP, et de fournir d'autres services SIP [**10**].

### 3.2.2 Réseau overlay P2P-SIP

C'est un réseau qui permet aux nœuds de participer à la distribution des données et donne lieu à l'enregistrement, le routage, et différents services SIP. L'ensemble basique des services qu'un réseau overlay P2P-SIP doit offrir est toujours à discuter [**10**].

1. www.Skype.com

### 3.2.3 Entités P2P-SIP [10]

Dans les systèmes P2P-SIP classiques on distingue deux entités :

- **Pair P2P-SIP**
  Un pair P2P-SIP est un nœud qui participe dans un P2P-SIP overlay qui permet de stocker et de router les services vers d'autres nœuds qui sont dans le même réseau overlay. Un pair P2P-SIP peut être localisé à travers les NATs tout en restant fonctionnelle. Ceci peut performer plusieurs opérations : rejoindre et quitter le réseau, stockage des informations, insertion des informations dans le réseau et enfin récupérer les informations à partir de ce réseau.
  Un pair P2P-SIP peut être client P2P-SIP sans être nécessairement un utilisateur agent SIP.
- **Client P2P-SIP**
  Un nœud client est un nœud qui ne participe pas dans le réseau overlay P2P-SIP, certaines fonctions comme le routage, le stockage et la récupération de données ne lui seront donc pas accordées. Lorsqu'un client P2P-SIP souhaite interagir avec un réseau overlay P2P-SIP donné, il doit pénétrer par une paire P2P-SIP dans un même réseau overlay.

## 3.3 Classification des architectures P2P-SIP

Dans les architectures P2P-SIP on distingue deux grandes classes : P2P-over SIP et SIP-using-P2P, la différence ne provient pas des architectures elles mêmes mais de la manière comment les requêtes, les DHT et le protocole SIP sont manipulés. Plusieurs variantes ont été proposées dans chaque classe. Par conséquent, nous résumons la classification dans la figure 3.1 :

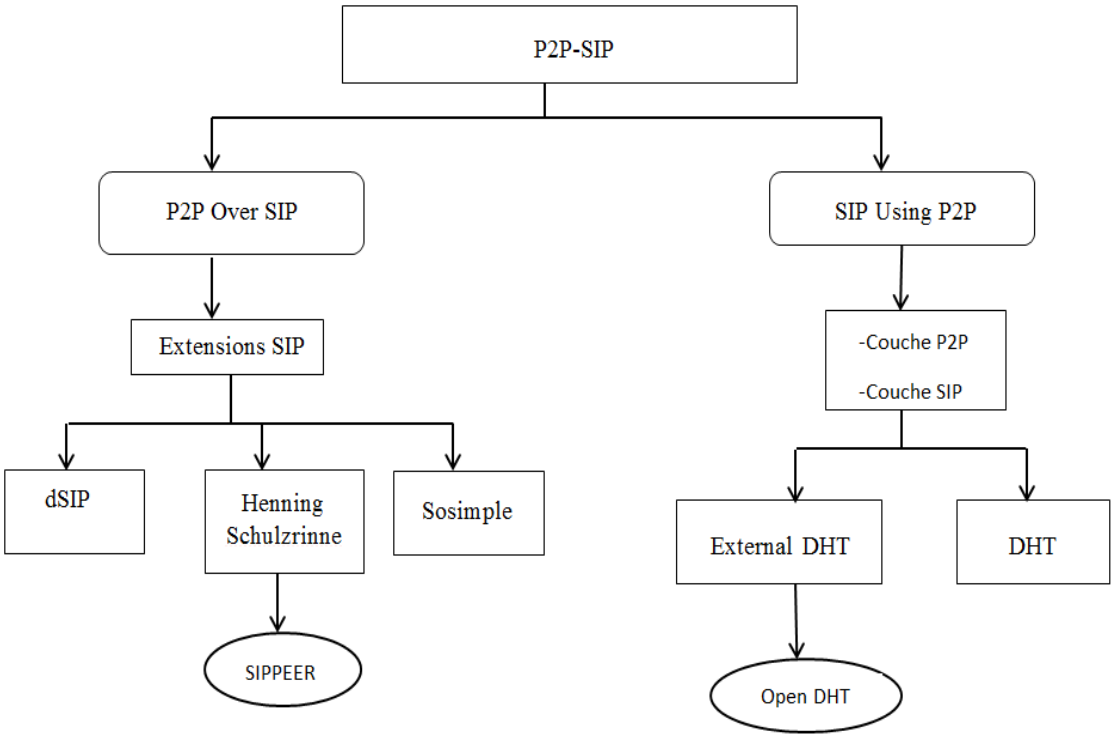


FIGURE 3.1 – *Classification des architectures P2P-SIP.*

## 3.4 P2P à travers SIP « P2P over SIP »

Dans cette approche, les messages SIP ne sont pas utilisés uniquement pour l'enregistrement des nœuds, la recherche des ressources et la gestion des sessions, mais le maintien des réseaux P2P a été aussi associé. C'est-à-dire les DHTs sont créées grâce aux messages SIP échangés, or la pile SIP n'a pas été conçue pour supporter les architectures distribuées. Pour répondre aux trafics P2P, plusieurs variantes de SIP ont été proposées parmi elles on distingue : dSIP, SIPPEER et SoSimple qui sont des extensions du protocole SIP. Elles utilisent l'enregistreur SIP pour la jonction des nœuds, la création et l'acheminement de tous les messages P2P.

### 3.4.1 Architecture Sosimple

Les auteurs dans [**12**] ont proposé une approche P2P-SIP nommée SOSIMPLE. Sosimple est un système P2P qui fournit des services de la VoIP et les messages instantanés en utilisant une DHT. Il combine la famille SIP/simple, il se base sur le protocole Chord pour la transmission fiable des messages SIP. La DHT est utilisée uniquement pour les communications directes entre les clients.

Sosimple utilise un système P2P décentralisé structuré lors de sa localisation, un nœud A publie ses ressources dans la DHT, contenant son adresse IP réelle. Quand le nœud B veut contacter le nœud A, il effectue une recherche dans la DHT, il obtient l'adresse IP de A et enfin A et B peuvent communiquer entre eux.

### Arrivé d'un nouveau nœud

Lorsqu'un nœud désire rejoindre l'overlay, il doit tout d'abord localiser un certain nœud dénommé Bootstrap se trouvant déjà dans l'overlay. Le nœud à joindre calcule son Node-ID (503 *dans notre exemple*), et l'envoie dans un REGISTER au Bootstrap, qui a le Node-ID 023 (1). Supposant que le Bootstrap n'est pas le nœud actuellement responsable de cette région, il répond avec des informations sur les nœuds les plus proches qu'il connait ou le nœud à joindre va être placé dans l'overlay dans ce cas, le nœud B, avec le Node-ID 445. Cette information retourne dans nos nouveaux en-têtes dans un SIP 302 avec une réponse temporairement déplacée (2). Le nœud à joindre répète le processus, en utilisant ce nœud proche comme le nouveau Bootstrap (3-4). Finalement, le nœud à joindre atteint le nœud qui est actuellement responsable du maintien du groupe approprié dans l'overlay, dans ce cas le nœud C, avec le Node-ID 520. Le nœud C répond avec une réponse OK SIP 200 y compris des informations détaillées à propos des voisins à proximité dans les en-têtes (5-6), permettant au nœud à joindre de s'insérer dans l'overlay. D'autres messages, non représentés, sont envoyés entre le nœud à joindre et le nœud étant auparavant responsable d'échanger les informations actuelles dans l'overlay que le nouveau nœud doit stocker. En outre, le nouveau nœud va envoyer des messages à d'autres nœuds dans l'overlay pour mettre à jour leurs tables de raccourcis.

La figure 3.2 représente l'arrivé d'un nouveau nœud :

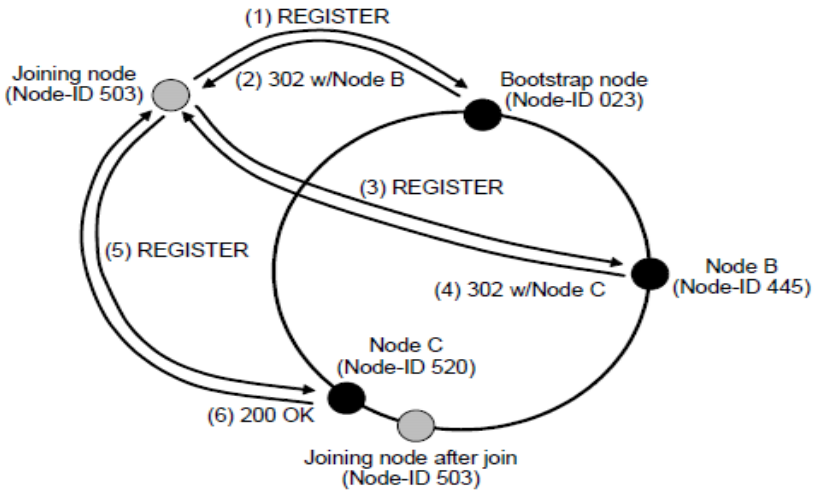


FIGURE 3.2 – *Exemple d'arrivé d'un nœud.*

### Localisation d'un nœud

Lorsqu'un nœud désire trouver le nœud responsable d'un utilisateur particulier soit lui-même ou un autre utilisateur qu'il souhaite contacter, il commence par hacher le nom de l'utilisateur pour produire un identifiant de ressource (Ressource-ID). Puisque le nœud de l'utilisateur est déjà bien dans l'overlay (*l'enregistrement du nœud a déjà eu lieu*), le nœud a un nombre d'entrées dans la table des raccourcis pointant vers des nœuds à l'intérieur de l'overlay. Dans notre exemple, Amanda tente de rechercher une ressource (*Amel dans notre exemple*). Le nœud d'Amanda cherche dans sa table des raccourcis et trouve le nœud avec le Node-ID le plus proche

au Ressource-ID à localiser. Dans ce cas, le nœud A. Le nœud d'Amanda envoie un message au nœud A (1). Le type exact dépend en pourquoi Amanda veut-elle localiser la ressource. Le nœud A n'est pas responsable de ce Ressource-ID, donc il envoie un SIP 302, y compris le nœud qu'il pense être le plus proche, le nœud B dans les en-têtes(2). Le nœud d'Amanda essaye le nœud B et reçoit de nouveau une réponse avec un autre nœud à essayer, dans ce cas le nœud C (3-4). Finalement le nœud d'Amanda essaye le nœud C, qui est responsable de cette ressource (5).

La figure 3.3 illustre la localisation d'un nœud :

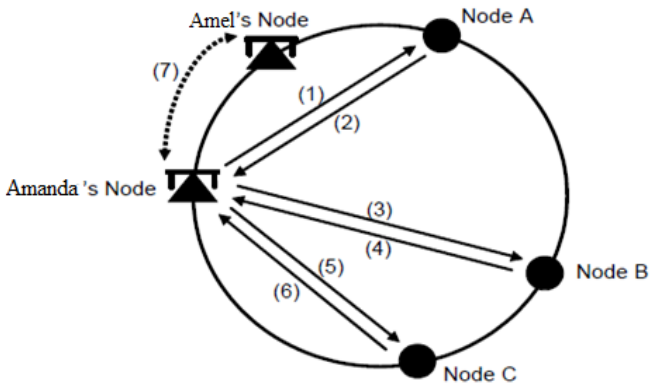


FIGURE 3.3 – *Exemple d'arrivé d'un nœud.*

### 3.4.2 Architecture de Schuzerlinne

Les auteurs de **[15]** ont proposé l'architecture P2P qui se base sur SIP pour sa robustesse, sa fiabilité mais aussi pour le fait de pouvoir le déployer sans pour autant modifier les infrastructures de contrôle comme le DNS. Le modèle P2P choisit est le structurel, il se repose sur les tables de hachages distribuées (*DHTs*) du protocole Chord. L'architecture proposée est une architecture hybride où les nœuds ordinaires sont des clients et les super-nœuds jouent le rôle des serveurs, cette architecture permet la téléphonie SIP traditionnelle ainsi que la recherche des utilisateurs sur le réseau P2P si le domaine locale ne possède pas de serveur SIP.

L'utilisation du protocole SIP permet d'implémenter les différentes fonctions de DHT dans P2P-SIP telles que : la découverte d'un pair, l'enregistrement des utilisateurs, la détection des nœuds défaillants, la localisation des utilisateurs, et enfin la configuration des appels.

La figure 3.4 est une illustration de l'architecture P2P-SIP proposée par H.Schuzerlinne :

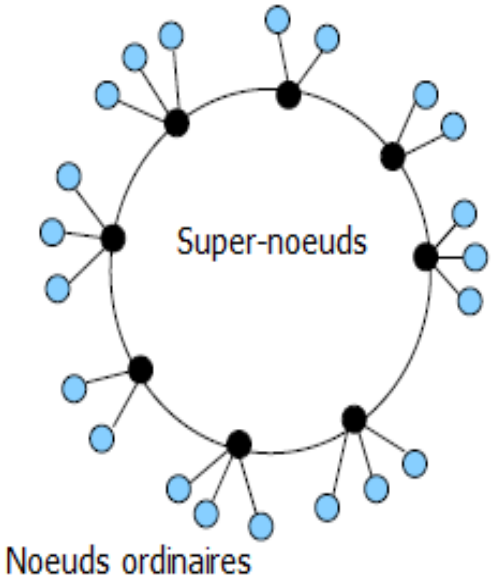


FIGURE 3.4 – *Modèle P2P-SIP de H.Schuzerlinne.*

**Enregistrement d'un utilisateur**

Lorsqu'un utilisateur souhaite s'enregistrer, il indique son nom utilisateur (*ex. Amanda@exchange.com*). Le nœud calcule la clé DHT à partir du nom indiqué puis rejoint la DHT. Lorsque la clé de l'utilisateur et celle du nœud sont identiques ceci ne permet ni l'envoie des messages offlines ni l'enregistrement des clients multiples pour un même utilisateur SIP, pour palier à ce problème la clé du nœud et celle de l'utilisateur sont calculées séparément où la 1ère est calculée à partir de l'adresse IP de l'utilisateur et la 2e est calculée à partir du nom

de l'utilisateur. Les messages Register sont utilisés pour l'insertion du nœud et l'enregistrement de l'utilisateur dans la DHT.
Les caractéristiques de l'enregistrement sont :

- un nœud ordinaire est uniquement un utilisateur agent SIP alors qu'un super-nœud joue le rôle d'un utilisateur agent SIP et d'un enregistreur aux autres nœuds.
- les nœuds ordinaires envoient périodiquement des messages Register d'actualisation afin de détecter toutes les défaillances du super-nœud.
- les super-nœuds envoient des messages Register aux nœuds attachés au super-nœud destination dans la DHT et se joint à cette dernière pour prendre part aux activités de la recherche de l'emplacement des utilisateurs.

La figure 3.5 représente l'enregistrement dans le modèle de Henning :

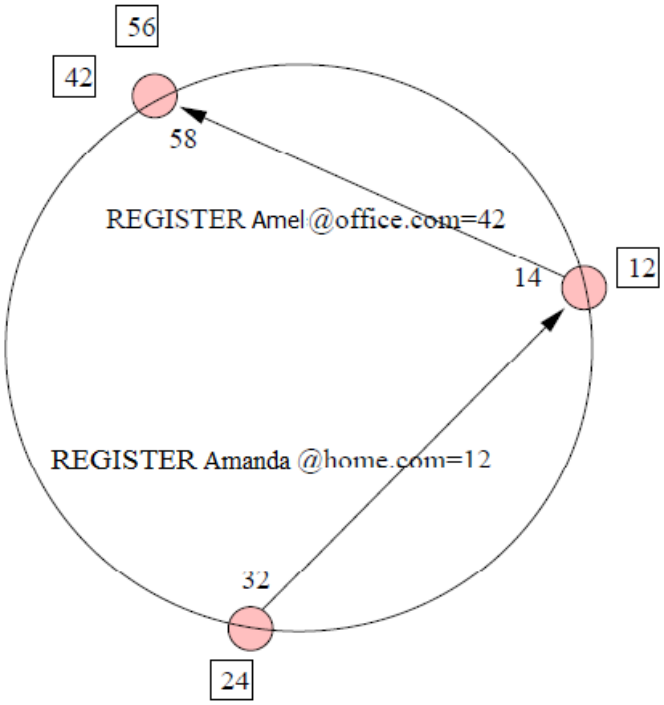


FIGURE 3.5 – *Enregistrement dans la méthode de Henning.*

### Découverte d'un pair

Quand le nœud démarre et l'utilisateur fait entrer son identifiant (*ex. Amanda@exchange.com*). Le nœud trouve toutes les adresses SIP possibles en utilisant DNS et envoie un message Register. Si l'enregistrement est réussi, le nœud peut être joignable en utilisant le mécanisme SIP en plus du mécanisme P2P.
Le nœud essaye de découvrir le maximum de super-nœuds afin qu'il puisse rejoindre le réseau overlay P2P.

### Nœud défaillant ou en panne

– Quand un nœud ordinaire quitte le système, il suffit uniquement de se désenregistré du super-nœud auquel il est attaché et ce dernier propage cette information

(*désenregistrement*) aux nœuds détenant la clé de ce nœud. Dans le cas où le nœud ordinaire est défaillant ceci n'affecte pas le système. Dans n'importe quel cas les super-nœuds détectent les panes à cause de la non réception des messages d'actualisation.

– Quand un super-nœud quitte le système, une mise à jour est nécessaire pour les nœuds ordinaires qui lui sont associés ainsi que pour les super-nœuds voisins existants dans la DHT. Si un super-nœud est en cours d'arrêt, il transfère le record (*enregistrement*) des utilisateurs qu'il détient sur les autres nœuds dans l'overlay P2P. Cela garantit que les autres utilisateurs peuvent localiser le record quand le nœud DHT s'est arrêté. Il envoie le message Register aux nœuds de la DHT qui seront tenue des enregistrements d'utilisateur, par la suite ce nœud quitte le système. Il n'a pas besoin d'informer les nœuds ordinaires attachés. Le nœud attaché détectera la défaillance sur le prochain enregistrement d'actualisation et donc il essayera de découvrir et se connecter aux super-nœuds qui détiennent le record.

## SIPpeer [16]

L'architecture P2P-SIP proposée par Henning Schurzlinne supporte des fonctions basiques telles que : l'enregistrement des utilisateurs et la configuration des appels, ainsi que des services avancés comme la délivrance des messages offline et les multi-conférences mais pour que les utilisateurs agents existants puissent participer dans les réseaux P2P, un adaptateur client est nécessaire et c'est ce qui a engendré la motivation de SIPpeer. SIPpeer est un adaptateur client qui a été implémenté dans l'architecture P2P-SIP proposée par H. Schurzlinne car elle permet aux utilisateurs agent existants comme X-lite et IP cisco ou au nouveaux UA de se connecter aux réseaux P2P-SIP sans pour autant modifier les utilisateurs agents. SIPpeer peut s'exécuter dans un hôte non seulement comme un PC basé sur un utilisateur agent SIP mais aussi comme un serveur proxy. Il peut aussi réagir comme un utilisateur agent, un serveur proxy ou comme un enregistreur d'une manière autonome et ceci en utilisant une interface utilisateur de ligne de commande.

### 3.4.3 Architecture dSIP

dSIP (*distributed SIP*) [**9**] est une approche qui se base sur P2P pour l'enregistrement SIP et la découverte de ressources en utilisant les tables de hachage distribuées (*DHT*) entretenues avec des messages SIP. Cette conception élimine la nécessité des serveurs centraux SIP, tout en offrant une compatibilité ascendante avec SIP, permettant la réutilisation des clients existants, et permettant également aux pairs P2P de communiquer avec les entités SIP classiques.

#### Enregistrement d'un nouveau nœud

Quand un paire souhait rejoindre l'overlay, il détermine son identifient Peer-ID et envoie un message REGISTER à un pair Bootstrap qui est un pair qui existe déjà dans l'overlay, en demandant à joindre le réseau. Tous les pairs dans la DHT peuvent servir comme un pair Bootstrap.
Le Bootstrap vérifie le pair, il connait l'id de pair (*Peer-ID*) le plus proche du pair qui veut rejoindre le réseau et répond avec redirection 302 à ce plus proche nœud. Le pair qui veut rejoindre va répéter ce processus jusqu'à ce qu'il atteigne le pair actuellement responsable de l'espace qu'il occupera.

**Enregistrement d'une ressource**

L'enregistrement par pair n'enregistrer pas l'utilisateur de pair ou d'autres ressources avec le réseau P2PSIP, il permet uniquement aux pairs de rejoindre l'overlay. Une fois un pair a rejoint le réseau, le pair hôte de l'utilisateur doit être enregistré avec le système. Ce processus est appelé l'enregistrement des ressources.
L'enregistrement d'une ressource se fait de la même manière que l'enregistrement d'un nœud.

**Etablissement d'une session**

Les sessions sont établies en communiquant avec l'agent utilisateur identifié par l'enregistrement dans la DHT. La première étape dans l'établissement d'une session est de localiser le pair, elle se fait par la recherche d'une ressource dans la DHT. Le nom de la ressource cible est utilisé pour calculer l'identifiant de la ressource (*res- ID*) et un message REGISTER avec aucune information de contact (*la recherche SIP classique*) est envoyé à un pair connu et plus proche de ce res -ID. La recherche se répète jusqu'à ce que le nœud responsable soit localisé, puis il renvoie un 200 OK avec les informations du contact pour la ressource ou un « 404 Not Found ». La session est alors initiée directement avec l'agent utilisateur de la ressource.

## 3.5 SIP utilisant P2P « SIP using P2P »

Contrairement à l'approche P2P over SIP qui utilise uniquement une seule pile, l'approche « SIP using P2P » dispose de deux piles : la première offre des services SIP, pour tout ce qui est d'enregistrement, la recherche des ressources et la gestion des sessions ; la seconde pile est donc assignée au maintien du réseau distribué. Cette approche réduit les coûts et la complexité. De plus, l'implémentation des deux piles peut s'effectuer dans différents nœuds dans les réseaux P2P-SIP d'où l'interopérabilité du système.

### 3.5.1 External DHT

Contrairement aux modèles cités précédemment, celui-ci utilise une DHT externe se reposant sur les fonctionnalités d'OpenDHT qui ne se créée pas au fur et à mesure et ne dispose nullement des super-nœuds ni de nœuds ordinaires. Deux autres notions sont alors définies : client P2P et proxy P2P.

- **Un client P2P** est un agent utilisateur qui ne nécessite pas un serveur et exécute directement les recherches et les mises à jour de manière autonome.
- **Un proxy P2P** est un serveur proxy SIP qui effectue les recherches et mises à jour P2P de manière transparente aux utilisateurs.

Les auteurs [**17**] dans ce cas ont pris en considération uniquement la gestion et le stockage des contacts des utilisateurs et selon eux une DHT est assez suffisante que d'implémenter une base de données P2P complète.

**Gestion des contacts**

La DHT est utilisée pour sauvegarder les informations des contacts de l'utilisateur. Par exemple, si Nassim sauvegarde ses contacts sous une clé DHT $k$ avec $k$ = H (SIP : Nassim@essone.com). Ce simple schéma permet aux utilisateurs multiples de s'enregistrer sous un même identifiant SIP.

- **Le client P2P** signe les données pour le compte utilisateur, un utilisateur doit être en mesure d'utiliser un autre client et mettre à jour les informations de ses contacts. Ce mode permet à l'utilisateur de choisir son propre identifiant SIP, tant qu'il peut prouver que l'identifiant lui appartient par un certificat ($S$). Il n'y a aucune dépendance à un serveur SIP.
- **Le proxy P2P** authentifie l'utilisateur, puis signe les données mises sur la DHT. Par exemple, lorsque l'utilisateur amanda@monday.com s'enregistre avec le proxy P2P du domaine monday.com, le proxy signe ses contacts à l'aide de l'identité du signataire en tant home.com.Pour permettre à d'autres mandataires (*proxy*) de modifier ou supprimer les contacts, toutes les procurations de home.com doivent utiliser la même clé pour la signature. Cela permet à l'utilisateur d'utiliser de manière transparente n'importe quel proxy.

### Gestion de clés

Pour éviter tout serveur central, les certificats, les clés, et toute configuration sont également stockées sur le DHT. Par exemple, Nassim peut stocker son certificat et la clé publique sur la DHT avec $k1$ = H (certificat : Nassim@essone.com ) et $k2$ = H (public : Nassim@essone.com), respectivement. Plusieurs certificats de Nassim de différents AC peuvent être mis sous la même clé de DHT. Puisque l'information doit être disponible à tout appelant potentiel, la valeur est non cryptée. Il y a un danger d'autres malveillants utilisateurs qui polluent les valeurs de DHT pour cette clé. Cependant, la vérification enchaîne des certificats peut être utilisée pour récupérer le certificat correct.

### Modèle de données et de services

1. **Modèle de données**
   Dans ce modèle, la DHT est utilisée en tant que mémoire de données partagée et les opérations de P2P- SIP sont exécutées par l'utilisateur. Par exemple, un utilisateur stocke ses informations de contact et un appelant emmagasine les messages hors ligne dans la DHT. De même, un proxy P2P met à jour les données de la table DHT pour le compte de l'utilisateur pour fournir un service SIP transparent pour les utilisateurs non-P2P.
2. **Modèle de services**
   Dans ce modèle, chaque client P2P-SIP ou proxy, se joint à la DHT pour le service P2P-SIP, par exemple, en utilisant l'interface OpenDHT. Le service P2P-SIP comprend l'enregistreur SIP, agent de présence, le stockage des messages offline, et les serveurs STUN et TURN au minimum.
   Lorsqu'un utilisateur, Amanda par exemple, veut envoyer un message SIP, dire sip : Nassim@essone.com, elle vérifie dans la DHT pour trouver le nœud responsable de cet identifiant d'utilisateur du service, et envoie une demande SIP à ce nœud. Le nœud de service agit en tant qu'un proxy, enregistreur et serveur de présence pour tous les utilisateurs pour lesquels il est responsable.
   Le modèle de service est adapté à la fois au P2P-over-SIP et SIP-using-P2P.

**NB** :le modèle de service peut être construit en utilisant le modèle de données sous-jacentes, du fait que les nœuds de service utilisent également le format de données spécifié pour le stockage dans la DHT.

La figure 3.6 illustre le modèle de données et services :

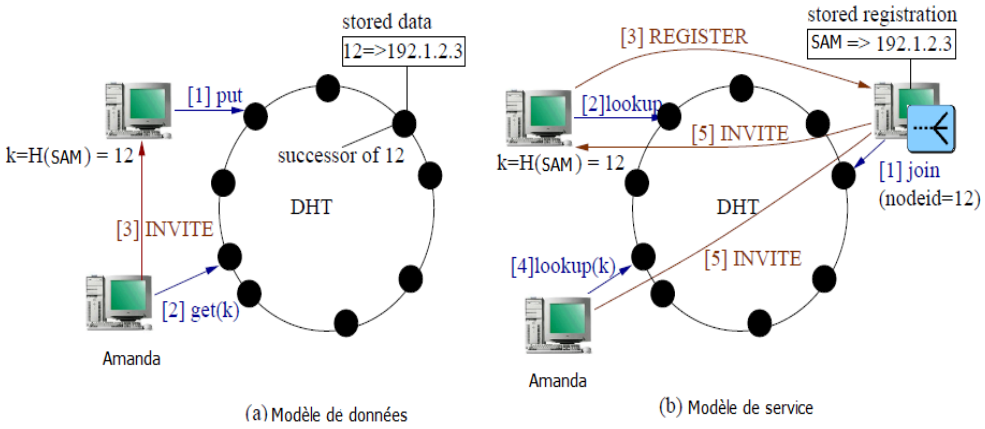


FIGURE 3.6 – *Exemple sur les modèles de données et de services.*

## 3.6 Comparaison entre les architectures P2P-SIP

Après avoir présenté les architectures P2P-SIP les plus fréquentes, nous avons aboutis à voir clairement que la différence se porte sur le déploiement des nœuds, la façons dont les nœuds sont atteints (*localisés*), l'acheminement des données ou la manière comment le réseau est maitenu. Le tableau 3.1 est un récapitulatif comparatif entre les différentes architectures P2P-SIP citées :

| **Architecture** | **Classe** | **Type** | **Avantages** | **Inconvénients** |
|---|---|---|---|---|
| H.Schulzrinne | P2P-over-SIP | Hybride, non-structurée | compléxité réduite à O(log(n)) | garde toujours la notion du client serveur |
| Sosimple | P2P-over-SIP | Décentralisé, structurée | pas de serveurs central,le nombre d'entrées dans la table de raccourcis est réduit à 16 au lieu de 160 | coûteuse en terme de messages |
| dSIP | P2P-over-SIP | Décentralisé, structurée | pas de serveurs centraux, permet la réutilisation des clients existants | très couteuse en terme de nombre de messages lors de la jonction et de la recherche |
| External DHT | SIP-using-P2P | Décentralisé, structurée | sauvegarde de plusieurs données avec une seule clé | problèmes de sécurité |

TABLE 3.1 – *Tableau comparatif entre les différentes architectures P2P-SIP*

## 3.7 Limites des architectures P2P-SIP

### 3.7.1 Tolérance aux pannes

Les mécanismes de détection des pannes dans les réseaux P2P ne font pas la différence entre les défaillances physiques et temporelles. Si un super-nœud n'envoie pas le message d'acquittement aux nœuds ordinaires qui sont attaché à lui, il sera considéré comme étant en panne. Dans le cas où le super-nœud d'un nœud ordinaire est en panne, alors ce dernier ne peut ni effectuer ni recevoir des appels.

### 3.7.2 Surcharge du réseau

La surcharge des super-nœuds est plus grande que les nœuds ordinaires car les super-nœuds stockent toutes les identifiants et clés des nœuds auxquels il est responsable. Ce qui implique l'entretien d'une grande quantité d'informations.

### 3.7.3 Sécurité

La sensibilité d'un système de réputation aux attaques dépend du coût auquel les nœuds peuvent être générés, du degré auquel le système accepte les messages provenant des nœuds qui n'ont pas de chemin de confiance liés à un autre nœud de confiance.

## 3.8 Conclusion

Dans ce chapitre, nous avons en premier lieu cité quelques notions de base sur les architectures P2P-SIP. En second lieu, nous avons résumé les classifications dans un schéma, nous les avons par la suite détaillées et expliquées afin de distinguer au mieux entre elles.

Nous avons, de surcroit, effectué une comparaison entre les deux approches (*P2P over SIP et SIP using P2P*) ainsi les différentes architectures proposées dans les deux approches.

En conséquence, nous avons déterminé les limites des architectures P2P-SIP qui nous serviront comme outil pour extraire notre problématique.

Nous allons expliquer en détail dans le chapitre qui suit la problématique, la solution proposée, les différentes techniques et algorithmes utilisés pour le développement de notre proposition.

Chapitre 4

# Routage prioritaire dans les architectures P2P-SIP structurées

## 4.1 Introduction

Plusieurs recherches dans le domaine P2P-SIP ont été effectuées, plusieurs architectures et travaux ont été proposés pour un seul et unique but : c'est d'améliorer et d'utiliser les architectures P2P-SIP sur la téléphonie IP de manière optimale, fiable et sécurisée pour qu'elle puisse enfin être mis en œuvre dans des applications fiables et robustes qui satisferont les utilisateurs.

Dans ce chapitre, nous allons soulever quelques problèmes d'optimisation rencontrés beaucoup plus par les architectures structurées en temps réel, nous allons par la suite proposer une solution pour remédier aux problèmes cités et enfin nous allons détailler le modèle proposé ainsi que les détails de son fonctionnement.

## 4.2 Problématique

Certes que les architectures structurées sont robustes, fiables et moins couteuses que celles non-structurées mais on retrouve des lacunes communes et ceci en terme de paramètres de qualité de service : le temps de transmission, perte de paquets, etc. Dans la téléphonie sur IP, la voix est transmise en temps réel, d'où l'importance d'optimiser le temps de transmission (délai), un paramètre très important de la qualité de service ($QoS$).

La transmission des messages de localisation via les protocoles classiques des architectures P2P structurées ne favorise pas les appels d'urgence, i.e., ils sont traités exactement comme n'importe quel autre appel ordinaire.

## 4.3 Proposition

Nous avons choisi d'effectuer notre étude sur le protocole Chord qui est un cas particulier des architectures structurées.

Pour pallier aux problèmes cités précédemment, nous allons rajouter au fonctionnement classique du protocole Chord lors de la localisation d'un utilisateur (*l'appelé*) un paramètre important pour favoriser les appels d'urgence par rapport aux autres. Ce paramètre doit permettre de distinguer les messages de localisation les plus prioritaires et urgents.

## 4.4 Modélisation de la solution proposée

Lorsqu'un nouvel utilisateur A souhaite retrouver un autre utilisateur, dit B, dans le réseau, il envoie une requête à son super-nœud, ce dernier vérifie dans la DHT si l'utilisateur recherché ($B$) existe, si c'est le cas la requête sera alors transmise à B, sinon le super-nœud responsable de A va effectuer une recherche dans sa table de raccourcis et vérifie le super-nœud ayant l'identifiant le plus proche de celui de B, et il lui envoie cette requête. Ce processus va se répéter jusqu'à ce que B soit alors retrouvé ou bien il n'existe pas dans le réseau.

### 4.4.1 Fonctionnement global

Nous souhaitons modéliser notre réseau P2P (*Chord*) par un réseau de files d'attente, il est alors considéré comme un graphe G= $(N, F)$ tel que N est l'ensemble des nœuds et F est l'ensemble des files, sachant qu'à chaque nœud est associé une file dans le réseau.

Nous allons élaborer notre étude sur l'arrivé des clients (*nombre de requêtes reçu par les files d'attente*) ainsi que le temps effectué lors de la localisation d'un utilisateur.

On considère que les requêtes ont un champ en plus ou on définit le degré de leurs priorité. Si un appel est urgent le champ serra mit à « 1 » sinon à « 2 ». Si une requête dispose d'un « 1 » elle sera placée à la tête de la file, dans le cas contraire, l'ordre sera selon l'ordre d'arrivé (*FIFO*), de même si deux requêtes prioritaires arrivent l'une après l'autre, l'ordre de traitement serra selon leurs ordre d'arrivé.
La figure 4.1 représente le modèle proposé :

### 4.4.2 Caractéristiques d'une file d'attente

Une file d'attente est représenté par un ou plusieurs serveurs et un seul buffer, les clients reçus attendent dans le tampon afin d'être servi.

Une file d'attente est représentée généralement suivant la notation de Kendall définit par : **A/S/C/K/L/D** où :

- **A** : indique la loi de probabilité pour générer la variable aléatoire de la durée entre deux arrivées successives.
- **S** : indique la loi de probabilité de la variable aléatoire de service.
- **C** : désigne le nombre de serveurs.
- **K** : c'est la capacité maximale du tampon.
- **L** : la population des usagers (*c'est le nombre des clients susceptibles à utiliser la file d'attente*).
- **D** : la discipline du service (*l'ordonnancement*).

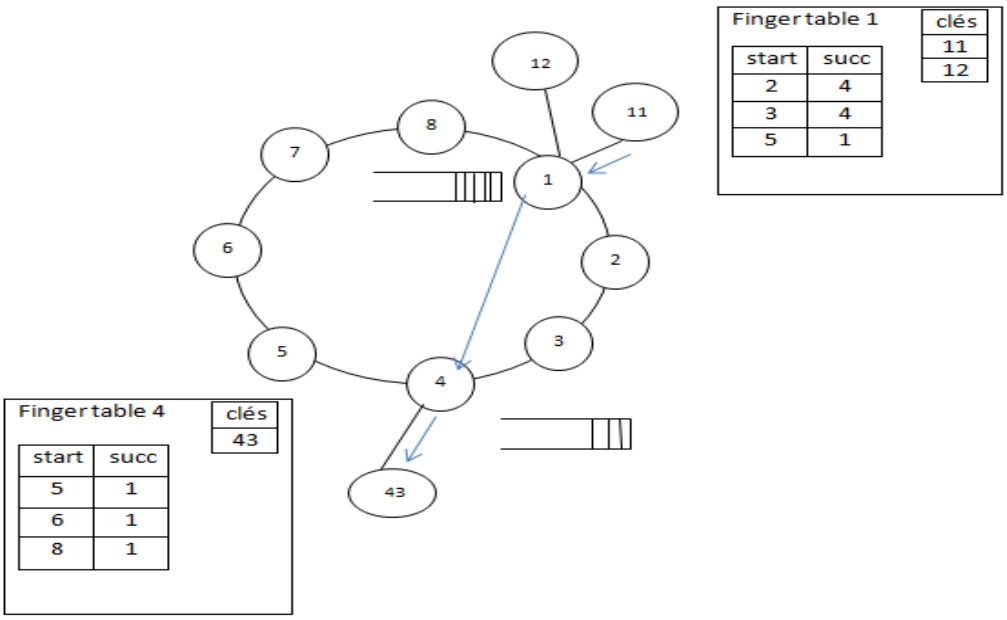


FIGURE 4.1 – *Réseau de files d'attente associé à Chord.*

Les symboles suivants sont utilisés pour représenter et définir les principales lois utilisées pour générer les variables aléatoires des inter-arrivés et du service :

- **M** : loi exponentiel markovienne.
- **D** : loi déterministe ou constante.
- **Ek** : loi d'Erlang à k étapes.
- **Ck** : loi de cox.
- **G** : loi générale.

**NB** :lorsque les trois derniers symboles ne sont pas spécifiés, ils correspondent par défaut aux valeurs suivantes : DS=FIFO, K= $\infty$, L= $\infty$.

Nous avons choisi d'attribuer à chaque nœud du réseau une file d'attente de type M/M/1, où la durée des inter-arrivés et de service sont des variables aléatoires qui suivent la loi exponentiel. La figure 4.2 est un exemple d'une file d'attente M/M/1 :

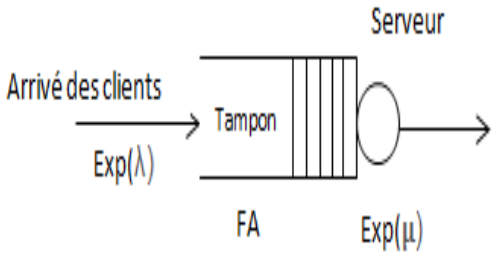


FIGURE 4.2 – *Modèle de file d'attente M/M/1.*

### 4.4.3 Métriques de performances

Les clients arrivent avec un taux d'arrivé $\lambda$ et se mettent dans la file d'attente selon FIFO, ils sont servi selon le taux de service $\mu$.
Les métriques de performances du modèle M/M/1 sont résumés dans le tableau 4.1 :

| **Métrique** | **Formule** | **Signification** |
|---|---|---|
| $\lambda$ | $1/Ta$ | le taux moyen d'arrivée des clients |
| $\mu$ | $1/Ts$ | le taux moyen de service |
| **R** | $(\lambda 1 + \lambda 2)/\mu$ | la charge du système |
| **P(n)** | $R^n/(1-R)$ | la probabilité qu'il y ait n clients dans le système |
| **E(N)** | $R/(1-R)$ | le nombre moyen de clients dans le système |
| **E(L)** | $R2/(1-R)$ | le nombre moyen de clients dans la file d'attente |
| **E(W)** | $R/\mu * (1-R)$ | le temps moyen d'attente |
| **E(T)** | $1/\mu * (1-R)$ | le temps moyen de réponse |

TABLE 4.1 – *Les métriques de performances du modèle M/M/1*

### 4.4.4 Simulation M/M/1

L'algorithme de simulation M/M/1 nous permet de trouver le temps de transmission dans le système. Les variables utilisées par cet algorithme sont citées dans le tableau 4.2 : **NB** : "i"

| **Variable** | **Designation** | **Initialisation** |
|---|---|---|
| **H** | l'horloge initiale | 0 |
| $n_i$ | le nombre de clients de le système | 0 |
| $Ta_i$ | instant d'arrivé des clients dans le système | 0, i=1<br>0.1, i=2 |
| $Ts_i$ | instant de fin de service | Tmax |
| $Tb_i$ | l'instant de réactivation du serveur | 0 |
| $b_i$ | la durée cumulée pendant laquelle le système est actif | 0 |
| $Hp_i$ | l'instant de réalisation de l'évènement | 0 |
| $x_i = \sum nj * Dj$ | le temps cumulé entres les clients | 0 |

TABLE 4.2 – *Les variables utilisées dans l'algorithme et leurs initialisation*

signifie qu'il y a plusieurs variable de même type et n'a pas la même valeur. Ex : Ta1 et Ta2.

**algorithm 1** Procédure porce

$x \leftarrow x + n * (H - HP)$
$n \leftarrow n + 1$
$Hp \leftarrow H$
**if** $(n = 1)$ **then**
    $Ts \leftarrow H - \ln(rand)/m$
**end if**

**algorithm 2** Procédure Ps

$x \leftarrow x + n * (H - HP)$
$n \leftarrow n - 1$
$D \leftarrow D + 1$
$Hp \leftarrow H$
**if** $(n = 0)$ **then**
    $Ts \leftarrow Tmax$
    $b \leftarrow b + (H - Tb)$
**end if**

**algorithm 3** File d'attente M/M/1

```
1: Tmax ← 100
2: Ta1 ← 0
3: Ta2 ← 0.1
4: Ts_i ← Tmax
5: while H ≤ Tmax do
6:   if Ta1 < Ta2 then
7:     if Ta1 < min(Ts_i) then
8:       H ← Ta1
9:       Ta1 ← H − ln(rand)/L1
          proce(x1, n1, Hp1, Ts1, Tb1, m1, H)
10:    else
11:      H ← Ts1
         Ps (Ts1,x1,n1,D1,Hp1,Tb1,b1,H)
12:      while Rand ≤ P[k] ∧ K ≤ n do
13:        k ← k + 1
14:      end while
          proce(x_{k}, n_{k}, Hp_{k}, Ts_{k} ,Tb_{k} ,m_{k}, H)
15:    end if
16:  else
17:    if Ta2 < min(Ts_i) then
18:      H ← Ta1
19:      Ta2 ← H − ln(rand)/L2
          proce(x1,n1,Hp1,TS1,Tb1,m1,H)

20:    else
21:      H ← Ts1
          Ps (Ts1, x2 ,n2 ,D2 ,Hp2 ,Tb2, b2, H)
22:      while Rand ≤ P[k] ∧ K ≤ n do
23:        k ← k + 1
24:      end while
          proce(x_{k}, n_{k}, Hp_{k}, Ts_{k}, Tb_{k}, m_{k}, H)
25:    end if
26:  end if
27: end while
```

### 4.4.5 Localisation d'un utilisateur

Dans cette section, nous allons présenter l'algorithme de localisation suivant le protocole Chord classique et l'algorithme Chord avec priorité.

#### Algorithme de localisation dans le protocole Chord classique

L'algorithme ci-dessus nous montre le fonctionnement du protocole chord lors de la localisation :

**algorithm 4** Algorithe sans priorité

1: **if** $(Nc = source)$ **then**
2: **while** $(i < m \wedge i > k \wedge i + 1 \leq k)$ **do**
3: $i \leftarrow i + 1$
4: **end while**
`Envoyer (req (k), i+1)`
5: **end if**
{A la récéption}
6: **if** trouve($k$) **then**
`Envoyer (req (k), s)`
{Envoyer k à la source}
7: **else**
8: **while** $i < m \wedge (i > k \wedge i + 1 \leq k)$ **do**
9: $i \leftarrow i + 1$
10: **end while**
`Envoyer (req (k), i+1)`
11: **end if**

**Algorithme de localisation avec priorité**

L'algorithme ci-après nous montre le fonctionnement du protocole chord avec priorité lors de la localisation :

**algorithm 5** Algorithme avec Priorité

1: **if** $(Nc = source)$ **then**
2: **while** $(i < m \wedge i > k \wedge i + 1 \leq k)$ **do**
3: $i \leftarrow i + 1$
4: **end while**
`Envoyer (req (k), i+1)`
5: **end if**
{A la récéption}
`Enfiler()`
6: **if** trouve($k$) **then**
`Envoyer (req (k), s)`
{Envoyer k à la source}
7: **else**
8: **while** $(i < m \wedge i > k \wedge i + 1 \leq k)$ **do**
9: $i \leftarrow i + 1$
10: **end while**
`Envoyer (req (k), i+1)`
`Defiler()`
11: **end if**

## 4.5 Conclusion

Dans ce chapitre, nous avons déterminé les problèmes rencontrés par les architectures P2P-SIP, nous avons proposé une solution qui permet d'améliorer et d'optimiser le temps de transmission des messages lors de la localisation dans les architectures structurées en utilisant un champ spécifique dans les requêtes afin de minimiser le temps de transmission et de traitement des appels d'urgence. Nous avons par la suite, modélisé notre architecture en utilisant un réseau de file d'attente. Par conséquent, nous avons élaboré l'algorithme de simulation ainsi que l'algorithme de découverte tout en implémentant les files d'attente.

Dans le chapitre suivant, nous allons effectuer l'évaluation des performances de notre système et c'est ainsi que l'on peut observer les résultats de la simulation.

Chapitre 5

# Aspects techniques et mise-en-œuvre

## 5.1 Introduction

Nous avons présenté en détail dans le chapitre précédent la solution que nous avons proposée pour minimiser un paramètre que l'on juge important en termes de qualité de service qu'est : le temps (*délai*).

Dans ce dernier chapitre, nous allons présenter en premier lieu les différentes techniques d'évaluation des performances existantes et c'est ainsi que l'on peut extraire la technique sur laquelle on va se basé. Nous allons ensuite, présenter l'outil de simulation avec lequel nous allons effectuer. On va expliquer en détail le déroulement de la simulation et nous présentons quelques résultats de simulation obtenus.

## 5.2 Techniques d'évaluation des performances

Ayant pour but de prouver qu'un système est plus optimal qu'un autre, l'évaluation des performances est le meilleur moyen qui permet cela. Il existe trois grandes approches pour évaluer les performances d'un système donné. Nous les résumons comme suit :

### 5.2.1 Technique de mesure

Cette technique permet d'évaluer un système existant ou au moins un prototype de ce système. Elle se base sur les outils d'instrumentations d'un système qui est une méthode indispensable pour mesurer les performances au cours de son exécution, en insérant des codes de mesures dans son code source ou en utilisant des outils d'observation collectant des mesures.

L'un des principaux avantages de cette technique est la réalité des résultats, puisque les données sont prises d'un système réel. Cependant, ces résultats, malgré leurs réalismes, sont relatifs et variables. Ils dépendent de plusieurs paramètres : la configuration du système, la charge et le temps de mesures qui sont généralement uniques et relatifs à chaque test d'Evaluation des peformances. Ces résultats ne sont pas généraux mais spécifiques à un test particulier. Un autre inconvénient à ne pas omettre : c'est le coût de cette technique qui est très important, car elle nécessite des équipements réels [**2**].

### 5.2.2 Evaluation par simulation

La simulation s'impose comme la technique la plus utilisée pour évaluer les performances des systèmes informatiques. Elle représente un moyen utile pour prédire les performances d'un système et les comparer sous plusieurs conditions et configurations

Un atout majeur de cette technique est sa flexibilité, puisqu'elle permet d'évaluer le système sous plusieurs conditions et configurations [**2**].

Un aspect à ne pas oublier sur la simulation c'est le fait que les résultats obtenus sont crédibles.

L'inconvénient de cette technique : est la nécessité de la maitrise en excellence des techniques statistiques pour une analyse pertinente des résultats, ainsi que la maitrise de divers langages de programmations pour développer le modèle à simuler sous un langage approprié. Le désagrément majeur de cette technique, est qu'elle nécessite, un temps potentiellement important et des ressources de calculs conséquentes.

### 5.2.3 Technique analytique

La technique analytique est l'un des moyens les plus rapides, comparé avec les deux autres pour évaluer les performances d'un système. Elle se base sur la modélisation des systèmes sous formes de paramètres, de variables et un ensemble de formules mathématiques qui régissent leurs **relations** [**2**].

Cette technique se base sur plusieurs méthodes : réseaux de files d'attentes, réseau de pétri, chaine de Markov et etc.

## 5.3 Outils de simulation

Nous avons opté pour la technique analytique où nous avons modélisé notre réseau via un réseau de files d'attentes. Pour un calcul rapide et meilleur des résultats nous avons choisi « **Matlab**» comme langage de programmation pour implémenter notre algorithme.

### 5.3.1 Qu'est-ce que Matlab ?

Matlab (*Matrix LABoratory*) est un logiciel qui permet d'effectuer des calculs numériques. Il a été conçu initialement pour faciliter le traitement des matrices mais il est désormais utilisé dans tous les domaines des sciences qui nécessitent de faire des calculs.

### 5.3.2 Pourquoi Matlab ?

Le choix du langage Matlab revient à :
- Sa simplicité d'implémentation ;
- Sa rapidité lors des calculs et l'exécution des programmes ;
- La richesse de sa librairie ;
- La possibilité d'inclure le langage C et C++ ;

- La possibilité d'exécuter du code en dehors du programme ;
- Code facile à comprendre et lisible ;
- Langage interprété.

## 5.4 Diagramme déploiement

Afin d'implémenter notre simulation, nous allons déclarer les variables et les initialiser. Par la suite, nous allons appliquer les algorithmes de simulations avec et sans priorité et pour conclure nous allons afficher les résultats obtenus. La figure 5.1 représentation du diagramme de déploiement :

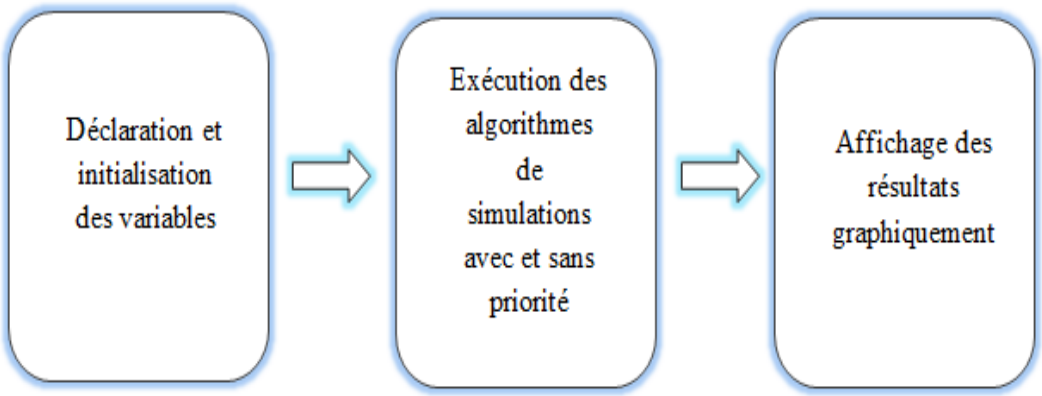


FIGURE 5.1 – *Diagramme général de déploiement du simulateur M/M/1.*

## 5.5 Déroulement de la simulation

### 5.5.1 Déclaration et initialisation des variables

Les variables que nous avons déclarés sont initialisées avant de participer dans l'algorithme de simulation, telles que le temps d'arrivés des clients prioritaires et non prioritaires, les temps de service des différents nœuds qui sont choisis selon certaines probabilités et c'est ce qui permet de designer le protocole Chord.

### 5.5.2 Exécution des algorithmes de simulation avec et sans priorité

Dans cette on explique en générale la différence entre les deux algorithme : avec et sans priorité.

#### Algorithme avec priorité

dans ce cas de figure, nous avons deux classes de clients (*inter-arrivés*). Nous avons la classe 1, où on retrouve les requêtes les plus prioritaires et les plus urgentes et une seconde classe où les requêtes ne sont pas prioritaires.

**Algorithme sans priorité**

Dans ce cas, nous n'avons qu'une seule classe de clients et ils passent selon leur ordre d'arrivée (*la politique FIFO*)

### 5.5.3 Affichage des résultats de la simulation

C'est dans cette étape que les résultats de la simulation obtenus seront afficher et donner de manière comparative. Autrement dit, on aura les résultats des deux simulations avec et sans priorité et ces résultats seront affiché dans un graphe.

## 5.6 Résultats de la simulation

Vu qu'on s'interesse à la qualité de service et parmis ces paramètres nous avons choisi le temps. On s'interesse alors à deux métriques de performances : le temps de réponse et le temps d'attente dans la file. Les résultats lors de l'execution des deux algorithmes avec et sans priorité sont affichés dans les figures suivantes :

**Résultats : N=8 nœuds**

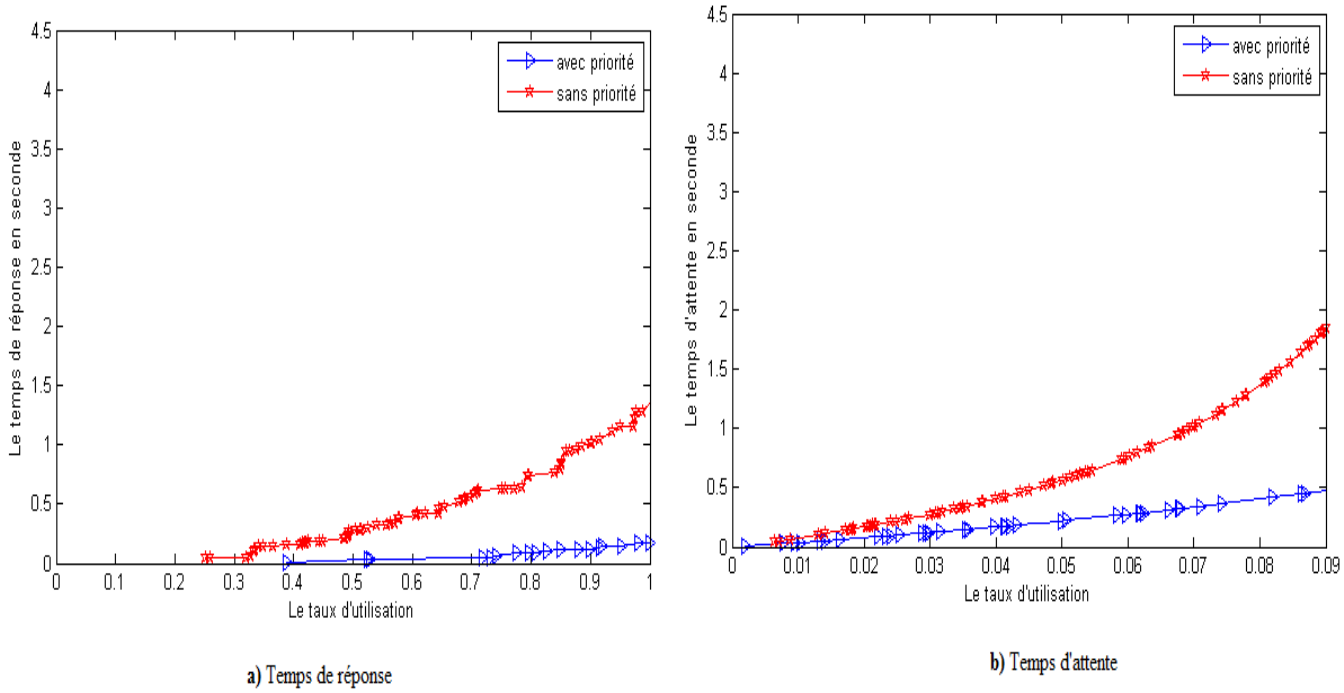


FIGURE 5.2 – *Résultats lorsque N=8.*

**Résultats : N=16 nœuds**

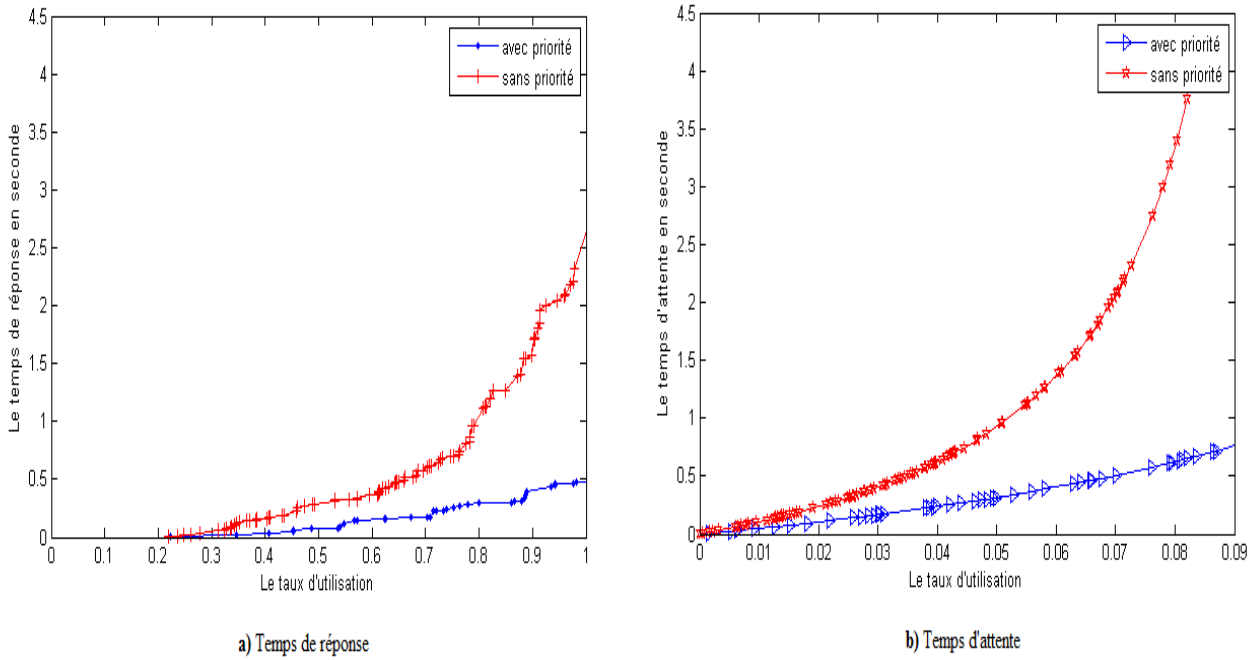


FIGURE 5.3 – *Résultats lorsque N=16.*

On remarque que :

- Dans la figure 5.2 il y a un grand écart entre le routage avec priorité et le routage sans priorité en terme de temps d'attente et de réponse et ceci est valable pour les figures 5.3 5.4.
- Les temps d'attente et de réponse augmentent lorsque le taux d'utilisation augmente car les nœuds sont surchargés
- Lorsque le nombre de nœuds s'accroît, le chemin entre la source et la destination devient long. Par conséquent les temps d'attente et de réponse du système s'accroîent.

**Résultats : N=32 nœuds**

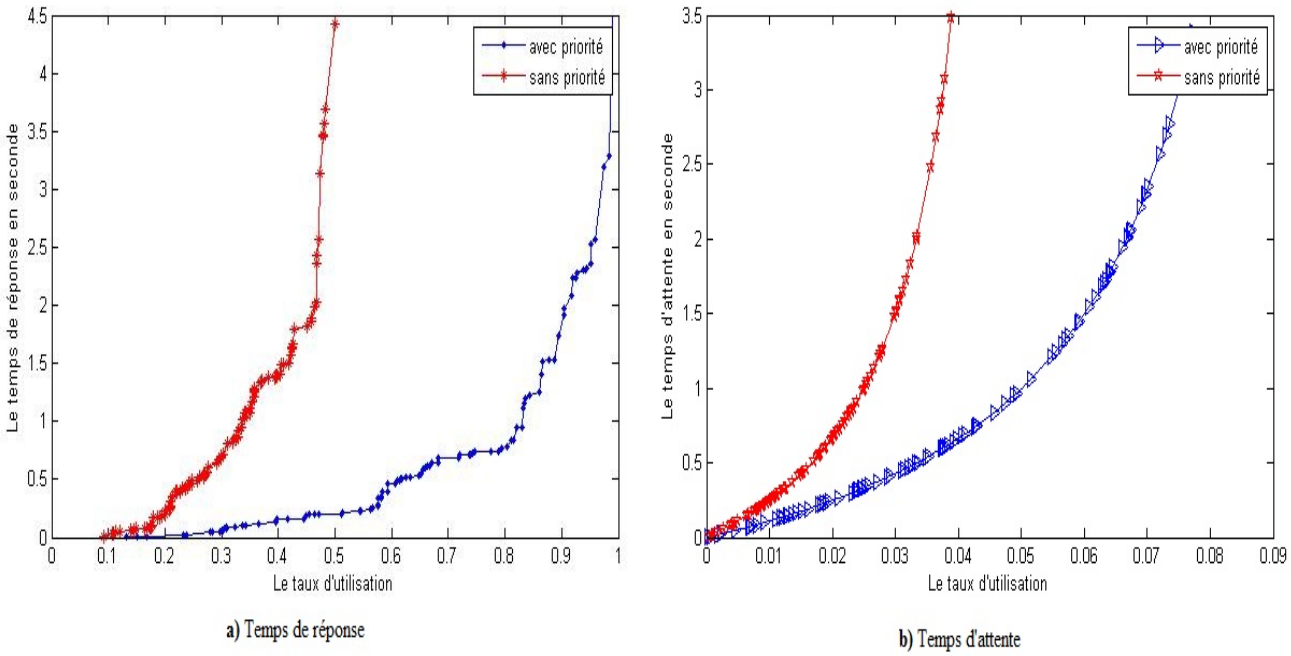


FIGURE 5.4 – *Résultats lorsque N=32.*

## 5.7 Conclusion

Dans ce chapitre, nous avons indiqué la technique de simulation choisie, les outils exploités, le diagramme de déploiement, le déroulement de la simulation et enfin les résultats obtenus. Ces résultats nous ont permis d'affirmer l'optimisation que nous avons apportée en termes de qualité de service dans les architectures structurées (*Chord*).

# Conclusion générale

Ce travail a été réalisé dans le cadre d'un projet de fin de cycle, en vue d'obtenir le diplôme de master recherche, il se résume en l'optimisation des architectures P2P-SIP pour la téléphonie IP.

Nous avons tout d'abord présenté les différentes approches des architectures P2P. Nous avons ensuite, expliqué l'intérêt et les avantages d'utiliser le protocole de signalisation SIP pour la transmission de la voix sur IP. Par ailleurs, nous avons cité les différentes architectures P2P-SIP proposées et c'est ainsi que nous avons pu les classifier, résumé dans un tableau comparatifs, puis soulevé les différents problèmes rencontrés par P2P-SIP existants.

Après avoir étudié en détails l'état de l'art de chaque notion, nous avons déterminé notre problématique qui se résume en qualité de service dans les architectures P2P-SIP structurées et nous avons pris en compte la qualité de service des appels. Nous avons pris en considération en particulier le délai de transmission lors d'un appel d'urgence.

Nous avons résolu la limite soulevée, en modélisant notre système par un réseau de files d'attentes, où chaque file d'attente représente un nœud dans notre réseau et que les requêtes des appels d'urgence passent à la tête de la file pour qu'elles puissent être traitées en premier.

L'évaluation des performances de notre système a été effectuée en utilisant la technique de la simulation, et nous avons choisis Matlab comme un langage de programmation. Grâce à sa rapidité, sa simplicité et ses librairies que l'on juge assez efficace et robuste pour notre simulation

En guise de perspective, la solution proposée peut être améliorée en minimisant le nombre des messages (*overheads*) circulant dans le réseau dans le but de garantir son maintien, en réduisant aussi la taille des tables de raccourcis afin de minimiser la capacité mémoire des nœuds. En résumé, on souhaiterait résoudre le problème de la surcharge de la bande passante et des nœuds du réseau, nous souhaitons aussi sécurisé les appels effectués entre différents utilisateurs.

# Bibliographie


[1] Client/Serveur. http ://www.mosaique-info.fr/glossaire-web-referencement-infographie-multimedia-informatique/c-glossaire-informatique-et-multimedia/218-client-serveur-definition.html. Accessed : 01/03/2016.

[2] Evaluation des performances. https ://hal.inria.fr/inria-00070409/file/RR-5598.pdf. Accessed : 04/06/2016.

[3] Notion hashage. https ://www.tbs-certificats.com/FAQ/fr/sha256.html. Accessed :01/03/2016.

[4] Notion Pastry et chord. http ://www.cs.toronto.edu/pub/eric/SREIS02-Sec.pdf. Accessed :01/03/2016.

[5] *Protocole SIP*. www.protocolesip.com. accessed : 27 février 2016.

[6] J.Stribling et les autres B.Y.Zhao, L.Huang. *Tapestry : a resilient global-scale overlay for service deploym. IEEE*, 22(1), 2004.

[7] J-F.Bouchaudy C.Hunt. *Les réseaux*, volume 5. Paris, eyrolles edition, août 2006.

[8] C.Jampathom. *P2PSIP Security*. Master's thesis, Helsinki university of abstract of technology, juin 2008.

[9] B.Lowerkamp et C.Jennings D.Bryan. *dSIP : A P2P Approach to SIP Registration and Resource Location. Inc*, 2007.

[10] D.Willis et les autres D.Bryan. *Concepts and Terminology for Peer to Peer SIP. IETF*, 2007.

[11] H.Sinnreich et A.B.Johnsten. *Internet communications using SIP : Delevering VoIP and Multimedia Services with Session Initiation Protocol*, volume 2. Wiley Publishing, Inc., Canada, 2006.

[12] D.Bryan et B.Lowerkamp. *SOSIMPLE : A Serverless, Standards-based, P2P SIP Communication System. IEEE*, 2005.

[13] S.Kim et J.Suk. *Efficient peer-to-peer context awareness data forwarding scheme in emergency situations*, volume 9, pages 477–486. Springer, 2016.

[14] F.Salque et X.Bruns. *La téléphonie sur IP*. cours, Université de Pau et des Pays de l'Adour, France, Decembre 2004.

[15] K.Singh H.Schulzrinne. *Peer-to-Peer Internet Telephony using SIP. IJCA Journal*, 11, 2003.

[16] K.Singh H.Schulzrinne. *SIPPEER : A Session Initiation Protocol (SIP)-based Peer-to-Peer Internet Telephony Client Adaptor. IJCSIS*, 2005.

[17] K.Singh H.Schulzrinne. *Using an External DHT as a SIP Location Service. IEEE*, 2429, 2006.

[18] D.Karger M.F.Kaashoek H.Balakrishnan I.Stoica, R.Morris. *Chord : A Scalable Peer-to-peer Lookup Service for Internet Applications*. In *SIGCOMM 01 : Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, volume 31, pages 161–172, New York, USA, Octobre 2001. ACM Press.

[19] G. Cmarillo A.Johnston J.Peterson R.Sparks E. Schooler J.Rosenberg, H. Schulzrinne. *Session Initiation Protocol*. Technical report, Request For Comments 3261, IETF, June 2002.

[20] L.G.Erice. *Un Réseau P2P Hiérarchique : Design et Applications*. PhD thesis, 2006.

[21] G.Pujolle L.Ouakil. *Téléphonie sur P : SIP, H.323, MGCP, QOS et sécurité, Asterisk, Viowifi, offre multiplaydes FAI, Skype et autre Software, architecture IMS*, volume 2. 2008.

[22] M.Amad. *Découverte et localisation de services en mode P2P*. Mémoire de magister en informatique, Université Abderrahmane Mira, Bejaïa, Novembre 2005.

[23] A.Krekelberg N.S.Good. *Usability and privacy : a study of KaZaA P2P file-sharing. CHI letters*, 5(1), 2003.

[24] S.Vaishnav. *Voice Over The Internet Protocol (VOIP) : the dynamics of techonology and regulation*. Master's thesis, Massachusetts Institute of Technology of Colorado State University, USA, June 2006.

[25] D.Stutzbach et R.Rejaie S.Zhao. *Characterizing files in the modern Gnutella network : a measurement study*. In C.Griwodz et Sose S.Chandra, editor, *SPIE*, volume 6071. CA, 2006.

[26] T.Guillet. *Sécurité de la téléphonie sur IP*. Thèse de doctorat en informatique et réseaux, Telecom ParisTech, Paris, Octobre 2010.