Table des matières

Liste des figures

Liste des tableaux

Liste des algorithmes

Introduction générale

DEPUIS l'invention du premier téléphone par Alexandre Graham Bell en 1869, la téléphonie n'a cessé d'évoluer : de la commutation de circuit à la commutation par paquet, d'un réseau fixe à un réseau mobile. Plusieurs architectures ont été crées où la voix est combinée aux données pour être transportées sur un seul réseau *data*. La convergence entre la téléphonie et les réseaux informatique a donnée ce qui est connu par la ToIP (pour *Telephony over Internet Protocol*), ou VoIP, (pour *Voice Over Internet Protocol*).

Les premières technologies de VoIP conçues étaient propriétaires et donc très différentes les unes des autres. Outre les protocoles de transport qui acheminent les paquets sur le réseau, comme RTP *(Real-time Transfer Protocol)*, un système qui est censé mettre des gens et des systèmes en relation exige une certaine mesure de standardisation. C'est pourquoi sont apparus des protocoles standards de signalisation, comme H.323 ou SIP. SIP peut être vu comme étant dérivé d'HTTP dont il reprend de nombreuses caractéristiques. Son but est de créer, modifier ou terminer des sessions avec un ou plusieurs participants. Le protocole tire son succès de sa simplicité. Cependant, il est devenu le protocole le plus utilisé dans la VoIP. Ce qui nous a amené a s'intéressé au développement de ce protocole.

Contrairement au routage IP classique, le routage SIP (de niveau applicatif) consiste simplement à relayer les messages de signalisation SIP à travers une succession de serveurs ou proxy SIP (indépendamment de la route suivie par le flux, envoyé en direct entre l'émetteur et le récepteur sans passer par une succession de proxy). Cette méthode n'est pas direct et donc pas optimale dans certaine conditions, tel que la défaillance d'un serveur proxy, ou la détection de NATs ou de Pare-feu...etc.

Historiquement, tous les services offerts sur Internet étaient développés selon le modèle de réseau client/serveur alors que les nouveaux services sont développés selon le modèle de

réseau Pair à Pair. Un système P2P représente un réseau overlay dont les nœuds (pairs) sont équivalents en fonctionnalité ; c'est-à-dire, ils peuvent à la fois demander et fournir des services. Chaque pair établit une connexion avec un ensemble d'autres pairs et il n'y a pas de contrôle central ni hiérarchique de tout le système. Cette décentralisation entraîne un avantage primordial de P2P : l'agrégation des capacités des pairs pour augmenter la performance globale du système. Par ailleurs, un système P2P permet les entrées et sorties dynamiques des pairs. Le manque de contrôle centralisé cause des difficultés pour le routage et la recherche d'objets. Les systèmes pair-à-pair structurés résolvent ces problèmes en définissant une structure de connexion entre les pairs. La connexion structurée limite le coût de routage (souvent à un nombre logarithmique de sauts) mais complique la maintenance du réseau.

Dans le but d'éviter les problèmes du routage SIP cités précédemment, tel qu'il est déployé sur une architecture client/serveur, notre travail consiste à voir si des mécanismes de routage de type réseau P2P nous permettrai de contribuer à une optimisation du "routage" SIP de niveau applicatif, et ce dans un contexte dynamique (des nœuds quitte le réseau, d'autre rejoignent le réseau... etc)

Ce mémoire est organisée en trois chapitres : Dans le première chapitre, nous présentons tout d'abord des généralités sur la téléphonie IP, ainsi que les différentes architectures possible de la ToIP. Ensuite, nous citons les principaux protocole et standards liés à cette technologie, en particulier le protocole SIP, l'objet de notre étude, qui à pris une grande place dans ce chapitre. Et nous terminons par la définition de la qualité de service et de quelques problèmes liés à la sécurités de la ToIP.

Le deuxième chapitre est consacré à la présentation des réseaux Pair à Pair, qui grâce à ces propriétés, ce paradigme présente une bonne solution pour l'optimisation de la signalisation SIP. Ainsi, nous nous intéressons aux réseaux pair-à-pair structurés qui sont basés sur les DHT, et nous détaillons le protocole Chord.

Le troisième et dernier chapitre porte sur l'optimisation du routage SIP sur une architecture Pair à Pair, où nous présentons deux travaux antérieur sur : P2P-SIP et SOSIMPLE. Ensuite, nous proposant une solution pour optimiser le routage SIP en minimisant le nombre de messages échangés, et tout ceci en se basant sur le principes du protocole Chord. Nous terminons par une conclusion générale et perspectives, dont nous indiquons les axes futurs de développement.

Chapitre 1

Architecture et protocoles de la téléphonie sur IP

1.1 Introduction

Depuis les années quatre-vingt, les éditeurs de logiciels cherchent à utiliser le réseau informatique pour y véhiculer de la voix. Suite à l'apparition de protocoles comme H.323 ou SIP, les constructeurs d'autocommutateurs ont progressivement intégré cette dimension IP à leurs solutions dans une optique de convergence voix-données (figure 1.1). Dans un premier temps, cette convergence a pris la forme de cartes optionnelles à intégrer dans les commutateurs privés (PABX : *Private Automatic Branch Exchange*) existants, pour être proposée aujourd'hui de façon native. C'est ce qu'on appelle la Téléphonie sur IP (ToIP).

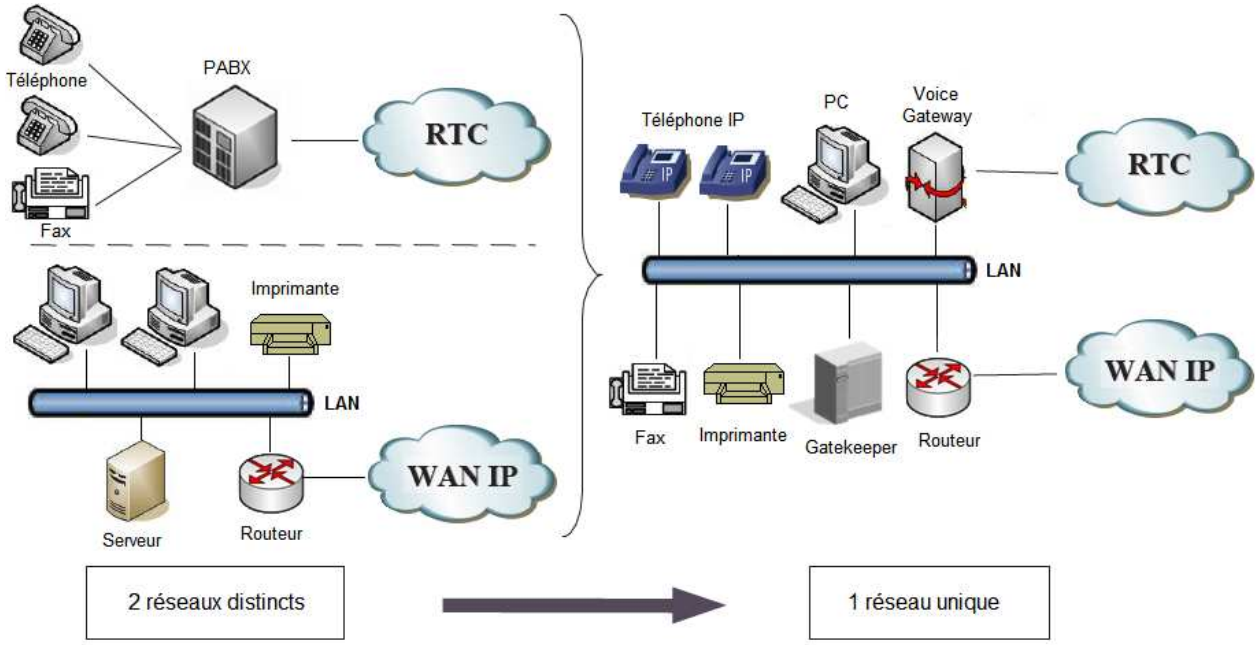


Fig. 1.1 – Schéma de convergence des réseaux

Le transport de la voix et des informations de contrôle (appeler, décrocher, mettre en pause, raccrocher, etc) sont deux flux distincts qui n'ont pas besoin d'être acheminés par les mêmes canaux ; ces deux flux d'informations se voient effectivement séparés dans le RTC que nous connaissons. Cette décomposition est préservée dans les propositions de technologies relatives aux communications vocales sur Internet, où l'on voit des protocoles dédiés au transport de données temps réel (RTP), mais également des protocoles dédiés à la communication d'informations de contrôle, comme c'est le cas du protocole de signalisation SIP.

Dans ce chapitre, nous donnons quelques généralités sur la téléphonie sur IP. Ensuite nous présentons deux protocoles de signalisation concurrents, SIP et H.323. Enfin, nous terminons par la définition de la qualité de services et de la sécurité dans la VoIP.

1.2 Généralités sur la téléphonie sur IP

Bien évidemment, la plupart des téléphones sont encore connectés aux réseaux téléphoniques classiques, qui reposent sur une technologie de commutation de circuits bien antérieure à l'informatique et aux réseaux de données, mais robuste et présentant une forte disponibilité (99.999%, appelée "règle des cinq neufs") [5]. Les services de téléphonie sur IP doivent donc pouvoir accepter tout trafic provenant de ces réseaux et assurer la terminaison d'une communication sur le RTPC (*Réseau Téléphonique Public Commuté*), le tout en parfaite continuité.

1.2.1 Principe de la ToIP

La téléphonie sur IP est basée sur un double principe[1] :

- Découpage du flux voix numérisé en suite de "paquets" : cette mise en "paquets" prépare le transport de la voix sur des réseaux informatiques en le mettant au format adéquat.
- Transit sur un réseau IP : grâce au "réseau des réseaux" ou "Internet", le protocole IP est devenu la technique de base la plus largement utilisée et disponible pour véhiculer en "paquet" des données entre deux points ou entre deux périphériques d'un réseau.

[1]Livre blanc, Téléphonie sur IP, France Télécom - juin 2004 / CESMO

Un autre principe fondamental dans la téléphonie sur IP est l'échange mutuel de paramètres de communication entre les participants[2] :

- quelle est l'adresse IP où le téléphone peut joindre son correspondant ?
- quels sont les ports TCP/UDP sur lesquels l'application de téléphonie peut transmettre les flux d'information numériques (les médias) qui correspondent à la voix, la vidéo, le texte, etc ?

Notons que les protocoles de transport classiquement utilisés pour transporter les données sont TCP et UDP. Le protocole TCP assure un bon contrôle de l'intégrité des informations transportées (mécanisme d'accusé de réception) mais n'est pas particulièrement performant en termes de délais. UDP est un protocole plus simple que TCP, présentant de ce fait, de meilleures performances moyenne car il permet l'envoi de paquets sans contrôle de réception (pas d'acquittement).

1.2.2 La téléphonie sur IP et le RTC

Le RTPC (*Réseau Téléphonique Public Commuté*, ou simplement RTC) est le réseau téléphonique qui donne accès à de multiple fonction. Le RTC a essentiellement pour objet le transfert de la voix. En effet outre le fait de pouvoir téléphoner, le RTC permet d'utiliser de multiples services tel que la transmission et réception de fax, l'utilisation d'un minitel, accéder à Internet ...etc. On distingue deux grandes parties dans ce réseau :

1. Le réseau capillaire ou de distribution, c'est le raccordement depuis chez l'abonné à un point d'entrée du réseau. Cette partie du réseau est analogique.

2. Le réseau de transit, effectue pour sa part le transport des communications entre les nœuds de transit (concentrateurs/commutateurs). Cette portion du réseau est actuellement numérique.

Le transport de la voix sur un réseau IP et sur le RTC sont deux opérations radicalement différentes. Sur le RTC, la voix est acheminée sous forme d'un flux constant, alors que sur le réseau IP, ce n'est pas du tout le cas. Le transport sur ce dernier se fait en découpant la voix en paquets au départ, à envoyer ces paquets avec le protocole adéquat sur le réseau IP et à tout reconstitué à l'arrivée. la figure 1.2 montre l'interconnexion entre un réseau IP et le réseau RTC.

[2]Le Rapport essentiel sur la téléphonie sur IP, par le Groupe d'experts sur la téléphonie sur IP / UIT-D

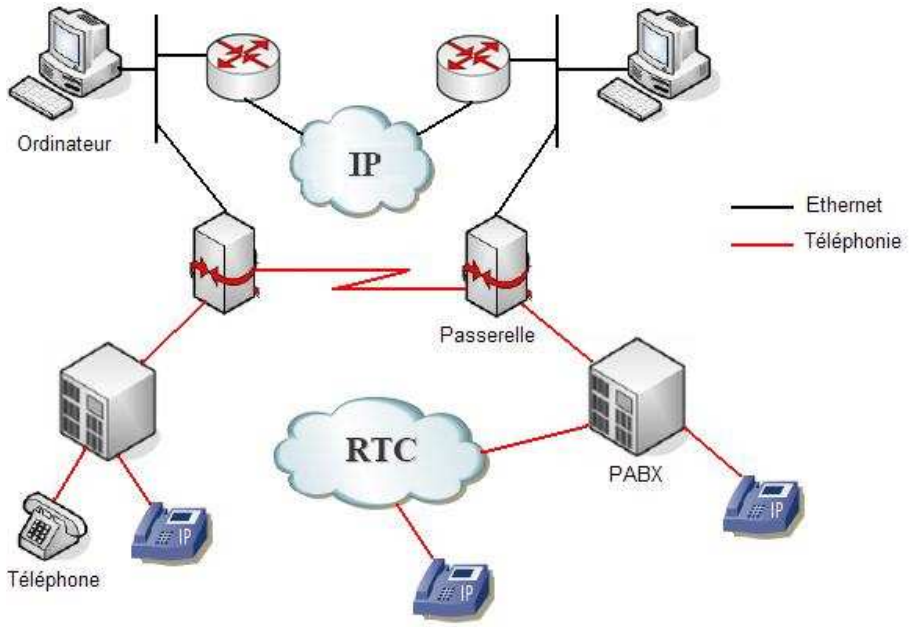


FIG. 1.2 – La téléphonie sur IP et le RTC

1.2.3 Voix sur IP vs téléphonie sur IP

La ToIP (*Telephony over Internet Protocol*) consiste en un ensemble de techniques qui, dans une entreprise ou un organisme, permettent la mise en place des services téléphoniques en utilisant les standards de la voix sur IP (VoIP) pour la transmission de messages vocaux sur des réseaux de données IP. La ToIP est gérée par une entité unique, une entreprise pour ses besoins internes ou un opérateur telecom.

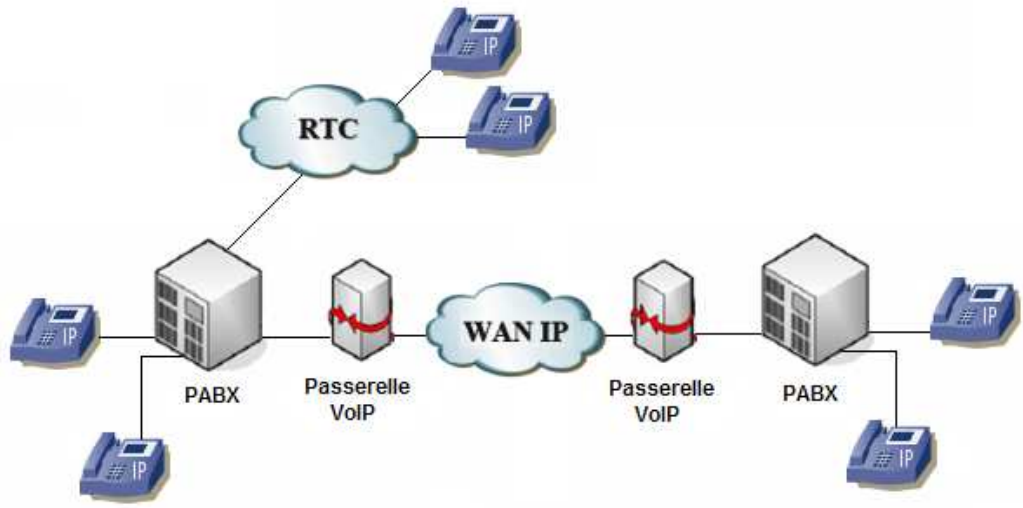


FIG. 1.3 – La VoIP

Dans le cas de la VoIP, on se contente d'interconnecter des PABXs en encapsulant la voix numérisée, dans les paquets IP (figure 1.3). Ces derniers sont ensuite véhiculés au sein du réseau de données, de manière classique, comme des paquets de données. La voix est simplement «reconstituée» lorsque les paquets arrivent chez le destinataire. Un exemple «parlant» de VoIP se trouve dans le raccordement de deux sites via une ligne spécialisée qui transporte la voix et les données.

Quant à la ToIP (figure 1.4), elle va plus loin que la VoIP en terme de mécanismes et d'équipement, cherchant à apporter aux utilisateurs la qualité de service (qualité de transmission, qualité de voix, disponibilité du service) et les services que ces derniers ont l'habitude de trouver du côté de la téléphonie classique (présentation du numéro, transfert d'appel, conférence, etc.) le tout sans faire appel aux PABXs traditionnels.

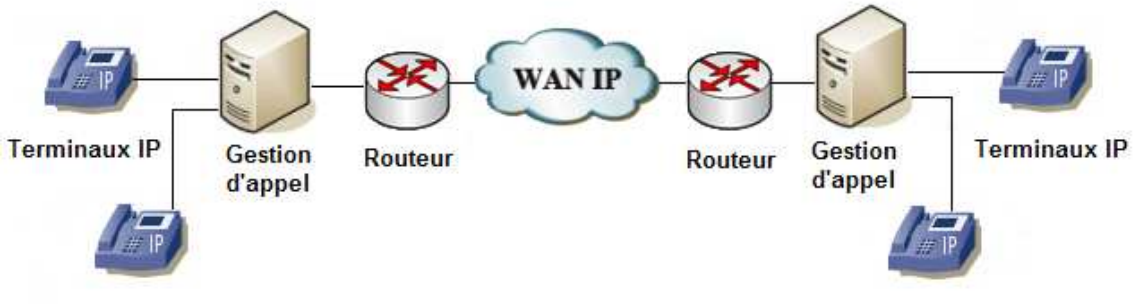


FIG. 1.4 – La ToIP

Il ne faut pas confondre ToIP et Voix sur IP (VoIP), cette dernière correspond plutôt aux technologies de transmission de la voix sur un réseau informatique. En ce sens, la VoIP peut être vue comme un des composants de la ToIP et sa sécurisation est, par conséquence, la condition qui doit être obligatoirement satisfaite pour la sécurisation de la ToIP.

1.3 Architecture de la ToIP

Dans cette section, nous présentons les différents éléments qui composent l'architecture d'une solution VoIP.

1.3.1 Les éléments de bases d'une architecture de ToIP

Le réseau

Le réseau interconnecte les différents équipements qui participent à une solution de voix sur IP. A la différence du réseau téléphonique, la signalisation et la voix sont transportés sur le même réseau partagé. Comme pour IPsec, des solutions alternatives ont dû être développées pour gérer les contraintes introduites par la traduction d'adresses (NAT).

Les terminaux IP

Un terminale IP peut se présenter sous deux formes : soit un téléphone classique (*hard-phone*), soit un téléphone logiciel (*softphone*) qui s'exécute sur l'ordinateur de l'utilisateur. En terminologie SIP c'est un UA (User Agent).

Les systèmes

La liste des protocoles utilisés dans les solutions de téléphonie et de voix sur IP est conséquente. Il en va de même de celle des systèmes.

Serveurs SIP : proxy, redirect et registrar

Le *SIP proxy* joue le rôle de relais et fait suivre une requête SIP au prochain serveur. Le *SIP redirect server* renvoit une réponse au client contenant le prochain serveur à contacter. Le *SIP registrar* enregistre le lien entre l'adresse IP et l'adresse SIP.

Le Call Manager (CM) / IP-PBX

Le CM fournit des fonctions de base de gestion d'appel, des utilisateurs et des configurations, et également des fonctionnalités avancées comme la conférence, les boîtes vocales, etc. Il peut être vu comme un IP-PBX. A ne pas confondre avec des PBX traditionnels (pas de VoIP) que l'on peut administrer à distance via une connexion TCP/IP (qui remplace la connexion locale ou de télé-maintenance via un modem).

Les passerelles

Une passerelle s'occupe des échanges entre le réseau voix sur IP et le réseau téléphonique classique. Cela comprend la conversion de la voix et de la signalisation.

1.3.2 Scénarios possibles pour la ToIP

Selon le type de terminal utilisé, on distingue trois Scénarios possibles de téléphonie sur IP :

Téléphonie de PC à PC

Dans ce scénario (figure 1.5), les deux correspondants utilisent un PC rattaché au réseau Internet par l'intermédiaire d'un fournisseur d'accès à Internet (IAP). Cette technique nécessite des participants à la communication d'avoir un logiciel de téléphonie sur IP compatible de chaque côté. Ce mode de fonctionnement nécessitait auparavant que les correspondants se fixent un rendez-vous préalable sur Internet ou soient connectés en permanence. De nos jours, des protocoles de signalisation ont été élaborés pour pallier à ce genre de problème.

FIG. 1.5 – Téléphonie de PC à PC

Téléphonie entre PC et poste téléphonique et vis-versa :

Dans ce scénario (figure 1.6), l'un des correspondants utilise un PC rattaché au réseau Internet par un fournisseur d'accès à Internet, l'autre utilise un téléphone rattaché au réseau téléphonique commuté. Une passerelle est nécessaire entre les deux réseaux pour faire la conversion Internet-RTC et vice versa. Elle se charge également de l'appel du correspondant et de l'ensemble de la signalisation relative à la communication téléphonique du côté du correspondant demandé. Du côté PC, une signalisation d'appels est nécessaire pour établir une communication et négocier les paramètres de communication multimédia.

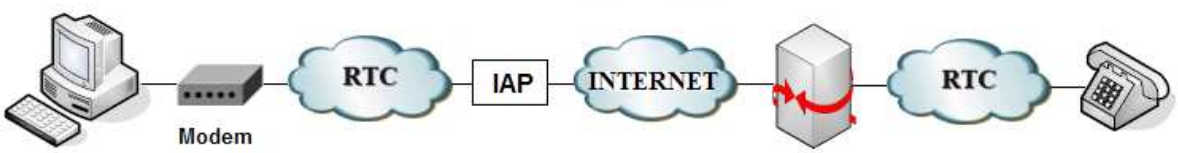


FIG. 1.6 – Téléphonie entre PC et poste téléphonique

Téléphonie entre postes téléphoniques :

Dans ce cas (figure 1.7), les deux correspondants utilisent un téléphone conventionnel via le réseau téléphonique commuté. Une passerelle est utilisée de chaque côté entre ce réseau et le réseau Internet pour convertir la voix sur IP en voix et vis-versa. Le réseau Internet est utilisé pour la connexion longue distance.

FIG. 1.7 – Téléphonie entre postes téléphoniques

1.3.3 Normalisation et standardisation de la ToIP

La normalisation technique de la téléphonie sur IP est en cours dans le cadre de nombreuses entités industrielles et d'organismes de normalisation tels que le secteur de la normalisation des télécommunications de l'IUT (IUT-T), le secteur des radiocommunications de l'IUT (IUT-R), et le groupe d'étude sur l'ingénierie Internet (IETF).

Un exemple de normalisation dans le cadre de l'IUT est la série de recommandations H.323 pour les champs suivants [33] : audioconférence, visioconférence multimédia, établissement et commande d'appel, gestion de la bande passante, interfaces entre différentes architecture réseaux, et le protocole d'initiation de session SIP défini et standardisé par l'IETF pour la conférence, la téléphonie, la détection de présence, la notification d'événements et la messagerie instantanée. Ces protocoles seront détaillés dans les sections suivantes.

Dans certains cas, l'IETF et l'UIT-T ont collaboré directement à la normalisation de la téléphonie sur IP [14], c'est ainsi qu'ils ont mis au point le protocole commun H.248 (nom donné par l'UIT-T) ou MEGACO (nom donné par l'IETF). Ce protocole définit les relations maître/esclave de gestion des passerelles médias qui assurent l'acheminement de différentes formes de trafic - voix, vidéo, télécopie et données - entre le RTPC et les réseaux IP.

1.4 Les standards de la ToIP

Il existe plusieurs standards et normes de téléphonie sur IP [33]. Certains sont ouverts : c'est-à-dire publiés et librement utilisables (non propriétaires). D'autres sont fermés : c'est-à-dire non entièrement publiés et/ou non librement utilisables. Les protocoles ouverts de signalisation les plus connus et les plus utilisés actuellement dans les réseaux de téléphonie sur IP sont SIP et H.323. La signalisation est indispensable pour établir une communication téléphonique. Elle permet dans un premier temps d'envoyer des messages avant la communication, d'avertir l'utilisateur et de connaître la progression de l'appel et enfin de mettre un terme à la communication.

1.4.1 Le standard H.323

H.323, défini par l'ITU-T en 1996, est le premier protocole développé pour permettre l'établissement, la libération et la modification de sessions multi-média (voix, vidéo, données). Les protocoles de signalisation spécifiés par H.323 sont :

- H.225.0 RAS (Registration, Admission, Status) : utilisé pour communiquer avec les GateKeepers, notamment pour découvrir d'autre GateKeeper et s'enregistrer auprès de l'un d'entre eux à la connexion d'un terminal afin de permettre le routage des appels.
- H.225.0 (Q.931) : Version simplifiée de la signalisation RNIS Q.931 pour l'établissement et le contrôle d'appels téléphoniques sur IP. Les signaux sont routés à travers un GateKeeper, si ce dernier existe. Sinon, ils sont rourés directement d'un terminal à un autre.
- H.245 : Permet le contrôle de l'ouverture et de la fermeture des canaux pour les médias et la négociation de formats (par exemple quel type de Codec utiliser), ou encore de mesurer le délai d'aller-retour d'une communication.

L'architecture de H.323 se compose de 4 éléments réseaux [7] : Terminal, Gatekeeper, MCU et Passerelle (figure 1.8). Les Terminaux peuvent se connecter directement entre eux ou grâce aux autres composants.

1. **Les terminaux :** Ce sont les appareils connectés au réseau : un téléphone IP, un ordinateur équipé d'un logiciel supportant le protocole H.323... Ils se regroupent en 2 catégories : matériel et logiciel.
2. **Les Gateways (Passerelles) :** Équipement réalisant l'interconnexion entre réseaux

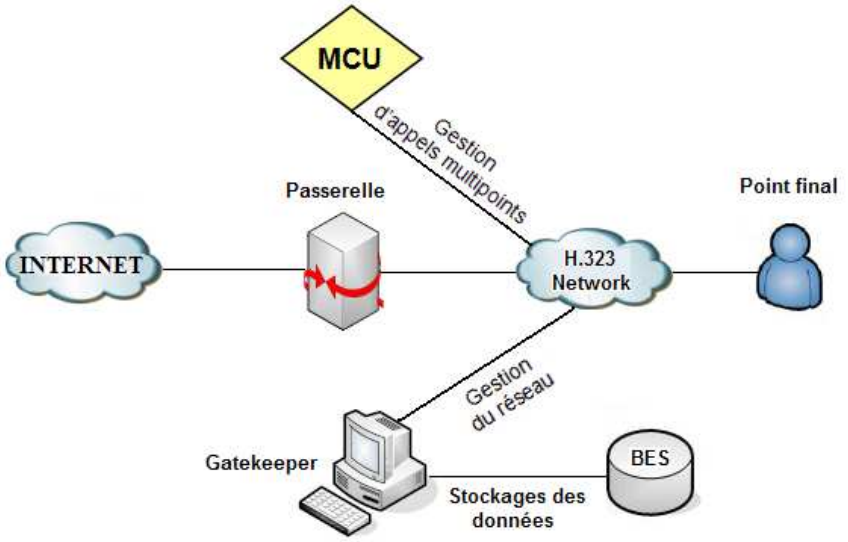


FIG. 1.8 – Architecture du standard H.323

hétérogènes à l'aide de cartes spécifiques. Elles assurent la conversion des données et des signaux de contrôle, et la cohésion des média. Pour cela, elle implémente les fonctions de transcodage audio (compression, décompression), de modulation, démodulation (pour les fax), de suppression d'écho, de silence et de contrôle d'appel. Bien souvent, il s'agit d'un ordinateur, mais on les trouvent également intégrées dans des routeurs ou bien des PABX.

3. **Les GateKeepers** (Logiciel des passerelles) : Élément optionnel réalisant la traduction d'adresse (n° de téléphone vers adresse IP et réciproquement) et la gestion des autorisations. Cette dernière permet d'établir ou non un appel, de limiter la bande passante ou encore de gérer le trafic sur le réseau. Il permet également de gérer les téléphones classiques et la signalisation permettant de router les appels afin d'obtenir des services supplémentaires tel qu'un service d'annuaire, le transfert d'appel...
4. **Les unités de contrôle multipoint** (MCU, Multipoint Control Unit) : Gère les conférences entre plusieurs terminaux. Elles peuvent communiquer entre elles pour s'échanger des informations de conférence. Elles sont composées d'un contrôleur multi-point (Multipoint Processor : MC) et de 0 à plusieurs processeur multi-point (Multipoint Processor : MP). Le premier gère la signalisation entre les participants, tandis que le second s'occupe du mixage et du traitement des données.

Le protocole H.323 reste un protocole complexe qui s'inspire de la téléphonie. Il a été créé initialement pour les conférences, et se retrouve aujourd'hui avec des mécanismes superflus, nécessitant une capacité de traitement et de mémoire plus importante sur les terminaux.

1.4.2 Le protocole SIP

SIP (Session Initiation Protocol) [29], est le standard IETF (Internet Engineering Task Force) pour la signalisation de communications multimédias interactives , A savoir : établissement, redirection, relayage, terminaison de sessions vidéo ou audio entre plusieurs terminaux, sur un réseau à commutation de paquet. Il se base entre autre sur le protocole HTTP, la structure de l'entête est semblable et la transmission se fait également en mode texte.

La première version de SIP [5], SIPv1, a été soumise à l'IETF comme une ébauche d'Internet en février 1996. *SIPv1* a employé le protocole SDP (*Session Description Protocol*) pour décrire les sessions et UDP comme protocole de transport. SIPv1 a été basé sur du texte.

Également en février 1996, Henning Schulzrinne a soumis une ébauche d'Internet à l'IETF spécifiant SCIP (*Simple Conference Invitation Protocol*). *SCIP* était également un mécanisme pour inviter les utilisateurs aux sessions point à point et multicast. Il a été basé sur le protocole HTTP (*Hypertext Transfer Protocol*) et a ainsi utilisé TCP (*Transmission Control Protocol*) comme protocole de transport. Comme SIPv1, SCIP a été basé sur du texte.

Le protocole résultant des deux précédents a gardé comme nom SIP, mais a changé la signification de l'acronyme en *Session Initiation Protocol* et a avancé la version au numéro2 (figure 1.9). *SIPv2* a été basé sur le protocole HTTP, mais a pu utiliser UDP et TCP comme protocoles de transport. Il a employé SDP pour décrire des sessions multimédia, et c'était basé sur du texte. Ce protocole reste la version en cours du protocole SIP. Les efforts de développement de SIP étaient le domaine du groupe de travail MMUSIC (*Multiparty Multimedia Session Control*), présidé par Joerg Ott et Colin Perkins.

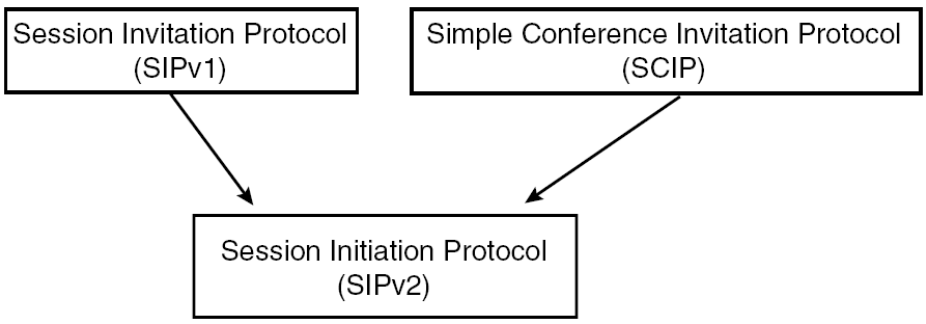


FIG. 1.9 – SCIP et SIPv1 ont été fusionnés dans SIPv2

1.4.2.1 Les messages SIP : requêtes et réponses

Le protocole SIP est réalisé par échange de messages codés en mode texte (UTF8) et ayant une sémantique similaire à celle des messages du protocole HTTP [5]. Chaque message est composé d'une adresse (URI SIP), d'un ensemble d'en-têtes, et d'un corps. Les messages SIP sont échangés entre deux entités sur un *mode client-serveur* : l'entité appelante envoie un message (*une requête*) a l'entité appelée. Cette dernière répondra par un autre message (*une réponse*). Dans ce contexte, l'appelant est considéré comme le *client*, l'appelé comme le *serveur* et l'échange de message comme une *transaction*.

Sémantique des requêtes

Les requêtes SIP constituent les messages qualifiés par un performatif (invitation, demande d'information, souscription et publication d'information, etc) appelé *méthode*, définissant la qualité de la transaction. la figure 1.10 représente le format des requêtes SIP :

INVITE sip :user@domain.com
Via : Call-ID : From : To : Call-ID : Cseq :
Subject :
Content-Type : Content-Length :
Ligne vide
Données SDP

FIG. 1.10 – Allure d'une requête SIP

Afin de mieux saisir les possibilités offertes par SIP, voici une liste des principales méthodes définies par ce protocole et utilisés pour qualifier les requêtes :

- **INVITE :** Invite une entité SIP à entrer en communication. Cette requête permet d'initier une session.
- **ACK :** Accusé de réception, utilisé pour confirmer une réponse à une requête INVITE, dans certains cas pour s'assurer de la clôture d'une transaction.
- **PRACK :** Accusé de réception pour confirmer une réponse provisionnelle.
- **BYE :** Clôture de la session (fin de communication ou refus d'un appel).
- **CANCEL :** Annulation d'une session en instance.

- **REGISTER :** Enregistrement de l'URI d'une entité SIP.
- **OPTIONS :** Demande de précision sur les options et les capacités d'une entité SIP.
- **REFER :** Redirection d'appels vers une URI.
- **SUBSCRIBE :** Souscription à un évènement.
- **NOTIFY :** Publication d'un évènement.
- **MESSAGE :** Envoi d'un message instantané.
- **INFO :** Information de session en cours.
- **UPDATE :** Modification des informations de session.

Sémantique des réponses

Les réponses constituent les informations renvoyées par le serveur au client, et concernent autant l'évolution de la transaction que les erreurs pouvant survenir (transport, serveur, client, etc). On distingue les réponses *provisionnelles*, qui donnent une information optionnelle, et les réponses *finales* qui clôturent une transition. La figure 1.11 représente le format des réponses SIP :

SIP/2.0 302 Moved temporarily
From : To : Call-ID : Localization : Expires : Cseq :
Ligne vide :
Données de réponse (SDP vide, SDP chiffré, etc)

FIG. 1.11 – Allure d'une réponse SIP

Les réponses reprennent les codes définis dans la norme HTTP. Les codes sont constitués de trois chiffres dont le premier caractérise la classe de réponse. Les types de réponse SIP définis dans RFC 3261 sont dans les catégories suivantes [38, 29] :

- **Provisoires (1xx) :** La requête a été reçue et est traitée.
- **Succès (2xx) :** L'action a été reçue avec succès, comprise, et acceptée.
- **Redirection (3xx) :** Autre action nécessite d'être prise afin d'accomplir la requête.
- **Erreur du client (4xx) :** La requête contient une mauvaise syntaxe ou ne peut pas être accomplie sur ce serveur.
- **Erreur du serveur (5xx) :** Le serveur échoue a accomplir une demande apparemment valide.

- **Échec global (6xx) :** La requête ne peut pas être accomplie sur aucun serveur.

1xx	100 Trying (tentative)	180 Ringing (sonnerie)	181 Call Is Being Forwarded (renvoi d'appel)
	182 Queued (en attente)	183 Session progress (session en cours)	
2xx	200 Ok		
3xx	300 Multiple choices	301 Moved Permanently	302 Moved Temporarily
	305 Use Proxy	380 Alternative Service	
4xx	400 Bad Request	401 Unauthorized	402 Payment Required
	403 Forbidden	404 Not Found (non trouvé)	405 Method Not Allowed
	406 Not Acceptable	407 Proxy Authentication Required	408 Request Timeout
	409 Conflicts	410 Gone	411 Length Required
	413 Request Entity Too Large	414 Request-URI Too Long	415 Unsupported Media Type
	420 Bad Extension	480 Temporarily Unavailable	481 Call Leg/Transaction Does Not Exist
	482 Loop Detected	483 Too Many Hops	484 Address Incomplete
	485 Ambiguous	486 Busy Here (occupé)	
5xx	500 Server Internal Error	501 Not Implemented	502 Bad Gateway
	503 Service Unavailable	504 Gateway Time-out	505 Version Not Supported
6xx	600 Busy Everywhere	603 Decline	604 Does Not Exist Anywhere
	606 Not Acceptable		

TAB. 1.1 – Liste complète des codes de réponse SIP

1.4.2.2 Les adresses SIP

Aujourd'hui la téléphonie sur IP tend à faire exploser le nombre des équipements pouvant jouer le rôle d'un téléphone. Ainsi, il ne sera pas rare de disposer de trois ou quatre équipements que l'on pourra enregistrer pour recevoir des appels. Il devient de plus en plus difficile de contacter une personne puisqu'on ne sait pas quel équipement est apte à recevoir l'appel. La solution est l'utilisation d'URIs [39].

***Définition* 1.** Une **URI** (*Uniform Resource Indicator*) est un identificateur générique [38] qui a le même format que des adresses e-mails. Elle contient principalement un nom d'utilisateur et un nom de domaine (user@domain). Une URI SIP ordinaire est de la forme : `sip :yaniss@hotmail.com`. Si une transmission sécurisé est exigée, "sip :" est remplacé par "sips :". Une URI peut être :

- une **URL** : la ressource est identifiée par sa *localisation*. (exp : l'URL utilisé pour les adresses Web) ;

- ou une **URN** : la ressource est identifiée par son *nom*.

Quand il s'enregistre, l'utilisateur spécifie l'adresse IP de l'équipement sur lequel il est connecté. L'URI de l'utilisateur est ensuite associée à cette adresse IP et il est seulement connu de l'extérieur par cet URI. Voilà quelques exemples typiques d'URIs SIP :

```
sip :yaniss.bkh@hotmail.com
sip :mourad@computer.mcs.com
sip :543578478543@pstn-gateway.com
```

Exemple d'établissement d'appel

La figure 1.12 illustre l'établissement d'une communication entre deux personnes disposant de terminaux SIP, et permet d'observer les transactions qui s'opèrent entre eux.

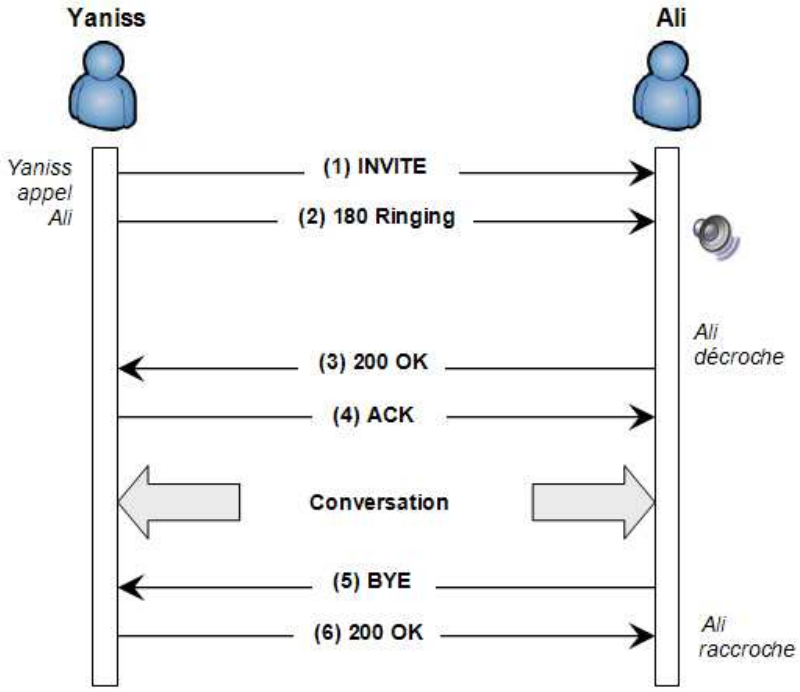


FIG. 1.12 – Initiation d'une session avec SIP

L'exemple suivant montre un message SIP initiant une communication :

```
INVITE sip :ali@172.25.49.134 :5080 SIP/2.0
Via : SIP/2.0/UDP 172.25.49.134 :5070 ; branch=z9hG4bK3563508590
From : <sip :yaniss@hotmail.com> ; tag=2174475352
To : <sip :ali@poste.home.com>
Call-ID : 2188029631@172.25.49.134
CSeq : 20 INVITE
Contact : <sip :yaniss@122.65.19.133>
Max-Forwards : 70
Content-Type : application/sdp
Content-Length : 142
```


Descriptif des différents champs d'un en-tête SIP :

- La première ligne contient le type de message (ici il s'agit d'un message **INVITE**)
- **Via :** ce champ représente le chemin parcouru par la requête, lors de l'envoie de la requête, il contient initialement l'URI de l'émetteur de cette requête. Ensuite, à chaque fois qu'un UA fait suivre la requête celui-ci rajoute son propre URI dans ce champ.
- **From :** URI de l'expéditeur du message
- **To :** URI du destinataire du message
- **Call-ID :** identifiant unique représentant la session SIP
- **CSeq :** contient un entier et un nom de méthode, l'entier est généré aléatoirement une première fois puis est ensuite incrémenté à chaque nouveau message.
- **Contact :** contient une ou plusieurs URI représentant une route directe pour contacter l'UA de l'expéditeur (ici l'UA de Yaniss)
- **Max-Forwards :** nombre maximum de sauts
- **Content-Type :** description du type de média contenu dans le corps du message.
- **Content-Length :** taille en octet du corps du message
- **branch, tag :** utilisés pour des raisons d'identification

1.4.2.3 Architecture et composantes du protocole SIP

L'architecture de SIP comporte deux entités principales : les agents SIP et les serveurs SIP. L'entité avec laquelle l'utilisateur interagit est le *client* qui peut être matériel ou logiciel. Les autres éléments sont des *serveurs* qui forment ce qu'on appelle une *plateforme* de signalisation.

Agents utilisateurs (*UA*)

Il existe plusieurs types d'agents utilisateurs. Ceux-ci peuvent être des périphériques *hardphone* ou encore des logiciels *softphone* interagissant ensemble afin de fournir des services tels que la vidéoconférence, la téléphonie sur IP, les jeux interactifs ou encore les échanges de fichiers. Ces agents utilisateurs (*User Agents*) sont dénommés *UA client* ou *UA serveur* selon qu'ils effectuent ou qu'ils reçoivent une requête d'ouverture de session. Lors d'une ouverture d'une session, une requête est effectuée du UA client au UA serveur (figure 1.13).

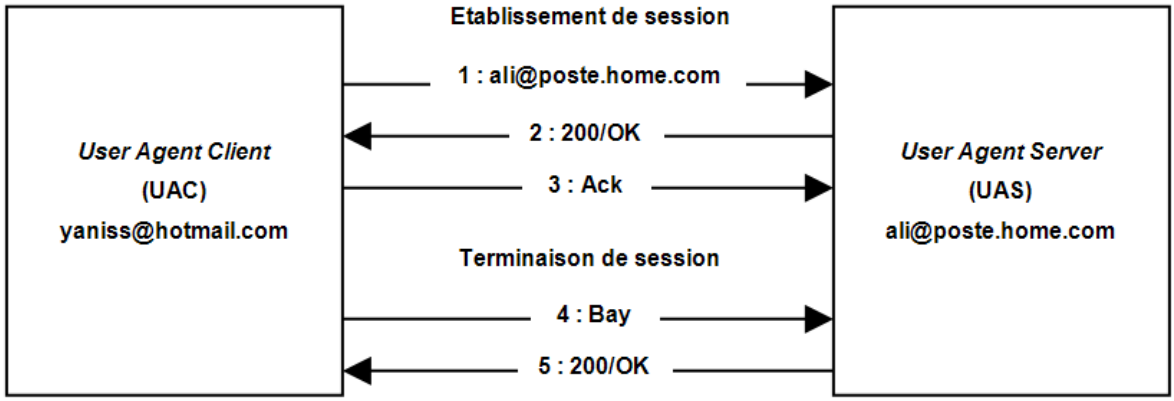


FIG. 1.13 – Établissement et terminaison d'une session SIP entre deux agents utilisateurs.

Serveurs SIP

Il existe quatres types de serveur SIP[3] : Serveur *Proxy* SIP, *serveur de redirection* SIP, *serveur de localisation* et *serveur d'enregistrement*. Ces types de serveur effectuent diverses tâches au sein d'un réseau tel que décrit dans l'article "SIP : Protocol Overview" [10].

1. Serveur Proxy SIP (*Proxy Server*) : Tel que décrit dans le RFC 3261 [29], ce type de serveur est un élément qui transfère les requêtes SIP à un UAS et les réponses SIP à un UAC (voir figure 1.14). Une requête peut traverser quelques serveurs Proxy SIP afin d'être acheminé vers un client. Chacun de ces serveurs va appliquer les règles de routage ainsi qu'effectuer les modifications nécessaires aux diverses requêtes avant de les transférer au prochain élément. Les réponses vont être réacheminer à travers des divers éléments suivant le chemin inverse de la requête.

2. Serveur de redirection SIP (*Redirect server*) : Un serveur de redirection est un serveur qui transfère vers le client l'information concernant le routage de la requête (voir

[3]Généralement le proxy serveur, le serveur d'enregistrement et le serveur de localisation sont combinés en une seule entité et sont physiquement situé sur la même machine du serveur.

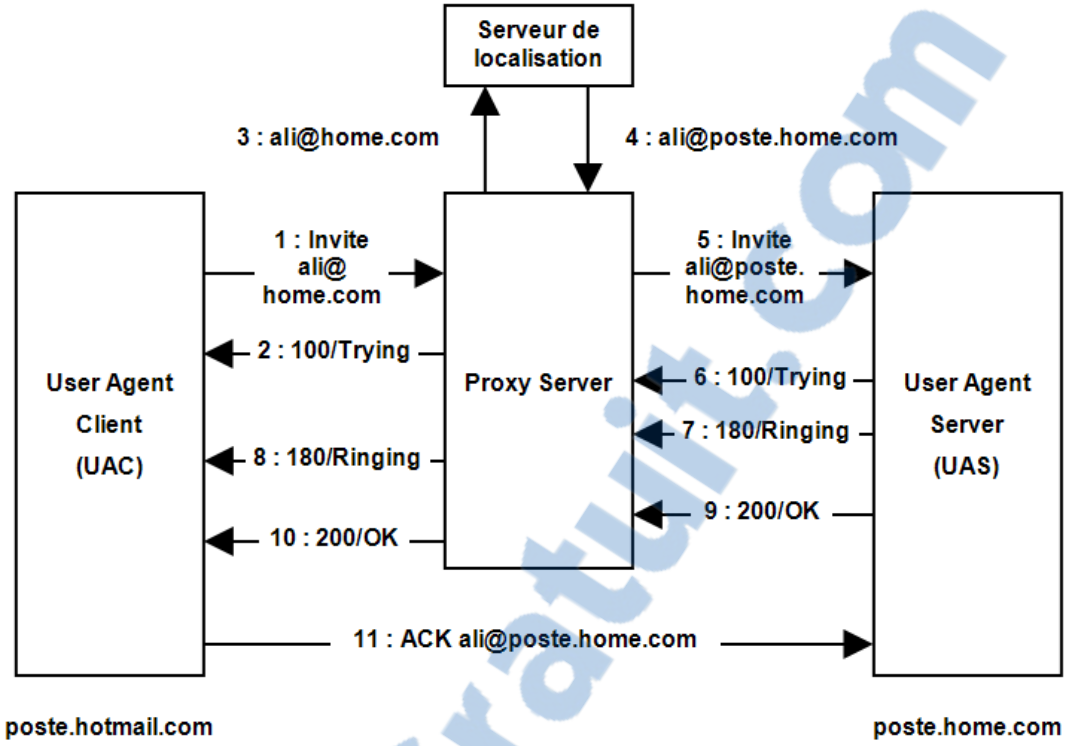


FIG. 1.14 – Établissement d'une session SIP avec l'utilisation d'un serveur proxy SIP.

figure 1.15) afin que celui-ci effectue les requêtes sans passer par le serveur. En faisant de sorte, le serveur n'est plus impliqué dans les échanges futurs de messages entre un agent utilisateur client et serveur.

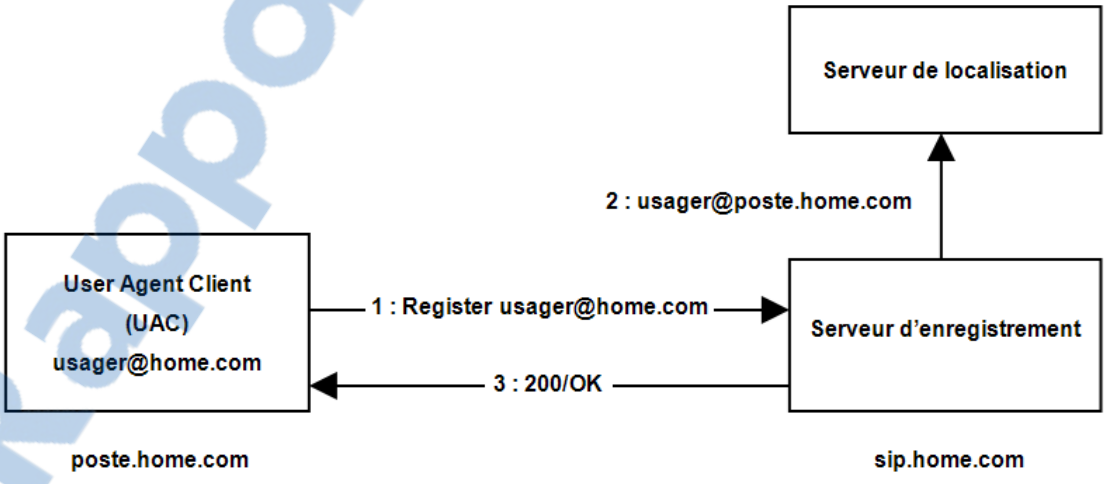


FIG. 1.15 – Enregistrement de la localisation d'un usager auprès d'un serveur d'enregistrement.

3. Serveur de localisation (*location server*) : Le serveur de localisation fonctionne comme une base de données pour les proxy serveurs, les serveurs de redirection et les serveurs d'enregistrement. Il contient une liste des liaisons entre les URIs SIP et les localisations des

UAs (adresses IP).

4. Serveur d'enregistrement (*Registrar*) : Un serveur d'enregistrement est un serveur qui traite les requêtes d'enregistrement afin de mettre à jour la localisation de l'usager dans la base de données, grâce aux informations fournies lors de la requête (voir figure 1.16).

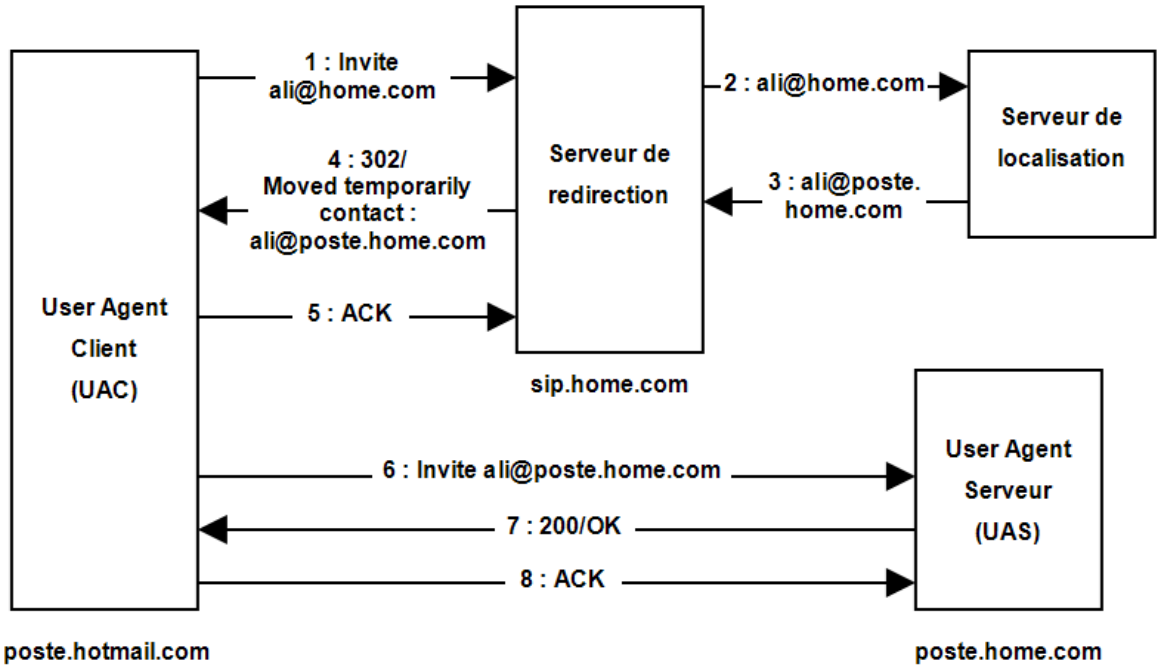


FIG. 1.16 – Établissement d'une session SIP avec l'utilisation d'un serveur de redirection.

1.4.2.4 Le routage de la signalisation SIP

La plupart des protocoles récents tels que SIP ajoutent des nouvelles dimensions multiples au routage de la couche application, tel que la mobilité et la configuration d'utilisateur. Dans son noyau, un contrôleur d'appel SIP (plus connu sous le nom de proxy SIP) est un moteur pur de routage. Il analyse une invitation entrante et prend une décision, après prise en compte de différents facteurs, sur la meilleure façon pour router le message.

Description du mécanisme

SIP peut fonctionner au dessus de TCP ou d'UDP. Afin de palier aux pertes éventuelles de messages possibles si UDP est utilisé, SIP doit donc avoir ses propres mécanismes de retransmissions. Les requêtes émises par les UAC sont donc retransmises périodiquement jusqu'à réception d'une réponse de l'UAS. Plus spécifiquement pour les requêtes de type

INVITE, l'UAC réémet la requête au bout d'une durée initiale de T1 secondes qui double à chaque réémission, tel qu'il est montré par la figure 1.17. L'UAC cesse de réémettre s'il reçoit une réponse de l'UAS ou s'il a réémit le paquet 7 fois. Un mécanisme similaire est utilisé par l'UAS. Pour initier une session SIP, l'UA doit simplement connaître l'adresse URI de l'UA qu'il désire contacter. Un message INVITE est envoyé en direction de cette URI et sera relayé par un ou plusieurs proxy en direction de l'UA correspondant à l'URI du destinataire.

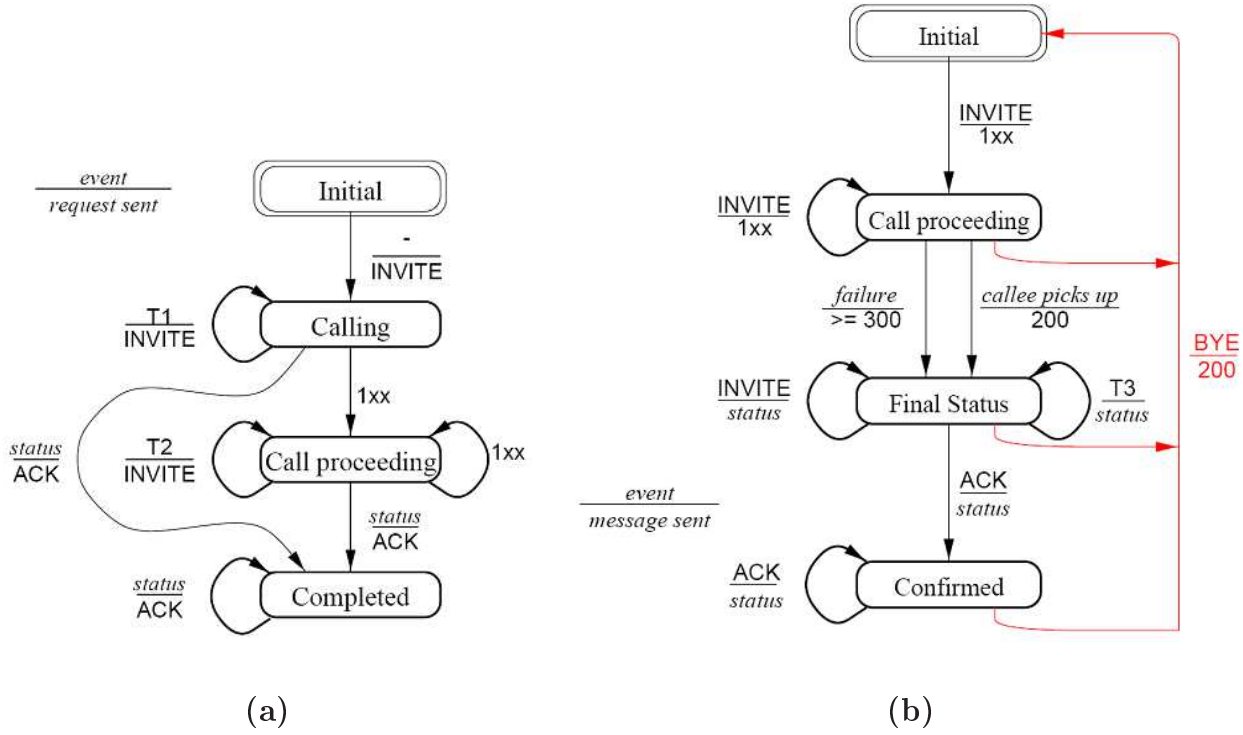


FIG. 1.17 – (a) Machine d'états SIP pour client. (b) Machine d'états SIP pour serveur

Exemples d'opération

Les spécifications de SIP sont tout à fait complexes [38] ; le document principal, RFC 3261, est de 269 pages. Pour donner certains sens à son opération, nous présentons un exemple. La figure 1.18 montre une tentative réussie par l'utilisateur Alice d'établir une session avec l'utilisateur Bob, dont l'URI est "bob@biloxi.com". L'UAC d'Alice est configuré pour communiquer avec un serveur proxy (le serveur *outbound* ou serveur du départ) dans son domaine, et commence en envoyant un message INVITE au serveur proxy qui indique son désir d'inviter l'UAS de Bob dans une session (1) ; le serveur reconnaît la demande (2). Bien que l'UAS de Bob est identifié par son URI, le serveur proxy outbound doit expliquer la possibilité que Bob n'est pas actuellement disponible ou que Bob s'est déplacé. En conséquence, le serveur proxy outbound doit renvoyer la requête INVITE au serveur proxy responsable du domaine "biloxi.com". Le proxy outbound consulte ainsi un serveur local DNS pour obtenir l'adresse IP du serveur proxy "biloxi.com" (3), en demandant du disque les ressources DNS

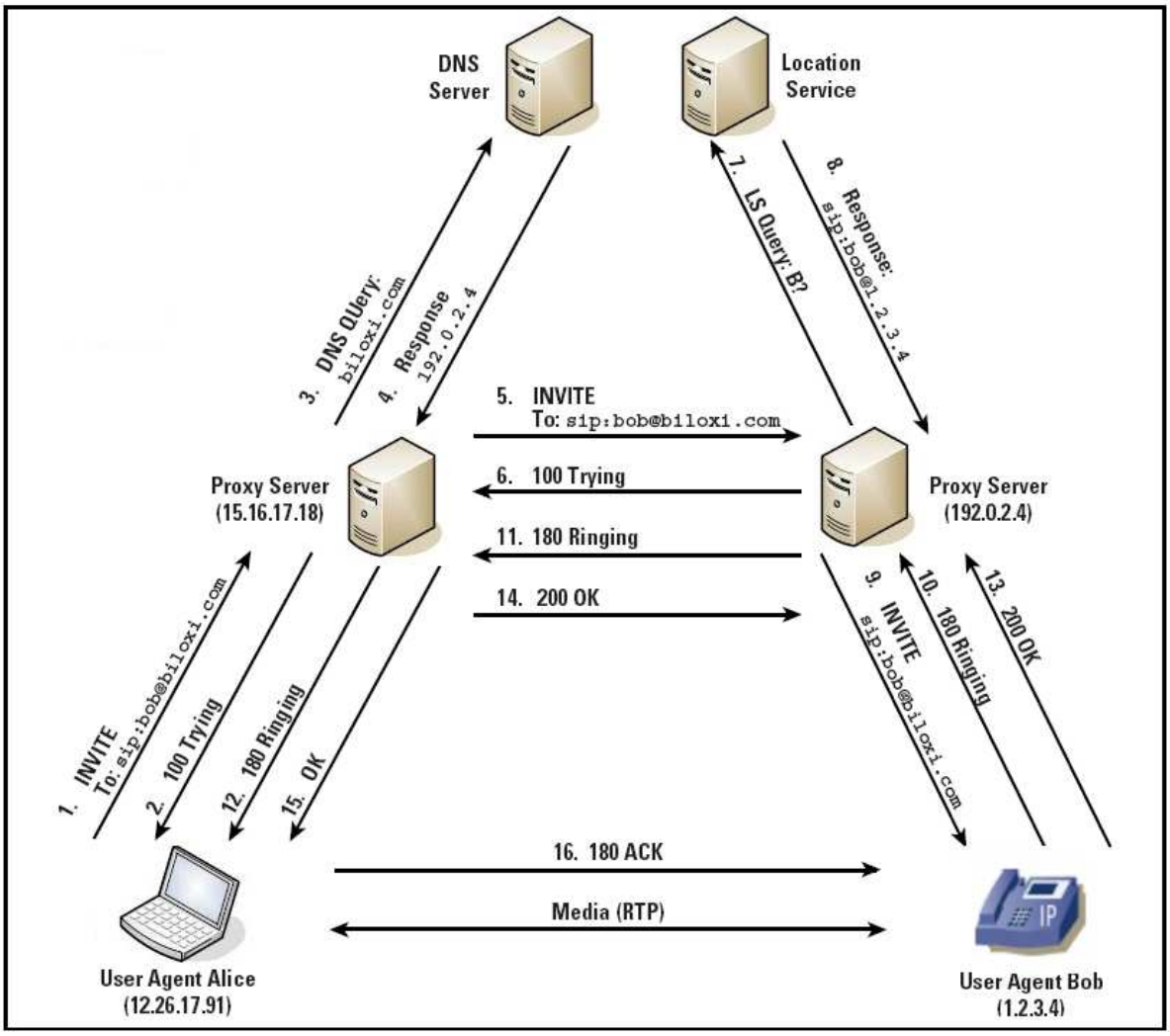


FIG. 1.18 – établissement réussi d'un appel SIP

SRV qui contient l'information sur le serveur proxy "biloxi.com".

Le serveur DNS répond (4) avec l'adresse IP du serveur proxy "biloxi.com" (le serveur *inbound* ou serveur d'arrivé). Le serveur proxy de Alice peut maintenant renvoyer le message INVITE au serveur proxy inbound (5), qui reconnaît le message (6). Le serveur proxy d'arrivée consulte maintenant un serveur de localisation pour déterminer la localisation de Bob (7), et le serveur de localisation répond avec la localisation de Bob, indiquant que Bob est enregistré, et donc disponible pour les messages SIP (8). Le serveur proxy peut maintenant transmettre le message INVITE à Bob (9). Une réponse sonnante est envoyée de Bob à Alice (10, 11, 12) tandis que l'UAS de Bob alerte l'application médias locale (par exemple, téléphonie). Quand l'application médias accepte l'appel, l'UAS de Bob renvoie une réponse OK à Alice (13, 14, 15).

A la fin, l'UAC de Alice envoie un message d'acquittement à l'UAS de Bob pour confirmer la réception de la réponse finale (16). Dans cet exemple, le message d'acquittement ACK est envoyé directement de Alice à Bob, en dépassant les deux proxy. Ceci se produit parce que les points finaux ont appris l'adresse de chacun de l'échange INVITE/200 (OK), qui n'a pas été connu quand l'initiale INVITE a été envoyée. La session médias a maintenant commencé, et Alice et Bob peuvent échanger des données au-dessus d'une ou plusieurs liaisons RTP.

1.4.2.5 Éléments critiques du routage SIP

Serveurs de VoIP/SIP critiques

Les serveurs de VoIP/SIP (proxy, registrar, redirect, etc.) sont des éléments dit "*critiques*" [18], car ils ont besoin d'être sécurisés et traités avec les mêmes précautions que tout autre serveur contenant de l'information sensible. Les systèmes de VoIP/SIP fournissent des dispositifs puissants de gestion, lesquels peuvent tagger les appels enregistrés de plusieurs manières afin d'aider pour les recherches futures. La mise en place des serveurs de VoIP/SIP est cruciale pour sécuriser l'environnement de traitement de la voix. Ces composants système doivent se trouver sur un segment réseau séparé et protégé par un firewall VoIP/SIP.

Problématique engendrée par SIP avec les pare-feu

Définition* 2.** *Un* ***pare-feu *(Firewall) est un dispositif informatique qui permet le passage sélectif des flux d'information entre deux réseaux, ainsi que la neutralisation des tentatives de pénétration en provenance du réseau public.*

La problématique engendrée par l'utilisation de SIP au travers des pare-feu vient du fait que ceux-ci sont généralement déployés en utilisant des politiques de filtrage qui empêchent tous les paquets qui ne proviennent pas ou qui ne sont pas destinés à une adresse IP et un port défini. Ces politiques, qui sont généralement statiques, ne permettent pas la traversée d'un flux de données à des protocoles comme SIP qui négocient de façon dynamique l'adresse IP et le numéro du port utilisé sur un poste [15].

Problématique engendrée par SIP avec les routeurs NAT

Définition* 3.** *Un routeur à translation d'adresse (routeur* ***NAT*) est un routeur qui est utilisé dans des réseaux où l'on veut limiter l'accès au réseau interne du réseau public ou partager une connexion Internet en utilisant une seule adresse IP publique.*

La problématique engendrée par l'utilisation de SIP au travers des routeurs NAT vient du fait que ceux-ci effectuent une translation d'adresse afin d'acheminer les paquets du réseau privé au réseau public. Lors de cette translation, l'adresse et le port utilisés changent selon le type de routeurs NAT. Étant donné que les routeurs NAT sont seulement en mesure de changer l'entête des paquets de type IP, les informations concernant l'adresse et le port d'origines contenues dans les requêtes SIP ne sont pas modifiés. L'utilisation du protocole SIP dans des réseaux utilisant des routeurs NAT pose une double problématique selon que le routeur est situé au niveau de l'appelant ou de l'appelé.

Protocoles de découverte des Firewalls et des NATs

STUN

Puisque les clients SIP souffrent de problèmes s'ils sont derrière un NAT, SIP peut être employé en commun avec STUN qui signifie "*Simple Traversal of UDP Through Network NATs*". STUN est défini par la RFC 3489 [30] et est un protocole client-serveur fait pour permettre à des paquets UDP de traverser un NAT. Les clients sur le réseau privé communiquent avec le serveur STUN du côté public afin de connaître leurs adresses et ports de réseau public, puis les paquets UDP sont envoyés avec des valeurs IP appropriées. Cependant STUN ne résout pas complètement tous problèmes qui peuvent se produire avec les NATs particulièrement si le proxy NAT est en mode symétrique.

TURN

TURN (*Traversal Using Relay NAT*) a été développé afin de pallier aux lacunes du protocole STUN. En effet, ce protocole simple a été conçu afin de permettre à des clients qui sont dans des réseaux utilisant des routeurs NAT ou encore des pare-feu d'effectuer des connexions (en utilisant TCP ou UDP) entre eux en passant par un serveur de relais, et ce, indépendamment de la topologie du réseau, des éléments de sécurité déployés et du protocole de communication utilisé. Son fonctionnement est basé sur l'utilisation d'un serveur TURN situé dans un réseau accessible par les postes désirant communiquer ensemble. Ce serveur servira de relais aux diverses communications entres les deux postes.

Interactive Connectivity Establishment(ICE)

Interactive Connectivity Establishment (ICE) est une technique en cours de développement [27] qui consiste à intégrer STUN et TURN au sein des clients SIP. Par cette intégration, cette technique permet de déterminer toutes les connections possibles entre deux postes. Pour ce faire, ICE caractérise premièrement les transmissions de média entre les postes et

le réseau public en effectuant des requêtes STUN à un serveur qui combine STUN/TURN. Deuxièmement, une liste d'adresses IP par lesquelles chaque poste peut être contacté est générée. Troisièmement, une caractérisation des transmissions de média entres deux postes est effectuée afin de déterminer la meilleure communication possible. Dans l'éventualité où aucune communication directe est possible entre deux postes, ICE utilisera les services d'un serveur TURN afin de permettre cette communication.

1.4.3 Les protocoles associés à SIP

A la différence de la téléphonie à commutation de circuit, les services de téléphonie Internet sont basés sur une gamme de protocoles à commutation de paquets, comme illustré dans figure 1.19. Le protocole SDP, défini dans la RFC 2327 [13], est Associé au protocole SIP. Ce dernier est employé pour inviter un ou plusieurs participants à une session, alors que le corps codé par SDP du message SIP contient les informations sur quels codage de médias (par exemple, la voix, la vidéo) les participants peuvent et doivent utilisé. Après que cette information soit échangée et reconnue, tous les utilisateurs reconnaissent les adresses IP des participants, la capacité disponible de transmission, et le type de média. Puis, la transmission de données commence, en utilisant un protocole de transport approprié. Typiquement, le protocole RTP est employé. Durant la session, les participants peuvent faire des changements aux paramètres de la session, tels que de nouveaux types de media ou de nouvelles parties à la session, en utilisant des messages SIP.

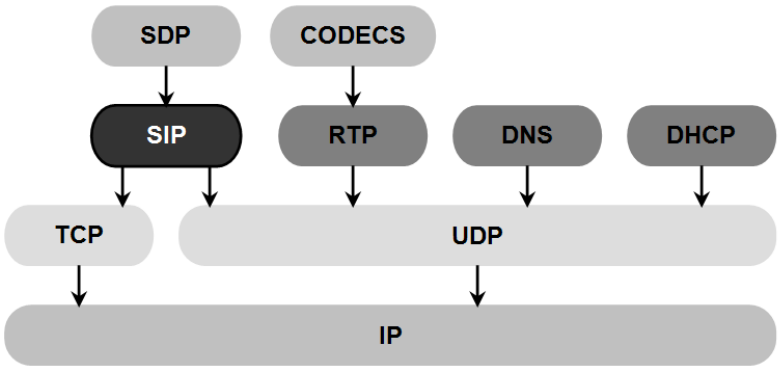


FIG. 1.19 – Architecture des protocoles associés à SIP

SDP (Session Description Protocol)

Le protocole de description de session (SDP) décrit le contenu des sessions, y compris la téléphonie, la radio Internet, et les applications multimédia. SDP inclut l'information suivante :

- **Flux médias :** Une session peut inclure des flux multiples de contenu différent. SDP définit actuellement l'audio, la vidéo, les données, le contrôle, et l'application comme des types de flux, similaire aux types de MIME utilisés pour la messagerie Internet.
- **Adresses :** SDP indique les adresses de destination, qui peuvent être une adresse multicast, pour le flux médias.
- **Ports :** Pour chaque flux, les numéros de port UDP pour l'envoi et la réception sont indiqués.
- **Types de charge utile :** Pour chaque type de flux médias en utilisation (par exemple, téléphonie), le type de charge utile indique les formats de médias qui peuvent être employés durant la session.
- **Temps de début et d'arrêt :** Ceux-ci s'appliquent aux sessions broadcast, par exemple, un programme de télévision ou de radio. Le début, l'arrêt, et les temps de répétition de la session sont indiqués.
- **Créateur :** Pour des sessions broadcast, le créateur est indiqué, avec l'information de contact. Ceci peut être utile si un récepteur rencontre des difficultés techniques.

Bien que le protocole SDP fournisse les possibilités pour décrire le contenu multimédia, il manque des mécanismes avec lesquels deux usagers conviennent sur les paramètres à employer. La RFC 3264 [28] remédie à ce manque en définissant un modèle simple d'offre/solution, par lequel deux usager échangent des messages de SDP pour se mettre d'accord sur la nature du contenu de multimédia à transmettre. Ce qui suit est un exemple de description de session SDP dans le corps SIP :

```
v=0
o=Bob 2890844526 2890842807 IN IP4 131.160.1.112
s=I want to know how you are doing
c=IN IP4 131.160.1.112
t=0 0
m=audio 49170 RTP/AVP 0
```


RTP (Real Time Transport Protocol)

Le protocole qui transporte la voix, associé au protocole de signalisation SIP dans la VoIP, est RTP. Ce protocole de transport et de contrôle adapté aux applications ayant des propriétés temps réel, a fait l'objet de la Recommandation RFC1889 de l'IETF et offre des moyens aux applications pour :

- reconstituer la base de temps des flux de données audio, vidéo et temps réel en général ;
- détecter rapidement les pertes de paquets et en informer la source dans des délais

Fiabilité	RTP n'est pas fiable s'il est utilisé avec UDP ou IP, qui ne sont eux-même pas fiables. RTP peut s'appuyer sur le service fiable fourni par les couches basses de réseaux en mode connecté (exemple : couches ATM)
Contrôle d'encombrement	RTP ne contient aucun mécanisme de contrôle d'encombrement intégré, comme TCP.
Stabilité des flux	RTP ne garantit pas la maîtrise des délais de transmission, ni la continuité du flux temps réel.
Ressources	RTP ne réserve pas de ressources et n'a aucun effet direct sur le comportement du réseaux.
Informations et outils pour le destinataire	L'en-tête RTP contient plusieurs informations pour la synchronisation et la restitution du signal par le récepteur : horodatage, indices de flux, de séquences, sources contributrices, etc.
Informations pour l'émetteur	Le RTP ne fournit aucune information utile à l'émetteur. Il est généralement utilisé avec le RTCP qui renvoie à l'émetteur un feed-back très complet sur la qualité de transmission : perte de paquets, délais, etc. Il permet à l'émetteur de moduler son débit de sortie en fonction des ressources disponibles.

TAB. 1.2 – Protocole RTP (Real Time Transport Protocol)

compatibles avec le service ;

- identifier le contenu des données pour en permettre la transmission sécurisée.

RTP fournit des fonctions de transport réseau de bout en bout (nous pouvons remarquer sur la figure 1.18 que le flux RTP ne transite pas par les serveurs SIP mais uniquement entre les deux agents en communications), et il ne réserve pas de ressources dans le réseau, puisque aucune action n'est faite sur les routeurs (le contrôle de la qualité de service n'est pas réalisé avec RTP) mais il est souvent complété avec profit par un protocole de réservation de ressources tel que RSVP *(Resource ReSerVation Protocol)*.

Le RTP n'apporte aucune fiabilité. Il n'offre que certaines caractéristiques d'un protocole de transport. Il n'assure pas la retransmission automatique des paquets manquants. Bien qu'il ne garantisse pas le délai de livraison, son apport aux échanges temps réel est important. Il fournit des informations très utiles au transport des contenus. Il assure l'horodatage des paquets en marquant l'heure de leur création ; leur remise dans l'ordre au destinataire s'en trouve simplifiée. Il comporte aussi des mécanismes de repérage et de synchronisation de flux différents ; chaque paquet est immédiatement reconnu comme appartenant à un flux bien précis. Le tableau 1.2 récapitule les principales caractéristiques du protocole RTP.

1.4.4 Comparaison des protocoles H.323 et SIP

Les principales caractéristiques des protocoles H323 et SIP sont comparées de manière synthétique à travers le tableau 1.3 [7] :

	H.323	**SIP**
Organisme initiateur (*Date*)	ITU-T (1996) *« Monde des télécoms »*	IETF (1999) *« Monde de l'IP »*
Protocoles de transport	TCP	TCP ou UDP
Caractéristiques	Lourde et robuste messages en ASN.1	Conçu pour être simple et leger, Messages textuels/UTF-8
evolutivité	protocole peu evolutif / Codecs prédéfinis, Structure rigide de la trame (H.221)	Protocole supportant les extentions (nouveaux en-têtes) et l'ajout de fonctions aux serveurs
Déploiement	Large (standard mature) Interoperable avec RTCP	Encore faible (en cours de normalisation)

TAB. 1.3 – Comparaison des protocoles SIP et H.323

1.5 Qualité de service et sécurité de la ToIP

1.5.1 Téléphonie sur IP et qualité de service

Une des principales difficultés soulevées par la téléphonie sur IP concerne la réalisation d'une qualité de service (QoS) similaire à celle à laquelle les usagers sont habitués sur les réseaux téléphoniques. Cette difficulté est liée, d'une part, aux aspects techniques propres au mode de transport des données sur les réseaux IP et, d'autre part, à ceux liés à l'organisation et au mode de fourniture de service sur les réseaux de données de manière générale et IP en particulier.

Le mode de transmission des informations par paquets utilisé par les réseaux IP introduit des facteurs de dégradation de la qualité de la communication. Nous pouvons répertorier quatre sources principales de difficultés liées au mode de transmission par paquets qui ont un impact sur le transport de la voix sur IP :

- **Perte :** La perte d'un paquet occasionne un manque d'informations lors de la réception du signal audio. En fonction du nombre de paquets perdus, la qualité sonore à

l'extrémité de réception peut être dégradée. Dans la philosophie IP, la perte de paquets fait partie intégrante du concept : les routeurs sont obligés de détruire des paquets afin d'éviter un éventuel encombrement. Le taux de perte de paquets dépendra de la qualité des lignes utilisées et du dimensionnement du réseau. Pour que la qualité de la parole soit acceptable, le taux de perte de paquets doit être inférieur à 20%.

- **Délai :** Le délai est le laps de temps exprimé en millisecondes qui s'écoule entre l'émission de la parole et sa reconstitution à l'extrémité de réception. S'il doit y avoir un échange interactif, les contraintes de délai doivent s'appliquer à la transmission de la parole. L'incidence de ce facteur sera modérée si le taux de perte est faible.

- **Gigue :** La gigue peut être décrite comme la variation du délai de transmission. Le protocole utilisé pour transporter les paquets vocaux sur l'Internet (un réseau IP) est le protocole UDP. La partie signalisation utilise la couche TCP. Le protocole UDP fonctionne en mode sans connexion, mode dans lequel les paquets ne prennent pas nécessairement la même route, d'où une variation du temps de transit. Une autre cause de variation du temps de transit peut être le nombre de routeurs rencontrés et de la charge supportée par chacun d'eux. Des buffers de compensation de gigue sont installés afin de reconstituer un flux synchrone à l'extrémité de réception. Cependant, ce processus augmente encore le délai de transmission. Pour conserver un niveau de qualité acceptable, la gigue doit rester en dessous de 100 msec.

- **Echo :** On peut décrire l'écho comme le laps de temps qui s'écoule entre l'émission d'un signal et la réception du même signal sous la forme d'un écho. Ce problème apparaît généralement dans le contexte de communications d'ordinateur à téléphone, de téléphone à ordinateur, ou de téléphone à téléphone. Il est causé par le renvoi d'une partie du signal traité par les composants électroniques des parties analogiques du système. Un écho de moins de 50 msec est imperceptible. Au-dessus de ce niveau, le locuteur entendra sa propre voix juste après avoir parlé. Pour résoudre le problème, des annuleurs d'écho haute performance sont installés au niveau des passerelles du réseau.

Les recommandations pour assurer la qualité de la voix sur IP :

Le paramètre d'échantillonnage ou codec (pour compression / décompression) est structurant en VoIP. Le codec détermine à quelle vitesse la voix est échantillonnée et dimensionne par la même le flux de données numériques que va générer la transformation d'un échantillon temporel de voix analogique. Les codecs sont répertoriés par leur nom à l'ITU. Le tableau1.4 récapitule les codecs les plus utilisés et leurs vitesses d'échantillonnage.

Codec	vitesses d'échantillonnage
G.711	64 kbps
G.726	32 kbps
G.726	24 kbps
G.728	16 kbps
G.729	8 kbps
G.723.1 MPMLQ	6.3 kbps
G.723.1 ACELP	5.3 kbps

TAB. 1.4 – Les codecs et leurs vitesses d'échantillonnage

Sauf à être contraint par un besoin impératif de gain de bande passante, on préférera le codec G.711 qui assure la numérisation la plus rapide de la voix ainsi que la plus grande clarté pour le récepteur[4].

En VoIP si l'on choisit un codec plus lent la valeur de délai de transit de 150 ms ne garantira pas la même QoS qu'en G.711. Les spécifications G.114 et G.131 de l'ITU-T fournissent la valeur recommandée pour le délai de transit de la voix de bout en bout, à 150 ms. Le tableau 1.5 présente les seuils de valeurs pour les paramètres critiques et les conséquences constatées pour le niveau de service de VoIP en codec G.711 64 kbps :

	Bon	Moyen	Mauvais
Délai de transit	$D < 150$ ms	150 ms $< D < 400$ ms	400 ms $< D$
Gigue de phase	$G < 20$ ms	20 ms $< G < 50$ ms	50 ms $< G$
Perte de données	$P < 1\%$	$1\% < P < 3\%$	$3\% < P$

TAB. 1.5 – Niveau de service de VoIP en codec G.711 64 kbps

1.5.2 Téléphonie sur IP et sécurité

Les risques de la VoIP sont comparables a ceux que l'on trouve dans les réseaux IP traditionnels [3], car les deux type de réseaux sont connectés à Internet, et de ce fait exposé à toutes les tentatives d'intrusion, et d'attaques que l'on y trouve. Certaines attaques sont plus faciles en VoIP, d'autres au contraire plus complexes.

Le problème de la sécurité de la VoIP est donc double. Le fait que la voix et les données partagent le même réseau implique qu'aux vulnérabilités de sécurité des réseaux de données s'ajouteront les nouvelles vulnérabilités propres à la VoIP.

[4] www.lucent.com.

1.5.2.1 Menaces héritant directement du modèle réseau de donnée IP

La téléphonie sur IP avoue volontiers parler de "Phreaker" afin de désigner toute personne mal intentionnée qui s'attaque à des réseaux téléphoniques. Voici donc une liste non exhaustive des vulnérabilités propres aux réseaux de données IP :

- **DoS (*Denis de service*) :** Privation d'accès à un service réseau en bombardant les serveurs (proxy, gateway, ...) avec des paquets malveillants.
- **Message integrity (*Intégrité de message*) :** Vérification que le message reçu n'a pas été altéré durant la transmission.
- **Paquets sniffing (*Capture de paquets*) :** Obtention d'informations (adresse IP/MAC, protocoles utilisés).
- **password attack (*Mot de passe*) :** Casser des mots de passe afin d'obtenir des privilèges.
- **MitM : Man-in-the-Middle (*Personne au milieu*) :** Paquets modifiés de manière à usurper une identité ou permettant la récupération d'information de transmission sur des utilisateurs.
- **Malware : virus, vers (worms), trojans :** MALicious softWARE : Applications malicieuse faisant référence à des programmes crapuleux.
- **Hijacking (*Détournement*) :** Attaque dans laquelle l'attaquant prend procession d'une connexion en cours entre deux entités en se faisant passer pour l'une des deux entités.
- **Misuse (*Mauvaise utilisation*) :** Modifier le but inhérent d'une fonction ou autre afin de pouvoir abuser du système.
- **Coupure de courant, Autres...**

1.5.2.2 Menaces propres à l'utilisation de la VoIP/SIP

La particularité de la VoIP face aux données IP standard est principalement associée à la notion de qualité de service. En effet, comme dans tout système de téléphonie, la VoIP apporte une très grande importance à la QoS. Ceci augmente notablement l'exposition aux attaques de types DoS. Cela affecte principalement :

- Délai/latence
- Perte de paquet
- Variation du délai de transfert de l'information (jiter)
- Bande passante
- Techniques de codage de la parole

La QoS doit donc faire face à ces nouvelles vulnérabilités propres au service de VoIP, les éléments SIP vulnérables sont :

1. La signalisation (SIP/SDP) : En s'attaquant à la signalisation, le pirate peut obtenir des informations sur les utilisateurs et se faire passer pour quelqu'un d'autre par exemple. Cela permet également de détourner des appels et de manipuler la taxation.
2. Les données (SIP-DATA) : En s'attaquant aux données le pirate peut écouter les communications, modifier/insérer et supprimer de l'information.

Voici quelques vulnérabilités propres au déploiement d'une plateforme de VoIP avec le protocole de signalisation SIP :

DoS-CANCEL : La première des attaques qui est à la fois simple et efficace à mettre en place est de type DoS. Le but est ici de faire croire à un UAC appelé que son partenaire désire annuler son appel. L'annulation d'un appel auquel le partenaire n'a pas encore répondu se traduit par l'envoi d'une réponse CANCEL en SIP. Il faut donc que l'on fasse passer pour l'appelant en injectant un message CANCEL destiné à l'appelé tout en respectant les champs de l'entête SIP. Il est également possible de faire l'inverse. C'est-à-dire que l'on peut faire croire à l'appelant que l'appelé ne désire pas répondre à l'appel.

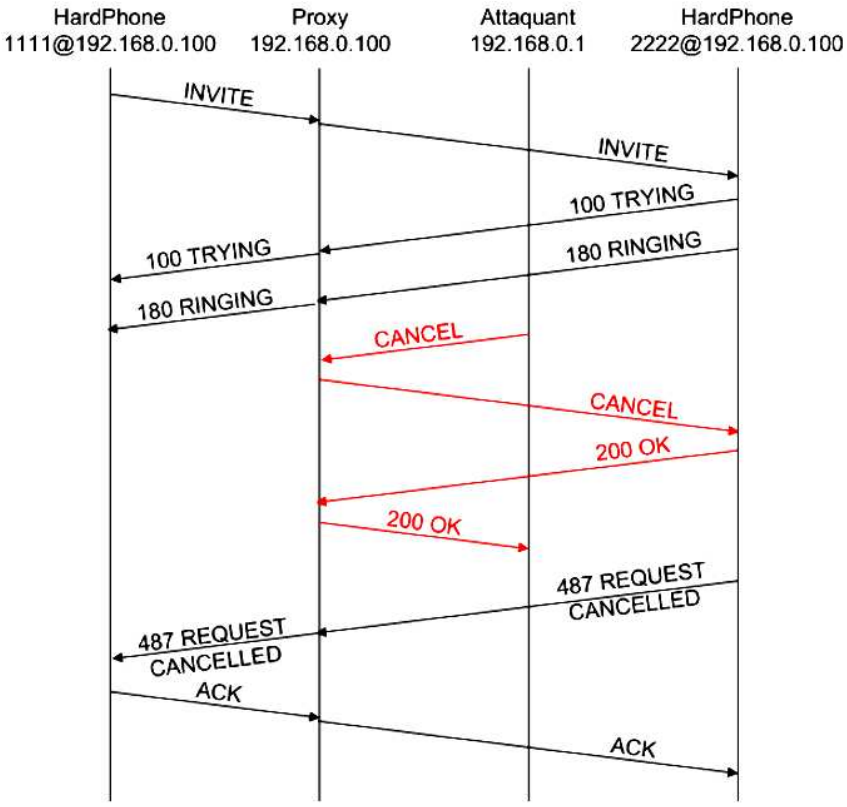


Fig. 1.20 – Diagramme explicatif d'un DoS-CANCEL

Pour cela, il suffit d'initier une communication depuis un des téléphones mais sans décrocher à l'autre bout. C'est à ce moment là qu'il faut injecter le message CANCEL (voir figure 1.20), et l'envoyer au partenaire qui n'a pas encore décroché. Ceci étant fait, le partenaire appelé comprend que l'appelant désirait raccrocher alors que ce n'était pas le cas. De son côté, l'appelant continue d'attendre que son partenaire décroche (mais en vain, en fait jusqu'à la fin du "time out INVITE" du proxy). C'est donc un DoS sur l'appelé et l'appelant.

Call hijacking (détournement d'appel) : Il est très simple de détourner un appel SIP. Cela peut se faire au moyen d'un message 301 moved permanently ou 302 moved temporarily. En effet, ces messages sont utilisés lorsque le call forwarding est activé de manière permanente ou temporaire. Il suffit donc de faire croire à un UAC qui tente d'établir un appel que son partenaire à activé le call forwading tout en spécifiant dans le message SIP qu'il faut contacter l'attaquant. La figure 1.21 illustre un exemple de diagramme en flèche des communications avec l'attaque de détournement d'appel :

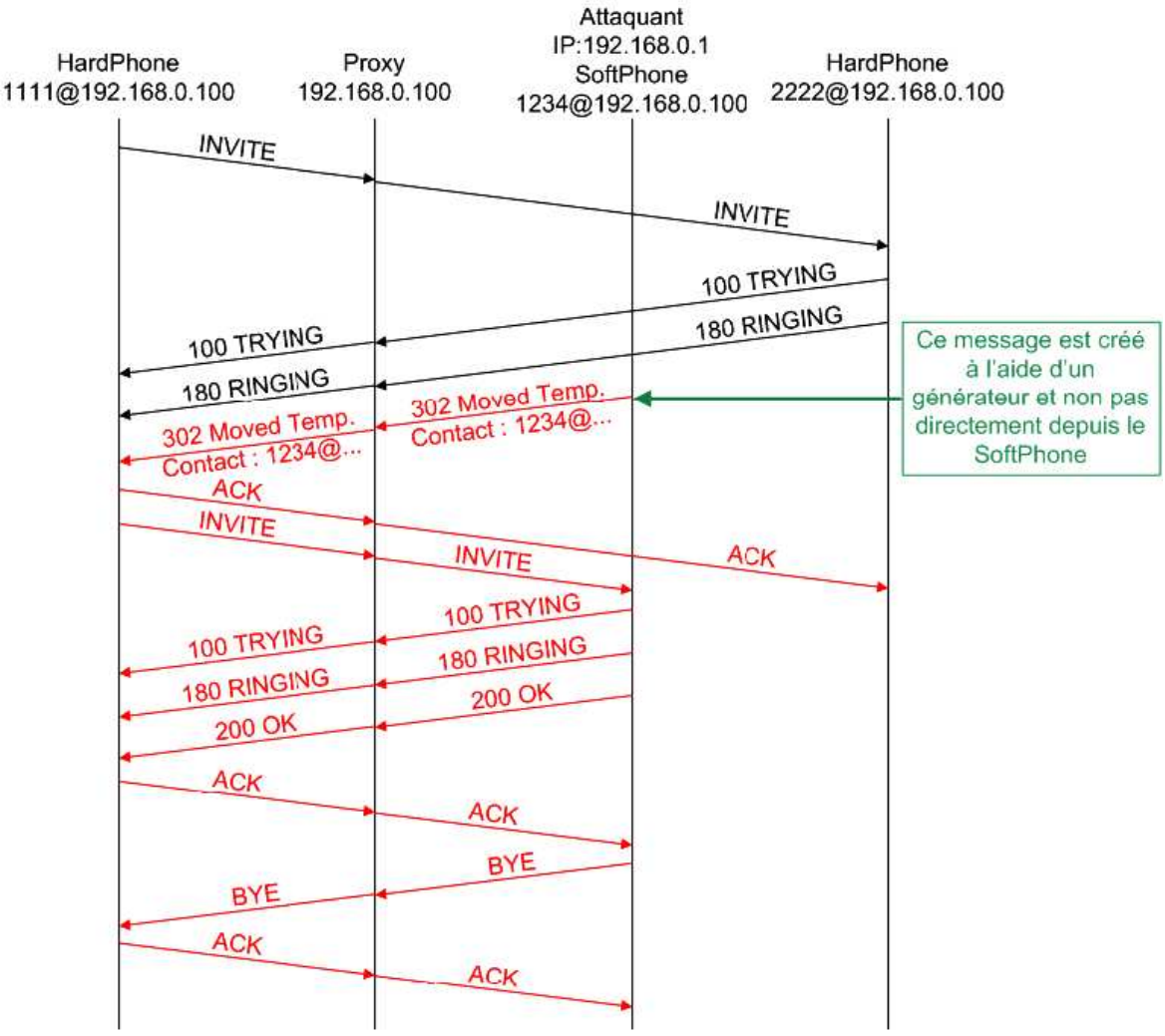


FIG. 1.21 – Diagramme de communication avec l'attaque "call hijacking"

SPAM-INVITE : Il est très facile de faire sonner plusieurs téléphones en même temps avec SIP. Il s'agit ici pour l'attaquant de connaître les URI de tous les téléphones concernés ainsi que l'adresse du serveur proxy/registrar. Ensuite il suffit de créer un message par URI et de les envoyer en même temps. Tous les téléphones sonneront donc en même temps. Cette attaque ne nécessite pas forcément de pouvoir sniffer les messages SIP si l'attaquant connaît déjà les URI et l'IP du/des serveurs. Cette attaque est représentée par la figure 1.22 :

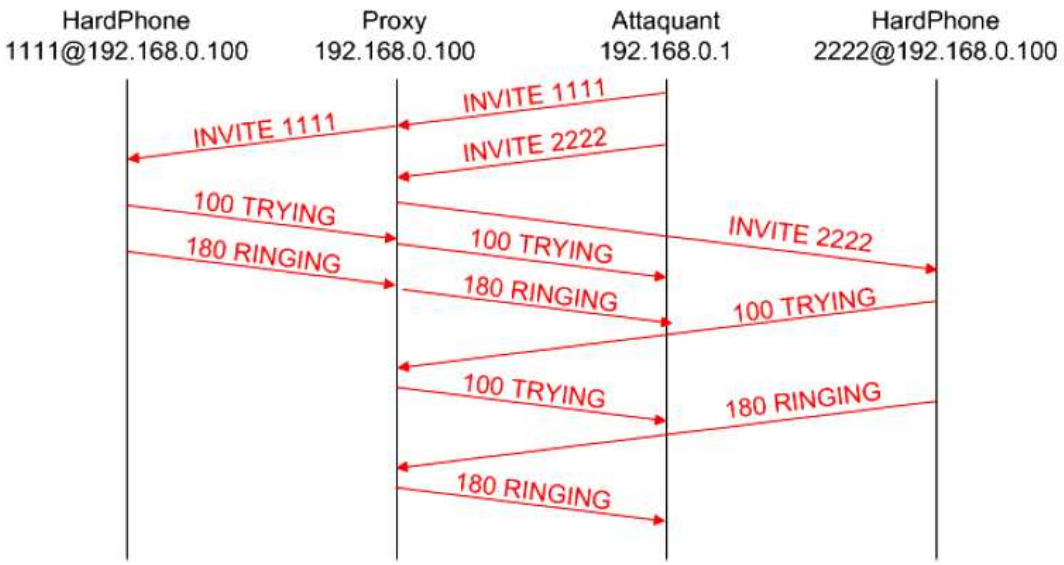


FIG. 1.22 – Diagramme de communication avec le "SPAM-INVITE"

Il existe quatres types de SPAM différent avec la VoIP/SIP :

1. CALL SPAM : Ce type de SPAM est définit comme une masse non-sollicitée de tentatives d'initiation de session (avec des messages INVITE), afin de tenter d'établir des sessions de communication audio. Si l'utilisateur répond, le spammer s'affaire à relayer ses messages sur le média temps réel. Ceci est le spam classique de telemarketer, appliqué à SIP.

2. IM SPAM ou SPIM : Ce type de spam est similaire au spam email. Il est définit par une masse non-sollicitée de messages instantanés, dont le contenu comprend le message que le spammer cherche à convoyer. Le spam IM est le plus souvent envoyé en utilisant les requêtes SIP MESSAGE. Toutefois, une quelconque autre requête qui provoquerait un affichage automatique du contenu sur l'écran de l'utilisateur devrait également suffire. Il est possible d'inclure des requêtes INVITE avec des grandes entêtes de sujet ou des requêtes INVITE avec du texte ou un corps HTML.

3. PRESENCE SPAM : Ce type de spam est similaire au spam IM. Il est définit par une masse non-sollicitée de requêtes de présence (c'est-à-dire des requêtes SUSCRIBE dans le but d'être sur la "white list" d'un utilisateur afin de pouvoir lui envoyer des IM ou pour initier d'autres formes de communications). A la différence du spam IM, le spam de présence ne doit pas réellement convoyer le contenu dans le message. Il n'est donc pas évident de trouver l'utilité ou la valeur d'un tel type de spam.

4. SPIT (SPam over Internet Telephony) : Le SPIT correspond à des messages téléphoniques non sollicités qui peuvent être envoyés à un utilisateur comme à l'ensemble de l'entreprise. En plus du dérangement de l'utilisateur, ce type de message peut provoquer une surcharge du système tant au niveau du réseau que sur les différents serveurs. Les auteurs de ce type d'attaques sont souvent difficiles à tracer sur Internet. Ils peuvent donc envoyer des messages qui ne sont pas seulement publicitaires mais qui peuvent aussi être exploités pour organiser des actions frauduleuses, pour utiliser des ressources non autorisées ou encore pour récupérer des informations confidentielles.

1.5.2.3 Solutions pour sécuriser la VoIP

Chiffrement avec IPSec

Le chiffrement des flux avec IPSec[5] permet de garantir la confidentialité et l'intégrité des flux échangés. IPSec est un mécanisme général pouvant être utilisé pour protéger les messages SIP au niveau IP. Avec SIP, chaque proxy sur le chemin doit avoir accès en lecture/écriture sur l'entête des messages SIP afin de pouvoir ajouter/retirer des entêtes VIA. Afin de permettre l'utilisation d'IPSec ESP (Encapsulating Security Payload) ou AH (Authentification Header), son fonctionnement doit être basé sur un mode Hop-By-Hop.

IPSec est basé sur un assortiment de mécanismes protégeant les données échangées sur le réseau. Il fonctionne à la couche IP et traite tous les paquets IP. Ainsi, il protège toutes les applications et peut être implémenté dans tous les appareils utilisant le réseau de manière point à point voir lien à lien. Son but est donc d'éviter l'espionnage du flux de données et l'accès illicite aux ressources. Il permet de garantir la confidentialité et l'authenticité des données échangées. Il fournit également une protection contre les replays-attacks. Il est bon de remarquer qu'il permet un haut niveau de protection s'il est utilisé avec des algorithmes forts et dans un environnement sécurisé. Ces fonctionnalités sont fournies par des mécanismes cryptographiques :

[5] http ://www.hsc.fr/ressources/presentations/websec2000/index

- Message Authentication Code (MAC) = Authenticité des données
- Chiffrement des données = Confidentialité des données
- Numéro de séquence = protection contre les replays-attacks

Ces mécanismes sont implémentés à l'aide de deux extensions du protocole IP :

- AH (Authentification Header) qui permet d'assurer l'authenticité des datagrammes IP,
- ESP (Encapsulating Security Payload) qui assure la confidentialité des données et/ou leur authenticité.

AH et ESP peuvent fonctionner avec plusieurs algorithmes cryptographiques, toutefois l'IETF préconise l'utilisation de triple DES (128 bits) pour le chiffrement et HMAC-MD5 ou HMAC-SHA1 pour l'authenticité.

La sécurité avec SRTP

SRTP (Secure Real-time Transport Protocol) est un profil et une amélioration du standard RTP pour assurer la confidentialité, l'intégrité et l'authentification des messages et la protection contre le rejeu. SRTP est un nouveau mécanisme de sécurité considéré pour sécuriser les réseaux de Voix sur IP.

SRTP crypte les données utiles d'un paquet VoIP (payload d'un paquet RTP) mais garde l'en-tête en clair. Il ne crypte pas les paquets de signalisation de la voix. Le but de SRTP est d'assurer la confidentialité des champs utiles des paquets RTP et RTCP, l'intégrité de tout le paquet RTP et RTCP avec protection contre le rejeu. Ces services de sécurité sont optionnels et mutuellement indépendants. Seule la protection de l'intégrité des paquets RTCP est obligatoire pour éviter la perturbation du flux RTP.

SRTP est indépendant des couches sous-jacentes utilisées par RTP. SRTP est caractérisé par un débit élevé et une faible expansion des paquets. SRTP utilise le additive stream cypher comme outil de cryptage, une fonction de hachage universelle pour l'authentification des messages et un numéro de séquence implicite pour le séquencement basé sur le numéro de séquence des en-têtes du paquet RTP.

Les réseaux locaux virtuels (VLAN)

La séparation logique des flux voix et data à l'aide de VLAN est une mesure fortement recommandée. Elle doit permettre d'éviter que les incidents rencontrés sur l'un des flux ne puissent perturber l'autre. Les VLAN ou réseaux locaux virtuels, peuvent être représentés comme une séparation logique d'un même réseau physique. Cette opération se fait au niveau

2 du modèle OSI. On notera cependant qu'un VLAN est souvent configuré pour correspondre directement à un subnet IP bien identifié, préparant ainsi le travail à effectuer sur la couche supérieure.

Dans un environnement commuté complet, les VLAN vont créer une séparation logique des domaines de broadcast et de collisions - des problèmes dus à ces deux éléments sont souvent rencontrés dans des LAN trop importants ou lorsqu'on utilise encore des hubs. De plus, la réduction des domaines de broadcast permet de réduire le trafic sur le réseau, libérant ainsi plus de bande passante pour les applications utiles et réduisant les temps de traitement sur les périphériques réseau.

On peut utiliser cette technique pour organiser les postes utilisateurs selon leurs situations physiques dans les bâtiments, le service auquel appartient l'utilisateur, la vitesse de connexion, ou tout autre critère ayant du sens dans l'entreprise. Un renforcement de la sécurité peut être réalisé en mettant en place un filtrage inter-VLAN, n'autorisant que les utilisateurs d'un VLAN à y accéder. Le risque de DoS peut ainsi être réduit.

Ce mécanisme est vraiment très intéressant. D'une part, son utilisation est "gratuite" à partir du moment ou le switch est déjà présent sur le réseau et qu'il permet d'utiliser des VLANs. D'autre part, les domaines de collisions sont séparés et ainsi le réseau fonctionne plus efficacement dans chaque VLAN. Mais surtout, ce mécanisme permet d'éviter toutes les attaques de VoIP/SIP visant directement un élément de VoIP (DoS par injection de messages, Buffer Overflow, SPAM, détournement d'appel, etc.) et venant de personnes qui ne font pas partie du VLAN de VoIP. Toutes les attaques de VoIP qui s'effectuent de manière indirecte (c'est-à-dire en passant par un élément réseau qui n'est pas forcément un éléments de VoIP), telles que l'épuisement de ressource DHCP et les attaques TFTP par exemple, ne peuvent pas être évitées sans devoir isoler ces éléments dans le VLAN de VoIP uniquement. Mais attention, cela n'est pas toujours possible puisque ces éléments sont souvent utilisés par des éléments du réseau de donnée (en particulier DHCP).

1.6 Conclusion

Au final, on remarque donc l'importance des serveurs SIP et en particulier des proxies, qui jouent un rôle de routage au niveau du protocole SIP. Les proxies relaient les messages alors que les serveurs de redirection donnent les informations nécessaires à la redirection. Le routage des requêtes SIP s'effectue donc par sauts de proche en proche entre proxies. Ce mécanisme n'est bien sûr possible que grâce au service de registration qui notifie les nœuds

du réseau de la (ou les) localisation(s) à appeler le destinataire.

La voix sur IP doit faire face à deux grands problèmes d'actualités : la qualité de service et la sécurité. Ce sont malheureusement deux objectifs n'allant pas dans le même sens. La sécurité de la technique VoIP doit, pour résumer, consister en la protection des protocoles VoIP, la protection des systèmes d'exploitation sur lesquels les solutions VoIP sont basées (buffer overflows et vulnérabilités diverses...), la protection des couches réseaux contre les attaques de type DoS, la protection de la bande passante dédiée à la voix sur le réseau. Toute sécurité impose des contraintes nécessaires, certes ; mais la question est de savoir, à quel point peut on sécuriser sans dégrader la qualité de la voix sur le réseau.

Historiquement, tous les services offerts sur Internet étaient développés selon le modèle de réseau client-serveur alors que les nouveaux services sont développés selon le modèle de réseau Pair à Pair. Ce changement d'architecture engendre certains problématique lors de l'adaptation des applications existantes. Parmi ces problématique, le problèmes de sécurité dans les réseaux pair à pair, vue que ces derniers manque de contrôle centralisé.

Chapitre 2

Présentation de l'architecture Pair-à-Pair

2.1 Généralités

Les systèmes informatiques peuvent être classifiés en deux grandes catégories, présentés par la figure 2.1 : les systèmes centralisés (par exemple de type *MainFrames*) et les systèmes distribués. Ces derniers peuvent être construits selon deux modèles : le modèle client-serveur (*plat* ou *hiérarchique*) et le modèle pair à pair qui peut être *pur*, *hybride* ou *centralisé* [19].

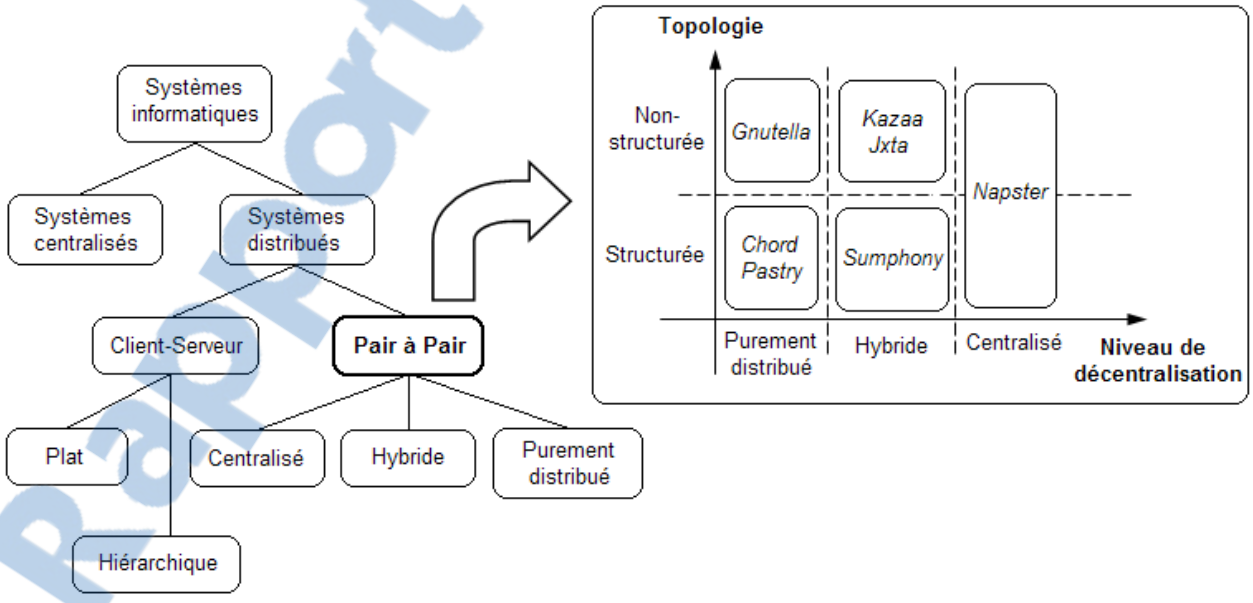


Fig. 2.1 – Classification des systèmes informatiques et des applications P2P

Les systèmes P2P[1] sont apparus avec l'émergence des échanges de documents multimédia

[1]Peer-to-peer signifie littéralement pair à pair. Ce concept introduit ainsi une relation d'égal à égal entre deux ordinateurs.

sur Internet. Gnutella, Freenet ou encore Napster constituent des exemples de systèmes pair à pair d'échange de fichiers. L'idée qui sous-tend les systèmes pair à pair est que chaque machine individuelle du système peut à la fois remplir le rôle de client et de serveur en même temps. Il n'y a alors plus de machines serveurs dédiées à l'application.

2.1.1 Définition

De nombreuses définitions existent concernant le concept de pair-à-pair (parfois, elles sont même contradictoires sur certains points). Certaines le définissent juste comme une collection de ressources hétérogènes et distribuées connectées par un réseau [32] ou encore comme l'opposé du modèle client-serveur [36]. D'autres définitions présentent le pair-à-pair comme un système permettant de faire communiquer entre eux les différents nœuds d'un réseau sans passer par un intermédiaire[2] ou encore comme une infrastructure dans laquelle les nœuds peuvent tous avoir potentiellement les mêmes fonctions. Les définitions que nous proposons se concentrent sur la nature décentralisée du modèle ainsi que sur les concepts de partage de ressources et de dynamisme :

***Définition* 4.** *Le terme pair à pair désigne une classe de systèmes et d'applications qui utilisent des ressources distribuées, où les entités appelées pairs jouent le double rôle de client et serveur et interagissent afin d'offrir à une communauté un service de manière décentralisée.*

***Définition* 5.** *Un système distribué est qualifié de pair-à-pair si les nœuds de ce système partagent une partie de leurs ressources matérielles (puissance de calcul, espace disque, interface réseau, imprimante, etc.). Ces ressources partagées sont nécessaires pour fournir les services et les contenus offerts par le système (par exemple le partage de fichiers ou d'espaces de travail collaboratifs). Ces ressources sont accessibles directement par les autres pairs sans passer par des entités intermédiaires. Les participants de ce genre de système sont à la fois des fournisseurs de ressources (services et contenus) et des utilisateurs de ces ressources. C'est le concept de ServEnt (contraction des termes « Serveur » et « Client »).*

***Définition* 6.** *Peer-to-peer systems are distributed systems consisting of interconnected nodes able to selforganize into network topologies with the purpose of sharing resources such as content, CPU cycles, storage and bandwidth, capable of adapting to failures and accommodating transient populations of nodes while maintaining acceptable connectivity and performance, without requiring the intermediation or support of a global centralized server or authority. [1]*

[2]Peer-to-Peer Working Group. p2pwg, 2003. http ://www.p2p.internet2.edu/.

2.1.2 Caractéristiques des systèmes P2P

Les réseaux pair à pair disposent d'un nombre de propriétés qui les rendent plus intéressants que les autres systèmes distribués. Une caractéristique de ces réseaux est que la qualité et la quantité des données disponibles augmentent à mesure que le nombre d'utilisateurs augmente. La valeur du réseau augmente donc avec sa popularité. Cette section décrit les caractéristiques d'un système pair-à-pair, à savoir, leur aspect décentralisé, l'extensibilité qu'ils doivent assurer (avec l'hétérogénéité des machines qui en résulte), leur capacité d'auto-organisation ainsi que l'anonymat et la sécurité qu'ils doivent conférer aux utilisateurs.

2.1.2.1 Décentralisation :

Dans la notion de pair-à-pair, les nœuds peuvent être à la fois des *serveurs* ou des *clients*, d'où l'apparition du terme de *ServEnt*. De cette manière, tous les pairs du systèmes doivent pouvoir communiquer directement entre eux. Cette *décentralisation* présente l'avantage de réduire voire même de supprimer complètement tout risque de *goulots d'étranglement*, très fréquents dans les systèmes mettant en œuvre le modèle client-serveur. Cependant une architecture complètement décentralisée peut être difficile à mettre en place puisqu'il n'existe pas d'entité possédant une vue globale de tous les pairs du système ni du type de ressource qu'ils partagent. De plus, cette décentralisation peut parfois même présenter certaines limitations, notamment en terme de performance lors de la recherche de ressources. C'est la raison pour laquelle des centralisations sont parfois envisageables dans les environnements pair-à-pair. Il faut cependant veiller à ce qu'elles ne limitent pas non plus l'extensibilité du système.

2.1.2.2 L'extensibilité et l'hétérogénéité :

Une conséquence directe de la décentralisation inhérente aux systèmes pair-à-pair est l'augmentation du nombre de ressources qu'il est ainsi possible de fédérer. Le nombre de nœuds composant le réseau n'est donc plus limité par une centralisation forte comme dans le concept de client-serveur. Il faut donc pouvoir assurer de bonnes performances au système, notamment lors des recherches de ressources. On appelle cette caractéristique l'*extensibilité* ou encore le *passage à l'échelle* mais on utilise plus souvent le terme anglais de *scalability*.

Le fait de fédérer un nombre si important de ressources entraîne également une grande *hétérogénéité* des machines, tant au niveau matériel (puissance et type de processeur, débit des connexions, etc) que logiciel (systèmes d'exploitations, etc). Il faut donc que le système soit

portable afin de pouvoir s'exécuter sur toutes les machines de l'infrastructure sans présenter de problèmes de compatibilité entre les pairs.

2.1.2.3 Connectivité Ad-hoc et Auto-organisation :

Dans les réseaux pair-à-pair, l'environnement des ressources est complètement *dynamique* puisque les pairs peuvent apparaître et disparaître à n'importe quel moment (on parle aussi de nœuds «*volatiles*»). Dans les systèmes mettant en oeuvre le modèle client-serveur, la connexion ou la déconnexion d'un nœud ne concerne que le serveur, qui devra mettre à jour la liste des clients connectés. Dans les systèmes pair-à-pair, de par les connexions établies entre certains nœuds, l'apparition ou la disparition d'une ressource doivent être détectées et prises en compte. Il faut donc que le système définisse une stratégie pour que les nœuds de l'infrastructure puissent *s'auto-organiser* lors de la connexion (ou la déconnexion) d'un pair.

2.1.2.4 L'anonymat et la sécurité des utilisateurs :

Dans les systèmes pair-à-pair, il n'est pas souhaitable d'identifier les utilisateurs (par un mécanisme de *logins* par exemple) puisqu'il y en a en général autant que le nombre très important de ressources. D'autre part, l'utilisateur du système peut ne pas vouloir que des informations le concernant soient disponibles dans le système. De plus, de nombreux fournisseurs d'accès allouent dynamiquement les adresses IP aux machines de leurs clients, ce qui peut rendre difficile leur localisation depuis le système. Ces deux constats mènent donc les développeurs de réseaux pair-à-pair à les concevoir de telle sorte que les utilisateurs soient *anonymes* et que les mécanismes de localisation des ressources reposent sur d'autres principes que celui des adresses IP.

Un autre problème concerne la *sécurité des participants*. En effet, l'exécution de l'application sur la machine de l'utilisateur et les connexions entrantes des requêtes des autres utilisateurs ne doivent en aucun cas pouvoir conduire à une corruption de la ressource ou de ses données. Le système doit donc assurer automatiquement la sécurité des participants. Enfin, les *pare-feux* de certaines machines bloquent les communications entrantes rendant ainsi impossible les connexions directes entre les pairs. Une situation encore différente mais tout aussi problématique concerne les machines d'un réseau utilisant les technologies *NAT (Network Address Translation)*. Dans ce cas, les machines situées dans ce réseau local ne sont même pas visibles de l'extérieur. Une infrastructure pair-à-pair doit donc proposer des solutions permettant de résoudre ces problèmes de sécurité.

2.1.3 Niveau de décentralisation

Actuellement, les applications P2P sont construites selon un degré de décentralisation variable qui représente une distribution plus ou moins forte des tâches accomplies. Les différentes propositions de classification des infrastructures P2P en fonction de leur degré de décentralisation scindent le modèle P2P en deux ou trois sous modèles. Nous considérons aussi que le modèle P2P peut se scinder en trois sous-modèles qui sont les modèles pur, hybride et centralisé.

2.1.3.1 Le modèle P2P pur

***Définition* 7.** *Le modèle P2P pur représente un modèle P2P tel qu'il est spécifié dans la définition 5 et dans lequel tous les pairs sont strictement équivalents.*

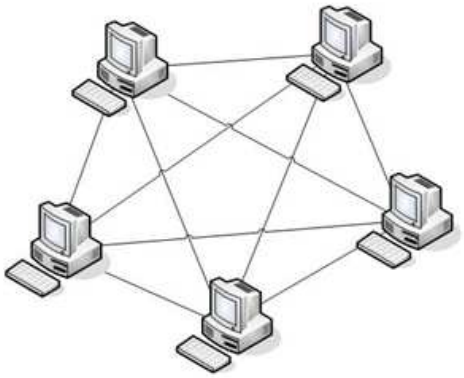


FIG. 2.2 – Topologie P2P construite selon le modèle pur.

Dans le modèle pur (figure 2.2), il n'existe pas de serveur centralisé. Actuellement, de nombreuses applications P2P sont construites selon ce modèle. On trouve par exemple Gnutella[3] , tel qu'il a était déployé dans sa première version, et toutes les infrastructures à base de table de hachage distribuée telles que Chord, Pastry, Tapestry, CAN,...

2.1.3.2 Le modèle P2P hybride

***Définition* 8.** *Le modèle P2P hybride représente un modèle P2P tel qu'il est spécifié dans la définition 5 et dans lequel certains pairs jouent un rôle particulier. Ces pairs exécutent des*

[3] "Gnutella : peer-to-peer file sharing software application." http ://www.gnutella.com.

fonctions différentes des autres pairs rendant les pairs non équivalents et apportant ainsi un certain degré de centralisation.

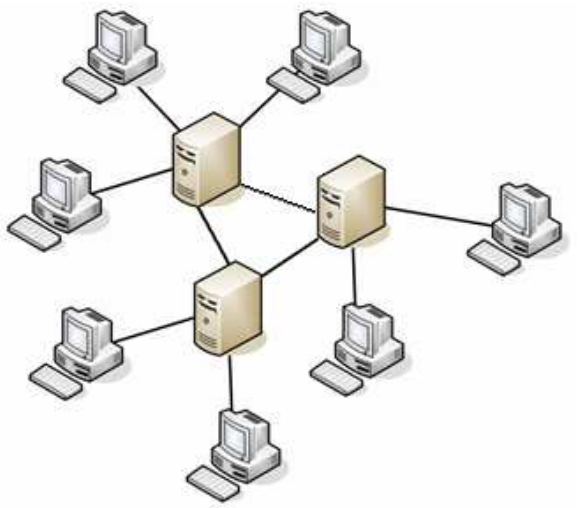


FIG. 2.3 – Topologies P2P construite selon le modèle hybride.

Le modèle hybride (figure 2.3) ajoute un degré de hiérarchie au modèle pur. Les pairs particuliers utilisés dans le modèle hybride sont généralement appelés des *super-pairs*. Le rôle qu'ils jouent varie d'une infrastructure à une autre. En général, ils sont utilisés pour assurer des fonctions relatives au routage ou à l'organisation fonctionnelle des pairs.

2.1.3.3 Le modèle P2P centralisé

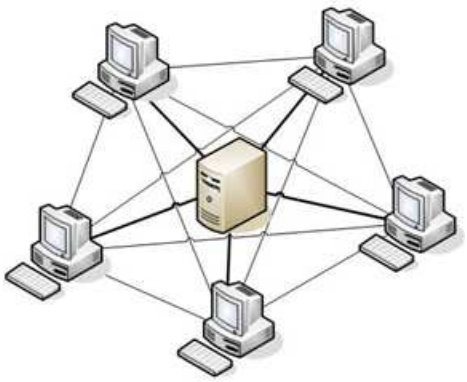


FIG. 2.4 – Topologies P2P construite selon le modèle centralisé.

***Définition* 9.** *Le modèle P2P centralisé représente un modèle P2P tel qu'il est spécifié dans la définition 5 et dans lequel un serveur dédié est utilisé pour assurer les fonctions de découverte et localisation de ressources.*

Le modèle centralisé est à la limite du modèle P2P car il repose sur un serveur dédié qui centralise et maintient l'ensemble des connaissances de la communauté, les ressources étant toujours hébergées sur les pairs. Ce modèle est utilisé dans des applications telles que Napster ou Skype. Un exemple de topologie P2P centralisée est représentée sur la figure 2.4, qui montre que chaque pair ne possède au minimum qu'une connaissance du serveur central et pas des autres pairs, bien qu'une fois les opérations de découverte et localisation effectuées, il puisse interagir directement avec eux. Ainsi, cette interaction possible entre les pairs différencie le modèle P2P centralisé du modèle client-serveur (voir le tableau 2.1).

Peer-to-Peer	**client-serveur**
Auto-organisé	Management centralisé
Evolution dynamique, Ad-hoc	Configuré
Découverte des pairs	Consultation de tables
Flux distribué (mesh)	Flux centralisé
Symétrie du réseau	Asymétrie du réseau
Communication par Messages	Orienté RPC
Adressage dynamique au niveau application	Adressage statique @IP
Entités Autonomes	Entités dépendantes
Attaques difficiles (mobilité, anonymat)	Attaques plus simples

TAB. 2.1 – P2P contre Client-Serveur

2.1.4 P2P structuré vis à vis non-structuré

Les systèmes P2P peuvent être largement classifiés en non structuré et structurés. Un système P2P *non-structuré* est un système P2P qui n'impose pas de règle de connexion entre les pairs. Les systèmes P2P non-structurés supportent des opérations simples et efficaces d'arrivée et de départ des pairs parce que les pairs n'ont pas à maintenir une topologie. Ces réseaux reposent sur la génération de graphes aléatoires entre les nœuds. Chaque nouveau nœud se connectant doit connaître un nœud appartenant au réseau, qui lui sert de *bootstrap* pour s'insérer dans le réseau. Les systèmes non structurés (tel que Kazaa[4] et Gnutella[5]) sont concentrés sur des problèmes pratiques tels que la traversés des NAT et des pare-feu. De plus, la recherche (typiquement exécutée par inondation de la requête à tous les pairs voisins) cause un grand coût de trafic du réseau. Les systèmes P2P structurés résolvent ce problème.

D'autre part, les réseaux P2P *structurés* (tels que Chord, CAN et Pastry) doivent maintenir une topologie. Ils définissent un espace de clés et projettent les objets sur cet espace.

[4] "Kazaa : peer-to-peer file sharing software application." http ://www.kazaa.com
[5] "Gnutella.com home page." http ://www.gnutella.com

La plupart des systèmes utilisent une fonction de hachage pour effectuer cette projection et donc sont appelés (*Distributed Hash Tables* ou DHT). Un système P2P structuré distribue la responsabilité des clés aux pairs. Les sous-ensembles de l'espace de clés occupés par les pairs peuvent être disjoints ou non pour des fins de fiabilité. Le système définit une structure de connexion entre les pairs en fonction des clés dont chacun s'occupe. La localisation d'un objet se fait en calculant la clé de l'objet et routant une requête au pair voisin le rapprochant le plus de la cible. Malgré l'absence de connaissance globale, chaque pair peut acheminer correctement la requête à l'aide de la structure de connexion connue. En théorie, un système P2P structuré garantit de trouver n'importe quel objet s'il existe. Il supporte habituellement un routage très efficace dont le coût est souvent O(log n) où n est le nombre de pairs.

2.1.5 Problématique générale

Les systèmes P2P ont connu ces dernières années un immense essor que ce soit au travers des systèmes partiellement décentralisés comme Napster et Kazaa, ou des systèmes complètement décentralisés comme Gnutella ou Chord. La conception d'un système P2P repose principalement sur :
(1) la conception d'un réseau virtuel (overlay network) inter-connectant les membres présents du système ;
(2) un mécanisme de recherche au sein de ce réseau.

La problématique générale des systèmes P2P peut se résumer en une bonne définition de la relation entre le réseau virtuel et le mécanisme de routage, afin d'assurer de bonnes propriétés au système, telles que : recherche rapide d'une donnée, mise à jour rapide du réseau lors du départ ou d'arrivée d'un membre, publication rapide d'une donnée, sécurité et anonymat des transactions. Ces propriétés illustrent bien l'importance stratégique que peut être le contrôle (au moins partiel) d'un système P2P. Le paradigme maître-esclave tend en effet à disparaître peu à peu pour être remplacé par celui plus prometteur (surtout en terme de passage à l'échelle) de P2P.

Certains systèmes, tels que Napster et Kazaa, n'utilisent pas de réseaux virtuels, mais utilisent des index plus ou moins répartis pour fournir directement l'adresse IP d'un membre disposant de la donnée recherchée. D'autres systèmes, comme Gnutella, reposent sur un réseau virtuel non structuré sur lequel la recherche s'effectue par inondation. Enfin, un troisième type de système repose sur un réseau structuré correspondant à une table de hachage distribuée (*DHT : distributed hash table*) sur lequel une stratégie de routage spécifique de la topologie de la DHT est appliquée.

2.2 Les tables de hachage distribuées (DHT)

Beaucoup de solutions de nature distribuée ont été envisagées pour répondre aux problèmes de localisation et de routage dans les réseaux P2P. Une solution reposant sur le principe d'inondation, a été mise en œuvre par Gnutella, mais présentant des performances médiocres. Ainsi la méthode d'inondation s'apparente à une première étape vers la distribution du processus de découverte de ressources mais son principe reste coûteux et non fiable (plus de 100 messages étaient nécessaires pour acheminer une requête de découverte de ressource). Actuellement, la solution la plus étudiée pour découvrir des ressources dans un environnement distribué repose sur le principe d'une DHT (*Distributed Hash Table*) [1] qui implémente de façon distribuée une simple fonction f telle que $f(hash) = data$.

Dans cette partie, nous présentons brièvement les réseaux pair à pair structurés, qui sont les plus adaptés à l'utilisation de la VoIP. Ces réseaux reposent sur des algorithmes de routage *proactifs*. La structure sous-jacente est appelée "*overlay*". Cette dernière est utilisée pour le stockage et la recherche fiable des localisations des utilisateurs. Par la suite nous essayons de dégager les principes communs entre les différents overlays existants comme Pastry, CAN, Viceroy, et Chord.

***Définition* 10.** *Un réseau de recouvrement (overlay network) est un réseau virtuel de nœuds et de liens logiques qui est construit sur un réseau existant dont le but est de fournir un service réseau qui n'est pas disponible dans le réseau existant.*

2.2.1 Protocoles fondés sur l'algorithme de Plaxton

2.2.1.1 Algorithme de routage Plaxton

Plaxton [23] propose un algorithme de localisation de ressources et de routage dans un environnement complètement distribué. Le nommage des objets et pairs est construit de manière aléatoire selon une *fonction de hachage* unique et connue. Chaque identificateur possède une longueur fixe. Il est écrit dans une base identique pour les objets et les pairs.

Chaque pair P maintient une table de routage de plusieurs niveaux. Chaque niveau N de la table contient une liste de voisins de P, dont les N premiers chiffres sont communs à ceux de P. Dans la figure 2.5 chaque nœud possède un identifiant écrit sous forme hexadécimale d'une longueur de quatre chiffres. Le nœud d'identifiant 0325 fait parvenir un message à un nœud d'identifiant 4598, il va alors consulter le premier niveau de sa table de routage et y cherche

un voisin dont le dernier chiffre est 8 (ici le nœud B4F8), puis lui faire suivre ce message. A son tour, ce dernier va consulter le second niveau de sa table de routage et cherche un voisin dont l'identifiant se termine par 98 (ici 9098) et lui faire suivre le message. Ce processus se répète jusqu'à atteindre le nœud considéré. Cette méthode de routage garantit qu'un nœud présent dans un système de N pairs peut être joint en $log_B(n)$ itérations, avec B la base des identifiants.

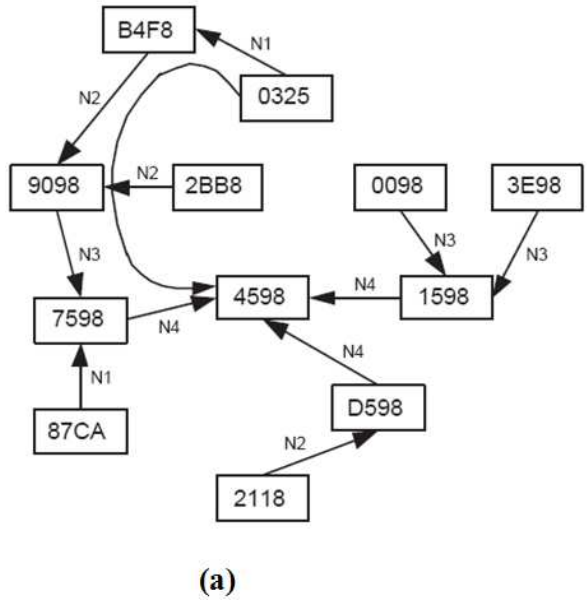


FIG. 2.5 – (a) Exemple de routage selon l'algorithme de Plaxton. (b) Patrons de la table de routage du nœud d'identifiant 0325.

2.2.1.2 Routage dans Pastry

Pastry [31] est un réseau recouvrant purement décentralisé, scalable et s'auto-organise en anneau. Les identifiants de noeud sont choisis aléatoirement et codés sur 128 bits, alors que les identifiants de ressources sont codés sur 160 bits (en rapport avec la taille des hash SHA-1). Les données sont sous la responsabilité du noeud ayant l'identifiant le plus proche numériquement. Le routage s'effectue en propageant le message vers un noeud partageant un plus grand préfixe commun avec la ressource, ou à défaut partageant un préfixe de même taille et plus proche numériquement de la ressource ciblée.

Principe : Chaque noeud possède un *Leaf set* et une table de routage. Le *Leaf set* contient L voisins dans l'espace logique, typiquement 16 ou 32. Il sert à donner au noeud une vision précise de sa localité, tout en permettant à certains noeuds de défaillir sans briser le réseau. Ce *Leaf set* est maintenu à jour de manière agressive. La table de routage quant à elle

est une table par préfixe plus précise pour les noeuds proches dans l'espace virtuel (figure 1.8). Elle est mise à jour de manière passive, en complétant les trous des noeuds défaillants et en remplaçant les entrées par des noeuds plus proches. Pour router un message, un noeud n vérifie d'abord si la ressource r est sous le contrôle de son *Leaf set*. Si tel est le cas, n transmet directement r au noeud responsable. Sinon, n cherche dans sa table de routage un noeud partageant un préfixe plus long avec r pour lui transmettre le message. Si n ne trouve pas de noeud satisfaisant, alors il transmet le message à un noeud partageant un préfixe de même taille que lui avec r, mais qui est plus proche numériquement. Un tel noeud doit exister dans le *Leaf set* si le taux de défaillance du réseau est tolérable. Cet algorithme est en $O(log(N)$.

Exemple : Lorsque le noeud de Bob s'insère dans un réseau Pastry, il lui faut initialiser son *Leaf set*, sa table de routage, et se faire connaître. Dans notre exemple illustré par la figure 2.6, le noeud de Bob d'identificateur 322, envoie un message à la ressource 2106. 322 résout d'abord le premier chiffre et envoie à 231 en utilisant son entrée **2x** ; 231 résout le deuxième chiffre et envoie à 211 en utilisant **21x** ; enfin, 211 résout le dernier chiffre et transmet à 210 en utilisant **210x** (dans Pastry, les identifiants de ressources sont plus longs que les identifiants de noeuds).

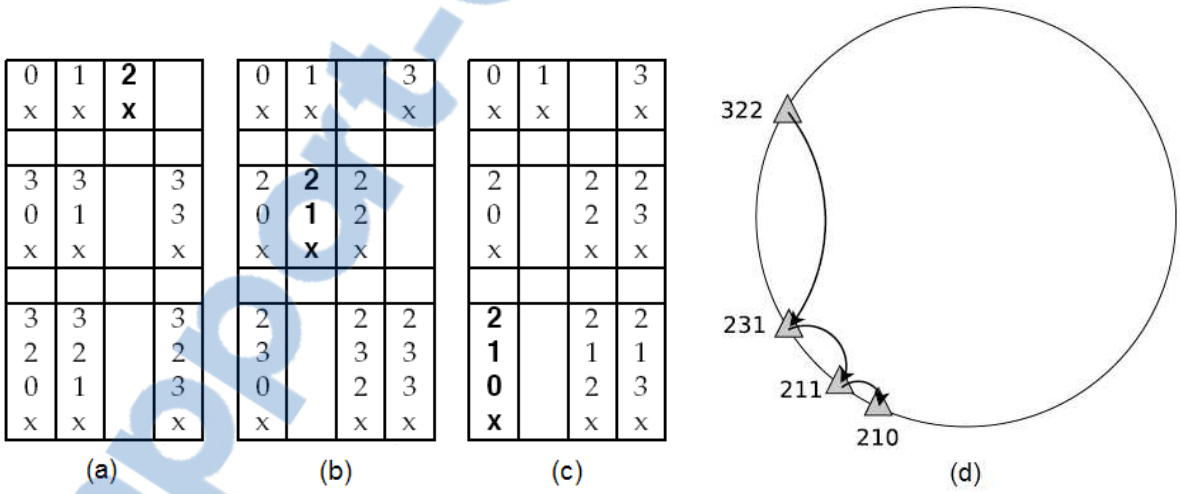


FIG. 2.6 – Exemple de routage par préfixe dans Pastry (simplifié). (a) Table de routage du noeud 322, (b) Table de routage du noeud 231, (c) Table de routage du noeud 211, (d) Chemin du message.

2.2.2 CAN : un protocoles fondés sur un hypercube

CAN [25] propose une structure multidimensionnelle. Pour faciliter la compréhension, nous nous limiterons à deux dimensions dans les explications et les illustrations. CAN four-

nit donc un espace fini à deux dimensions. Chaque noeud est repéré par les coordonnées (x, x', y, y') de l'espace qu'il gère et chaque ressource est repérée par ses coordonnées (x, y) (figure 2.7 a). L'espace est séparé en carrés de tailles variables contenant chacun un noeud. Chaque noeud est responsable de toutes les ressources situées dans son carré. Le routage s'effectue en parcourant l'espace vers la ressource, jusqu'à atteindre le bon carré.

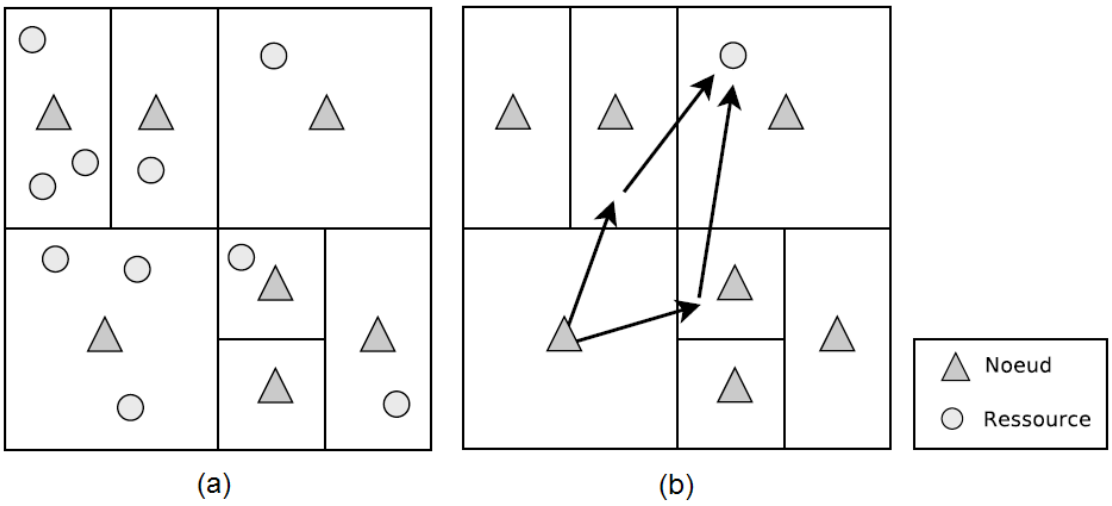


FIG. 2.7 – (a) Structure de l'espace virtuel de CAN en 2 dimensions, (b) Exemple de routage possible.

Routage : Chaque noeud connaît tous ses voisins immédiats dans l'espace virtuel, ainsi que la zone que chacun d'entre eux couvre. Lorsqu'un noeud veut router un message, il choisit de l'envoyer vers un noeud qui le rapproche de sa destination. En effet, plusieurs noeuds peuvent convenir et donc plusieurs routages sont possibles : le noeud cherche à optimiser le routage en envoyant par exemple vers un noeud proche géographiquement (ayant une latence faible). Ces multiples possibilités permettent également à CAN d'être naturellement résistant aux défaillances (figure 2.7 b).

Exemple d'nsertion d'un noeud : Pour s'insérer, un noeud n doit tout d'abord contacter un noeud m, obtenu par un moyen tiers. n choisit un identifiant id (des coordonnées) aléatoirement et m route un message $JOIN$ vers id. n connaît alors le noeud n' responsable de la zone dans laquelle il veut s'insérer. Cette zone est alors découpée en deux, et n et n' obtiennent chacun une moitié. n construit son voisinage en se basant sur celui de n, n' se met à jour pour intégrer n, puis n et n' contactent leurs voisins pour les notifier du changement. L'insertion ne nécessite de communiquer qu'avec un nombre de noeuds très faible.

2.2.3 Viceroy : un protocoles fondés sur les graphes en anneau

Viceroy [20] est un réseau P2P construit comme un réseau *butterfly* approximatif. Chaque pair de Viceroy a un *id* choisi dans l'espace réel [0, 1] et un numéro de niveau l (positif). Le pair choisit l aléatoirement dans l'intervalle $[1, logn]$ où n est le nombre approximatif de pairs. Chaque pair a maintient sept liens :

- *predecessor* et *successor* qui pointent vers les pairs avant et après a dans le cercle des ids [0, 1], respectivement,
- *down-right* qui pointe vers un pair au niveau $(a.l + 1)$ et à une distance approximativement $1/2^{a.l}$ de a,
- *down-left* qui pointe vers un pair au niveau $(a.l + 1)$ et proche de a,
- *up* qui pointe vers un pair au niveau $(a.l - 1)$ et proche de a,
- *next-on-level* et *prev-on-level* qui pointent vers les pairs avant et après a sur le même niveau, respectivement.

Exemple : La figure 2.8 présente un exemple de Viceroy avec 12 pairs. Les pairs s'organisent en 3 niveaux. Pour éviter l'enchevêtrement, elle n'illustre que les liens *down-right* et *down-left*. Le routage vers une clé k inclut trois étapes : (1) il suit les liens *up* pour monter jusqu'au niveau 1 ; (2) si la cible n'est pas touchée, le routage descend de niveau à niveau en choisissant un lien *down-right* ou *down-left* qui s'approche le plus de k (à chaque niveau i, la distance du pair courant à k doit être au plus $1/2^{i-1}$) ; et (3) si la cible n'est pas encore touchée, le routage l'atteint via les liens *next-on-level*, *prev-on-level*, *predecessor* et *successor*.

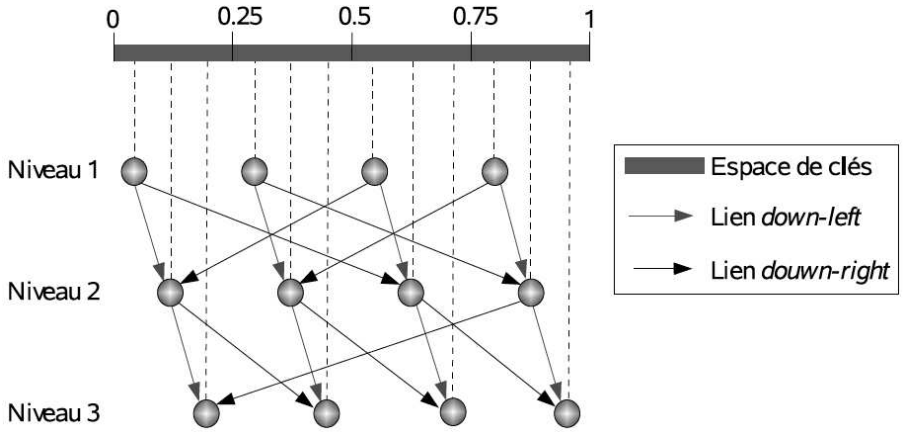


FIG. 2.8 – Exemple de topologie Viceroy.

Malgré un nombre constant (7) de liens sortant d'un pair, Viceroy fournit un routage avec le coût logarithmique. C'est l'avantage de Viceroy en comparaison avec les systèmes précédents.

2.2.4 Le protocole Chord

Organiser les pairs selon un anneau est envisagé dans plusieurs travaux. Tout d'abord dans Chord qui propose d'organiser les pairs selon un anneau simple. Ensuite, dans Viceroy qui est une version améliorée de Chord, reposant sur les réseaux Butterfly. On peut noter que de nombreuses DHTs utilisent de manière plus ou moins directe l'organisation des pairs selon un anneau. C'est par exemple le cas de CAN dont la topologie est un hypercube torique, ou de Pastry qui utilise un anneau pour finaliser son routage. Néanmoins, nous avons choisi de ne pas détailler ces infrastructures dans la section précédente mais plutôt dans celle qui suit, nous présentons le protocole Chord en détail.

2.2.4.1 Aperçu

Le principe clé de Chord est d'utiliser une fonction de hachage qui soit à la fois rapide et répartie uniformément afin que la charge soit égale sur chaque pair du réseau. Un tel hachage est dit *consistant*. De plus, lorsqu'un pair quitte ou entre sur le réseau, il est hautement probable que seulement $1/N$ (N nombre total de pairs) clés soient déplacées. La scalabilité de Chord est maintenue grâce à la gestion d'une table de routage ne contenant que m entrées (m nombre de bits d'une clé). Grâce à cela, on s'assure qu'une requête sera effectuée via seulement $\log(N)$ pairs.

2.2.4.2 La fonction de hachage

La fonction de hachage assigne à chaque nœud et à chaque clé un identifiant de m bit. La valeur m doit être bien entendu assez grande pour que la probabilité de collision entre identifiants soit faible. L'identifiant d'un pair peut être créé par exemple en hachant son adresse IP ou bien son nom d'hôte. La politique d'assignation d'une clé hachée à un pair est simpliste. La clé est associée au premier nœud dont la valeur de l'identifiant est inférieur ou égale à la valeur de la clé. Ce nœud particulier est appelé successeur.

Comme on peut le voir sur la figure 2.9, on représente le réseau Chord sous la forme d'un anneau. Chaque nœud est marqué par la lettre N (pour node) suivie de son identifiant. Les clés stockées sur les différents nœuds sont représentées par des cadres à l'intérieur desquels, on retrouve la lettre K (pour Key) suivie de l'identifiant de la clé. Une flèche indique qu'une clé donnée est sur un nœud donné. Ici, on peut voir que la clé $K10$ est stocké sur le nœud $N14$ car $N14$ est le nœud le plus proche supérieur à $K10$. L'un des principaux intérêt de

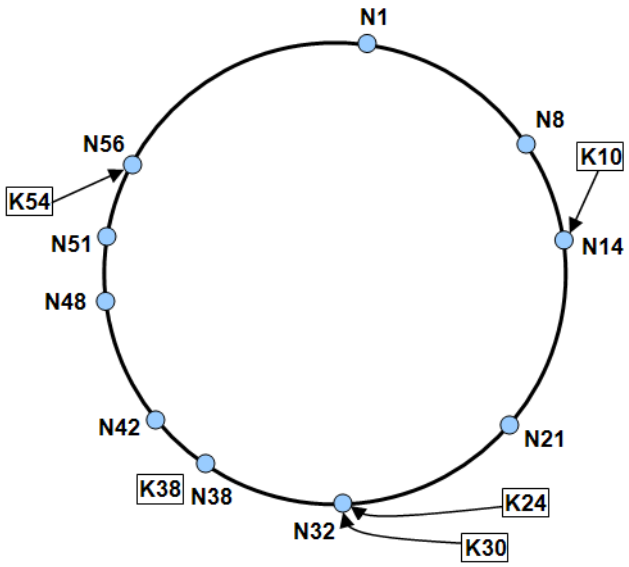


FIG. 2.9 – Répartition des clés

cette fonction de hachage est de minimiser les déplacements des clés entre pairs lorsque de nouveaux pairs arrivent ou sortent du réseau. Afin que les clés soient équitablement réparties, lorsqu'un nouveau nœud apparaît sur le réseau, son successeur lui transmet automatiquement les clés dont il sera à charge. Il est prouvé que le nombre de clé déplacée est environ égale à $1/N$ (N nombre de pairs). Dans la figure 2.9, si un nouveau nœud d'identifiant 26 venait à se connecter, il capturait toutes les clés d'identifiant compris entre 21 et 26. On utilise fréquemment comme fonction de hachage le $SHA-1$ qui possède de très bonnes propriétés de distribution. Il est malgré tout possible de générer des clés qui soient maladroitement réparties par exemple en fournissant à la fonction de hachage des clés particulières qui cibleront le même pair. On considéra ce cas comme hautement improbable.

2.2.4.3 Assignation simple des clés

Il s'agit de la plus simple implémentation de Chord. Il n'y a pas de table de routage. Chaque nœud ne communique qu'avec son successeur à la manière d'une chaîne. Lorsqu'une requête est formulée par exemple lors de la recherche d'une clé, chaque nœud regarde si la clé recherchée est inférieure ou égale à l'identifiant de son successeur. Si cela s'avère vrai alors, le nœud renvoi l'identifiant de son successeur au nœud qui est à l'origine de la requête sinon la requête est transmise au successeur direct qui sera plus à même de répondre.
Dans l'exemple de la figure 2.10, le nœud 8 recherche quel nœud détient la clé 54. Comme la clé n'est pas en sa possession, la fonction de répartition lui renvoie le nœud 14 qui pourra répondre à la requête . Le processus se répète sur chaque nœud jusqu'au nœud 56 qui

possède la clé 54. Il s'agit d'une implémentation simple et très peu efficace. Effectivement, la complexité est de l'ordre de N. Si une clé se trouve sur le dernier nœud, la requête passe par tous les nœuds. Nous allons donc voir dans la prochaine section comment optimiser cet algorithme.

Algorithm 1 Algorithme simple de recherche (*Chord*)

```
// ask node n to find the successor of id
n.find_successor(id)
if (id ∈ [n, successor])
 return successor;
else
// forward the query around the circle
 return successor.find_successor(id);
```

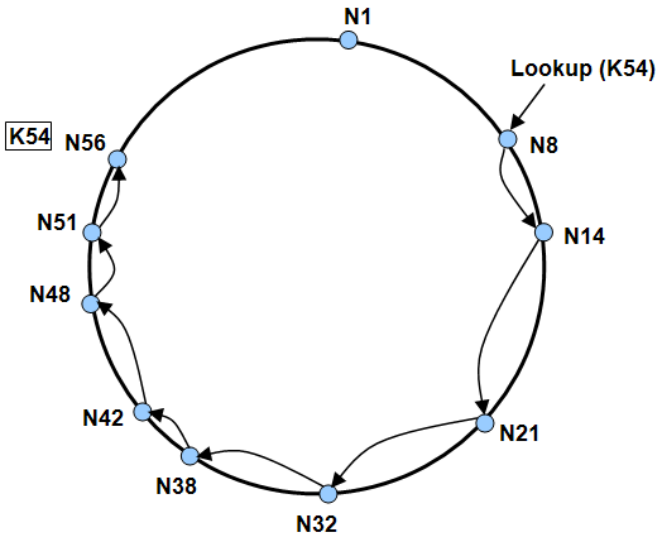


FIG. 2.10 – Répartition simple des clés

2.2.4.4 Assignation optimale des clés

Dans le but d'optimiser l'algorithme précèdent, il est nécessaire de gérer une table de routage qui permette une transmission des requêtes bien plus efficace. Grâce à cela, on pourra sauter des pairs dont on sait qu'ils ne pourront pas répondre à la requête limitant ainsi le nombre de pairs parcourus. Cette table comporte m (le nombre de bits d'une clé) entrées qui à un identifiant de clé donnée relie un identifiant de nœud. Elle est créée en utilisant une incrémentation exponentielle. Chaque entrée dans la table est de la forme $n + 2^{i-1}$ avec $1 \leq i \leq m$ avec m le nombre de bits d'une clé et n l'identifiant du nœud qui gère la table. La valeur associée dans la table est le successeur qui pourrait géré cette clé donc $successeur(n + 2^{i-1})$. Il est facile de voir que cette table optimise la recherche car elle

recouvre l'ensemble des clés utilisées. On nomme communément cette table *finger*. Dans le schéma suivant, le premier élément $finger[1]$ est le successeur direct du nœud possédant la table.

L'exemple de la figure 2.11 illustre le fonctionnement de la table de routage pour le nœud 8. On constate que le premier élément de la table est son successeur direct comme $successeur(8 + 2^0) = 14$. De manière similaire, on remarque que le dernier élément est le nœud 40 car $successeur(8 + 2^5) = 40$. Il est à remarquer que cette table ne stocke que les nœuds nécessaires pour un parcours en temps logarithmique du réseau. Afin de mieux comprendre le principe de fonctionnement de cette table de routage, analysons le second schéma de droite de l'exemple (figure 2.11). Le nœud 8 recherche le successeur de la clé 54. Étant donné que le plus grand identifiant précédent cette clé dans la table finger est le pair 42, celui-ci sera alors chargé de répondre à la requête. A son tour, le nœud 42 recherche dans sa table l'identifiant le plus proche de la clé et lui renvoie la requête, en l'occurrence, il s'agit du nœud 51. Celui-ci constate alors que son successeur a la charge de cet identifiant, la recherche est terminée.

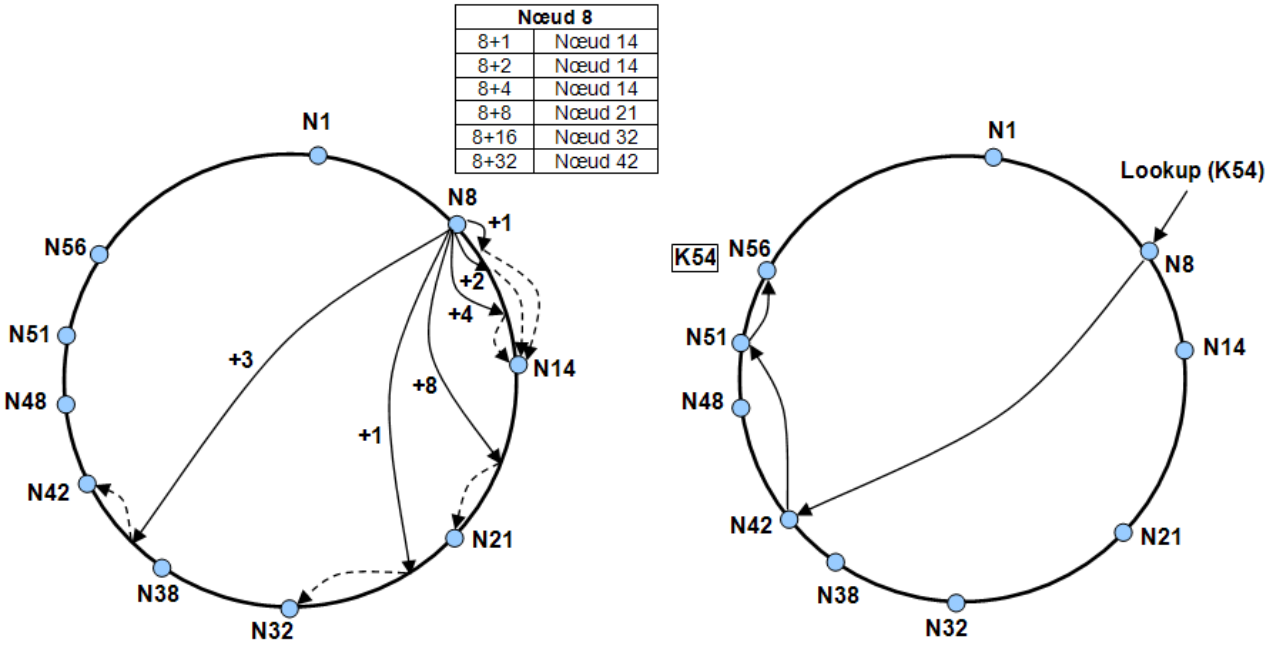


FIG. 2.11 – Répartition des clés

Le code suivant illustre comment compléter l'algorithme de recherche de clé simple pour qu'il se serve de la table finger (la table de routage). Lorsque le nœud effectuant la recherche ne parvient pas à trouver de successeur valide, il effectue une recherche du nœud précédent le plus proche de l'identifiant voulu. La requête est alors envoyée à ce nœud. Comme on peut le constater, on a fortement réduit le nombre de nœuds participant à la recherche par rapport à l'implémentation simple du protocole Chord. En réalité, la recherche se comporte

de façon dichotomique, réduisant par deux à chaque passage dans un nœud l'espace entre la clé recherchée et le nœud courant.

Algorithm 2 Algorithme de recherche optimisé (*Chord*)

```
// ask node n to find the successor of id
n.find_successor(id)
if (id ∈ [n, successor])
 return successor ;
else
n' = closest_predicting_node(id)
 return n.find_successor(id) ;
// search the local table for the highest predecessor of id
n.closest_predicting_node(id)
for i = m downto 1
 if (finger[i] ∈ (n, id))
 return finger[i] ;
return n ;
```

2.2.4.5 Opérations de maintenance

Nous avons vu précédemment comment Chord répartissait les clés parmi les pairs du réseau. Le principe reste le même que l'on soit en mode recherche ou insertion de clé, le but étant de rechercher le successeur, le nœud responsable de la clé à insérer ou à rechercher. Cependant, en soit le réseau n'est pas encore viable car il reste à prendre en compte l'un des problèmes les plus importants et les plus récurrents des réseaux pair-à-pair, à savoir les entrées et les départs du réseau. En outre, nous verrons quels sont les changements que ces mouvements entraînent au sein de la table de routage et de stockage des clés.

Entrées et Stabilisation

Le protocole Chord fournit une fonction de création exécutée uniquement au moment de la création du réseau Chord par le tout premier nœud du réseau. La fonction fixe simplement le successeur du nœud sur lui-même, ainsi il supportera toutes les insertions et recherches de clés jusqu'à l'arrivée des prochains nœuds. La seconde fonction que le protocole Chord fournit est bien entendu une fonction d'entrée dans le réseau que l'on nomme communément join. Le principe de fonctionnement est des plus simples ; cela agit comme si on tentait d'insérer une clé sauf qu'il s'agit d'un nœud. Cette fonction renvoie donc le successeur du nœud qui veut entrer dans le réseau. Cependant, il est clair que les choses ne peuvent rester en l'état. L'insertion d'un nouveau pair entre deux nœuds provoquent un défaut de successeur au niveau du nœud précédent. Il est donc nécessaire de l'avertir de ce changement. De plus, ce problème peut être généralisé à l'ensemble des opérations provoquant des changements

du nombre de pairs (départ volontaire ou involontaire, entrées). On met donc au point une fonction de vérification périodique dont le but est de garantir que le prédécesseur du nœud successeur sur lequel le nœud pointe est bien lui-même. Cette fonction est appelée stabilize.

Impact de l'entrée d'un nœud sur la recherche d'une clé

Nous allons étudier dans cette section, les conséquences qu'entraîne l'entrée d'un ou plusieurs nœuds sur le réseau. L'entrée d'un nœud sur le réseau alors qu'une recherche est en cours mène à trois situations différentes :

- La recherche de la clé n'intervient pas sur cette partie de l'anneau du réseau, ses performances restent optimales, à savoir en log(N).
- La recherche intervient alors que les tables fingers des pairs ne sont pas encore à jour. Dans ce cas, la recherche est ralentie mais reste possible. L'analyse de la baisse de performance entraînée dans cette situation est faite juste après le troisième performances restent malgré tout en log(N).
- Enfin, la recherche intervient alors que les pointeurs successeurs ne sont pas encore à jour dans ce cas la recherche échoue.

En clair, lorsque la recherche ne porte pas sur la partie du réseau modifiée, les performances du réseau au moment de la recherche restent optimales ce qui est logique. En outre, l'impact des modifications sur une recherche de clé alors que les tables fingers ne sont pas à jour peut être considéré comme minime.

Nœuds défectueux

Étudions à présent le cas des nœuds qui échouent dans le réseau. Cela se produit lorsque le nœud se déconnecte violemment. Dans ce cas, il est nécessaire, malgré le nœud faisant défaut, que les requêtes puissent toujours être effectuées. Un nœud défectueux peut soit apparaître en tant que successeur direct d'un nœud, soit figuré dans la table de routage. Dans les deux situations le comportement a adopté est assez simple et n'entraîne qu'une modification mineure des algorithmes présentés précédemment. Il suffit d'envoyer la requête au plus proche des nœuds valides dans la table finger par rapport au nœud défectueux. Ensuite grâce aux algorithme de mise à jour, les tables fingers et le nœud dont le successeur est défectueux corrigeront d'eux-mêmes les incohérences. On constate que la chaîne n'est pas brisée, de plus on transmet toujours la requête au nœud le plus proche, les performances restent donc en log(N). Bien entendu, il est ici question des requêtes ne concernant pas le nœud défectueux puisqu'on a aucune chance de récupérer les clés stockées sur ce nœud.

2.2.5 Chord^2 : un nouveau protocole à deux couche

Dans [17] Y. J. Joung et J. C. Wang, proposent une structure à deux couches appelées Chord^2. La couche inférieure est l'anneau régulier de Chord, alors que la couche supérieure est un anneau pour la maintenance. Comme dans Chord, nous laissons les nœud de la couche régulière explore périodiquement leurs successeurs, qui coûte seulement $O(1)$ messages par activation. Quand un changement est détecté, certains super nœud seront informés pour aviser des nœud affectés ($O(logN)$ en moyenne) de mettre à jour leurs tables de routage, de ce fait les coûts de maintenance de l'anneau régulier se réduit à $O(logN)$. La couche de maintenance est également mise en application comme l'anneau de Chord. Cependant, puisqu'elle est construite des super nœud qui sont plus stables que les nœud ordinaires, les coûts de maintenance de l'anneau sont relativement bas comparés à un anneau équivalent aux caractéristiques diverses.

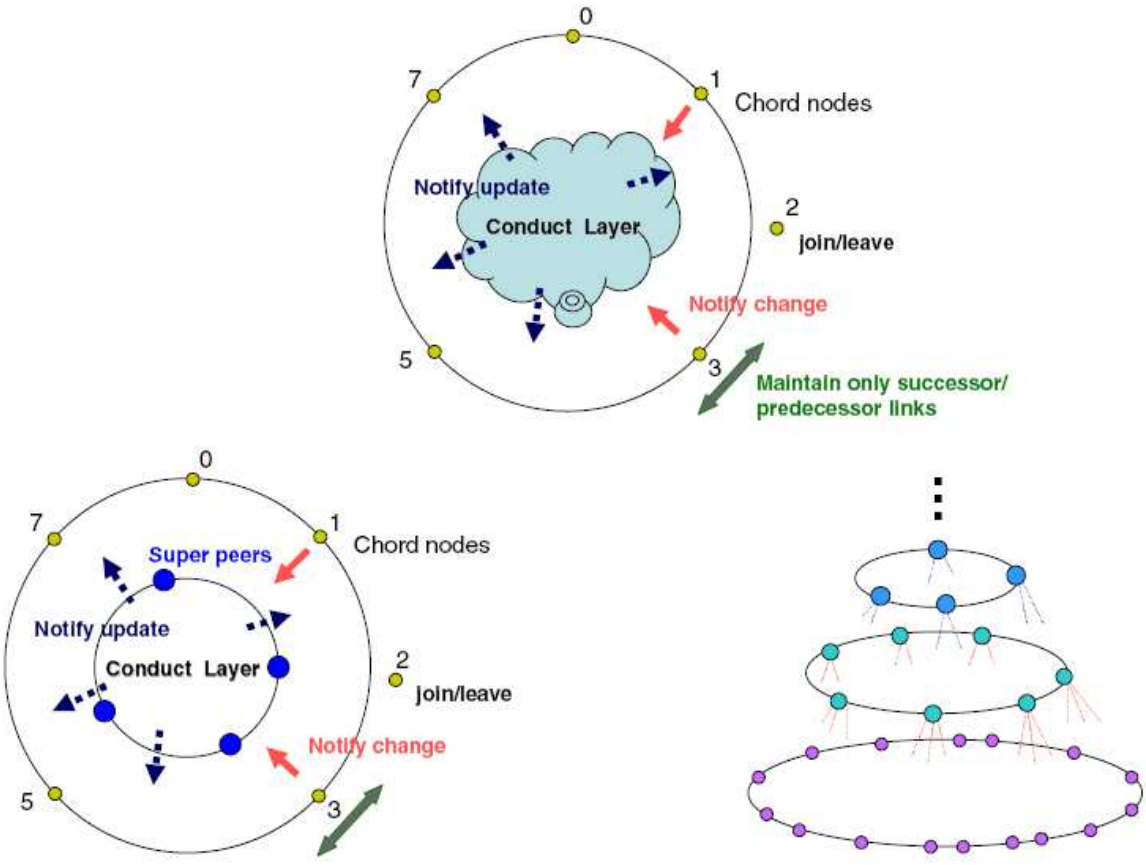


FIG. 2.12 – Séparation de Chord et de la maintenance des fingers

2.2.6 Comparaison des caractéristiques théoriques

Des comparaisons des performances théoriques des différentes DHTs ont été effectuées dans de nombreux articles dont [8, 12, 21, 22, 24]. De même, l'étude du comportement des DHTs face à la dynamicité est largement étudiée [16, 19, 26]. Le tableau 2.2 recense les principales propositions de DHTs ainsi que leurs propriétés théoriques. On y a inscrit le modèle topologique, le degré de connectivité, le nombre de sauts moyen pour acheminer une requête et le taux de congestion des pairs. Cette dernière caractéristique représente le nombre de clés dont un pair est responsable. La première remarque concernant cette vue synthétique des caractéristiques théoriques des DHTs réside dans la similitude des propriétés. En particulier, le taux de congestion des pairs qui, à part pour CAN, est le même pour les différentes approches. De même, le nombre de sauts moyen pour router une requête s'exprime toujours en $O(\log N)$.

DHT	Topologie	Degré de connectivité	Nombre de sauts	Congestion
Pastry	Plaxton	$O((b\text{-}1) * \log_b N)$	$O(\log_{2^b} N)$	$O((\log N)/N)$
CAN	Hypercube	$O(d)$	$O(n^{\frac{1}{d}})$	$O(d.n^{\frac{1}{d-1}})$
Chord	Anneau	$O(\log_2(N))$	$O(\log N)$	$O((\log N)/N)$
Viceroy	Butterfly	$O(1)$	$O(\log N)$	$O((\log N)/N)$

TAB. 2.2 – Extrait de [11]. Comparaison des caractéristiques théoriques de différentes DHTs

2.3 Aspects de sécurité dans les réseaux P2P structurés

Les réseaux structurés présentent un grand nombre de similitudes, la principale différence résidant dans l'algorithme de routage utilisé. Ces réseaux partagent donc également certaines faiblesses. L'attribution des identifiants est un point clé de la sécurité du routage et de la disponibilité [37]. En effet, un ensemble d'identifiants bien choisis permet toujours de modérer l'accès d'un nœud ou de contrôler une ressource. Il apparaît également qu'une trop grande liberté des entrées dans les tables de routage (Pastry) pose des problèmes de sécurité, un adversaire pouvant alors occuper une part non négligeable de cette table, même avec des identifiants de nœuds sécurisés. L'utilisation de routage aléatoire ou vers plusieurs réplicats augmente les chances de trouver une ressource valide. Pour le cas du routage vers plusieurs réplicats, la stratégie de réplication est importante, afin d'éviter qu'un réplicat corrompu ne se propage. Notamment, à l'insertion d'une nouvelle clé, la réplication peut être gérée par le nœud responsable de la clé ou par le nœud qui veut l'insérer. Dans le cas ou c'est le nœud responsable qui s'en charge et si ce nœud est corrompu, la réplication initiale peut également être corrompue.

2.3.1 Routage sécurisé dans un réseau structuré

Un premier point faible des réseaux Pair-à-Pair réside dans le routage. En effet, le routage demande la collaboration de l'ensemble des nœuds traversés. Un nœud malicieux peut ainsi assez facilement modifier ou effacer un message lors du routage. Avec la présence de plusieurs nœuds malicieux, il devient possible de censurer des informations, d'exclure des utilisateurs, ou d'insérer des utilisateurs dans un "faux" réseau contrôlé par l'attaquant (attaque de type *phishing*). Dans [6], une méthode de routage sécurisé est proposée, garantissant que chaque message arrive inaltéré à l'ensemble des nœuds sains possédant la clé demandée, avec une grande probabilité. Dans cette méthode, les nœuds malicieux sont des processus *byzantins* supposés pouvoir agir dans un but commun (*coalition*). Cette méthode se décompose en trois parties : l'assignation sécurisée de *nodeIds*, la mise à jour sécurisée des tables de routage et la retransmission sécurisée.

2.3.1.1 Assignation sécurisée des *nodeIds*

Dans les systèmes à DHT, les nœuds choisissent leur *nodeId* aléatoirement. Cependant, il est tout à fait possible pour un nœud de forcer un *nodeId* spécifique. En forçant ce *nodeId*, une coalition de nœuds peut s'arranger pour contrôler une clé ainsi que toutes ses répliques et ainsi censurer cette clé. Avec des *nodeIds* bien choisis, une coalition peut aussi maîtriser intégralement la table de routage d'un nœud et ainsi avoir un rôle de médiateur dans tous ses accès au réseau.

Afin de garantir un véritable aléa dans les *nodeIds*, la solution la plus simple est l'utilisation d'un serveur centralisé. Ce serveur est responsable de fournir un certificat signé, contenant un *nodeId* choisi aléatoirement. Chaque nœud peut ensuite vérifier que le nœud avec lequel il communique possède bien un certificat signé. Cette opération ne demandant pas la collaboration du serveur de certificats, ce serveur n'est nécessaire que pour l'initialisation : le passage à l'échelle et la disponibilité du réseau ne sont donc pas remis en cause.

Cependant, rien n'empêche un nœud d'obtenir une grande quantité de certificats, afin de reproduire les attaques présentées. Ceci est l'*attaque sybile*. Il y est présenté que sans serveur centralisé, il n'est pas réalisable de se prémunir de ce genre d'attaques. Dans le cas d'un serveur centralisé, afin de limiter le nombre de certificats qu'un nœud peut obtenir, une première solution est de fournir des certificats associés à une identité physique. Une autre solution repose sur des *crypto-puzzles*, requérant un temps de calcul du client pour obtenir un certificat (et compliquant ainsi l'obtention d'un grand nombre de certificats).

2.3.1.2 Mise à jour sécurisée des tables de routage

Lorsque le réseau contient une proportion f de nœuds malicieux, la table de routage de chaque nœud devrait contenir une proportion f de nœuds malicieux également. Or, étant donné les optimisations des tables de routage par rapport à la localité (utiliser le nœud le plus proche physiquement dans chaque entrée de la table de routage), un attaquant peut exploiter cette localité pour insérer une proportion de nœuds malicieux supérieure à f.

La solution retenue repose sur des *tables de routage contraintes*. Chaque entrée de la table est contrainte à un nœud de *nodeId* de même suffixe que le nœud courant. Ainsi, étant donné que les *nodeIds* sont générés aléatoirement, un attaquant ne peut plus profiter de la localité pour augmenter la proportion de nœuds corrompus dans une table de routage.

2.3.1.3 Retransmission sécurisée

La mise à jour des tables de routage se passant de manière sécurisée, chaque table de routage contient une proportion f de nœuds malicieux. Supposant que le routage vers une clé prend h sauts en moyenne, un routage est réussi avec une probabilité (1 - f)h-1 (< 1 - f) (chaque nœud sur la route peut être malicieux et corrompre le message). Le mécanisme proposé se passe en deux temps. Dans un premier temps, l'expéditeur fait un envoi normal avec test d'échec de routage, puis, si besoin, il fait une transmission sécurisée selon diverses routes.

Un nœud malicieux peut réaliser deux actions : jeter le message, ou retourner une réponse falsifiée. Pour se prémunir d'un jet de message, l'expéditeur A initialise un chronomètre à l'envoi et considère le message perdu au bout de s secondes. Pour détecter une réponse falsifiée, l'expéditeur A réalise un test sur le *nodeId* du nœud responsable de la clé demandée et des nodeIds des répliques. Le test repose sur le fait que, les *nodeIds* étant aléatoires, la densité de nœuds en toute zone de l'espace virtuel est supérieure à la densité de nœuds malicieux ; de plus, la densité est constante dans tout l'espace virtuel. A calcule donc la densité dans la zone couverte par lui et ses voisins directs (connus dans la table de routage), Dl. Puis, dans les overlays où les répliques sont situées sur les nœuds voisins du nœud responsable (Pastry par exemple), A calcule la densité dans la zone délimitée par les répliques Dr(pour d'autres overlays, des systèmes équivalents existent). A compare ensuite Dl et Dr, et si Dr " Dl, A peut en déduire que la réponse a été falsifiée.

2.4 Skype : une application de ToIP en mode P2P

Skype [2], de part sa simplicité d'utilisation et son excellente qualité d'écoute, est récemment devenu un logiciel de téléphonie sur Internet incontournable. C'est un protocole propriétaire basé sur l'architecture Kazaa, il supporte la messagerie instantanée ainsi que les conférences audio. L'originalité de l'architecture du système (P2P) et sa popularité grandissante nous ont amenés à nous intéresser à son fonctionnement général, malgré le peu de documentation disponible sur ce sujet.

L'avantage principal de Skype [35], est qu'il implémente l'équivalent des serveurs STUN et TURN dans le noeud lui-même pour manipuler les NAT, à la différence de la configuration explicite de serveur dans les applications SIP existantes. Skype distingue entre les supernoeuds et les noeuds ordinaires dans son architecture aussi.

Les services disponibles sur Skype sont actuellement :
- Communication de type "PC to PC" entre deux terminaux équipés de Skype,
- Communication de type "PC to Phone" entre un terminal équipé de Skype et un téléphone traditionnel (SkypeOut),
- Communication multi-utilisateurs (ou conférence) entre terminaux équipés de Skype,
- Echange de fichiers,
- Messagerie instantanée,
- Gestion d'une liste de contacts (buddy list).

L'ensemble de ces services est gratuit à l'exception de SkypeOut.

2.4.1 Architecture de Skype

Skype s'appuie sur une architecture P2P pour transmettre la voix sur Internet (voir figure 2.13). A l'instar de Kazaa, Skype repose sur un ensemble de "super-nœuds" (supernodes, SN). Tout client Skype (CS) possédant une adresse IP publique peut tenir le rôle de SN si ses capacités en terme de CPU et de bande passante le permettent. Chaque client Skype (SC) maintient une table pour sauvegarder les adresses IP et les numéros des ports des supers nœuds appelée le cache des hôtes (HC). Il utilise TCP pour la signalisation et UDP/TCP pour le transfert des données. Le trafic de médias et la signalisation ne sont pas envoyés sur le même port.

Host Cache (CS) : Chaque CS stocke et met à jour régulièrement une liste d'adresses

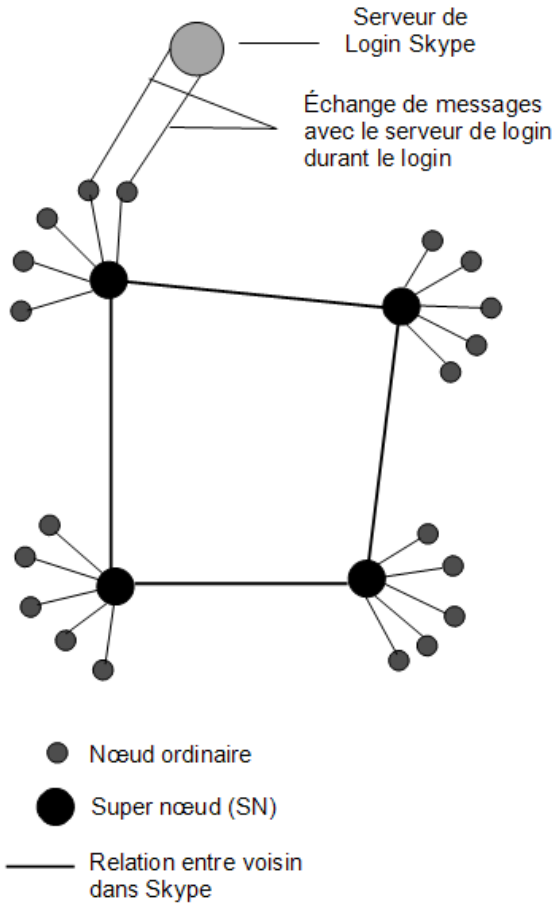


FIG. 2.13 – Architecture de Skype [2]

de SN et les ports correspondants. Dans la version actuelle, cette liste, appelée Host Cache (HC), est stockée avec d'autres informations dans un fichier XML1. Cette liste est constituée d'au plus 200 entrées. Au moins une de ces entrées doit être valide pour permettre la connexion au réseau.

Codecs audio : D'après [2], Skype utilise iLBC, iSAC (qui proposent une qualité et une tolérance aux fautes supérieures au codec G.729) ou un troisième codec inconnu. Ils laissent passer les fréquences comprisent entre 50 et 8000 Hz. Ces codecs large bande permettent de disposer d'une qualité d'écoute raisonnable même avec une bande passante réduite (32 kbps).

Liste de contacts : La liste de contacts est chiffrée dans des fichiers localisés sur la machine de l'utilisateur.

Chiffrement : Skype utilise AES[6] (***A****dvanced* ***E****ncryption* ***S****tandard*) avec une clé de 256 bits pour le chiffrement de toutes les communications. L'échange des clés symétriques AES est protégé par l'utilisation de l'algorithme RSA[7] avec des clés de 1536 à 2048 bits. Les

[6] AES : algorithme de cryptographie symétrique
[7] RSA : algorithme de cryptographie asymétrique

clés publiques des utilisateurs sont certifiées par Skype via le serveur d'authentification.

NATs et Pare-feux : Le CS utilise vraisemblablement une variation des protocoles STUN et TURN afin de déterminer, le cas échéant, le type de NAT et de pare-feu derrière lequel il se situe. Cette détection a lieu à l'établissement de la connexion et semble être vérifiée de manière périodique.

2.4.2 Fonctionnement

1. **Connexion au réseau :** Lorsqu'un CS désire se connecter au réseau, il tente de joindre l'un des SN de sa liste HC en UDP . Si la connexion ne peut pas être établie, le CS va essayer successivement de se connecter en TCP via les ports 80 (HTTP) et 443 (HTTPS) ; ceci permet la traversée d'éventuels pare-feux ou NAT bloquant les trafics UDP ou certains ports TCP. Si la connectivité ne peut être obtenue avec aucun des SN, la phase de connexion échoue.
2. **Login :** Durant cette phase, le client Skype s'authentifie par son nom d'utilisateur et son mot de passe, dans le serveur de login (*skype.com*), donc il déclare sa présence à ses amis ainsi que les autres peers. Il détermine le type de NAT ou de Firewall derrière lesquels il est et il découvre les autres nœuds Skype qui sont connectés. La figure 2.14 décrit l'organigramme de login dans Skype.
3. **Recherche d'un utilisateur :** Vue que skype n'est pas un protocole ouvert et les messages sont cryptés, ainsi que les communications ne pouvant être tracées au delà du SN, le processus de recherche d'un utilisateur reste partiellement inconnu.
4. **Gestion d'un appel :** Si le correspondant est présent dans la liste de contacts, la connexion est établie directement via un échange TCP (signalisation). Dans le cas contraire, l'établissement de la connexion est précédé par la phase de recherche d'utilisateur.
5. **Conférence :** Considérons une conférence à trois personnes. Le participant qui possède les ressources les plus importantes est élu pour jouer le rôle de "mixer". Notons qu'il n'est pas forcément l'initiateur de la conférence. Le "mixer" est le cœur de la conférence. Toutes les sources lui envoient leurs paquets de voix. Il se charge alors de les multiplexer de manière adéquat avec les siens, puis de les transmettre aux participants

2.4.3 Limites du Skype

Depuis 2003, le nombre d'utilisateurs de Skype croît considérablement ; Malgré cet essort, nous avons dégagé quelques limites de Skype :

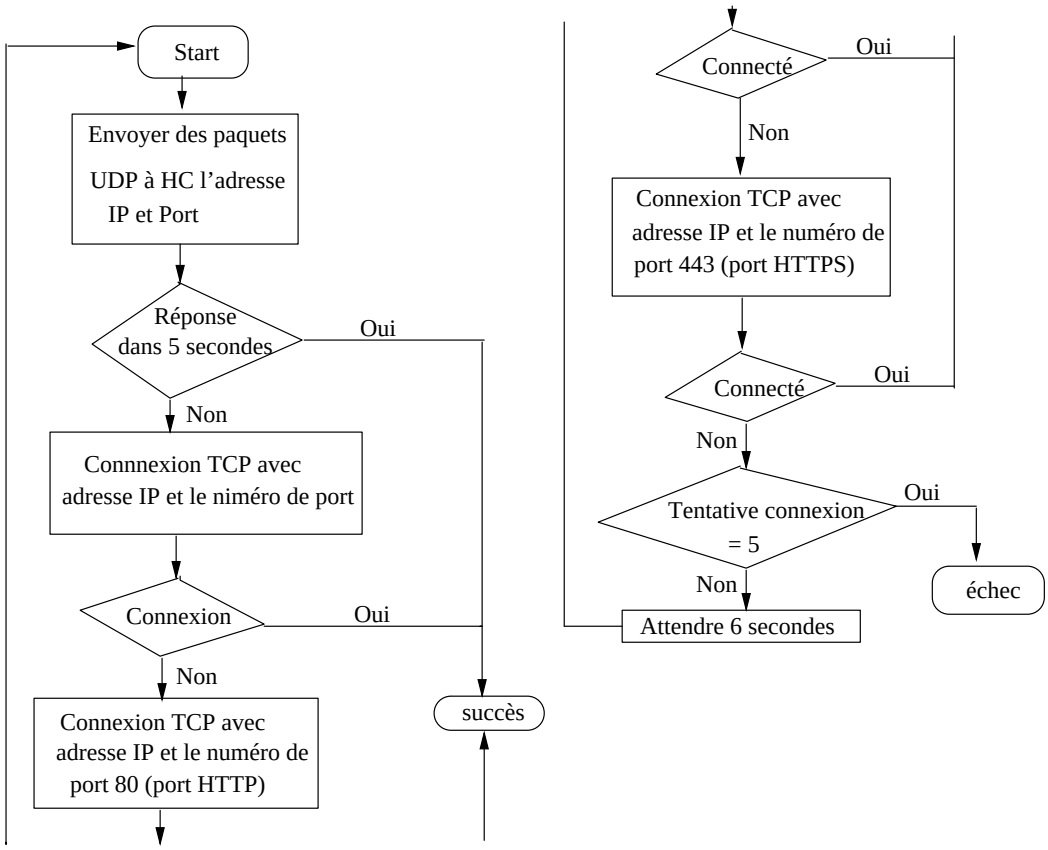


FIG. 2.14 – Organigramme de login dans Skype

- Le caractère propriétaire limite son utilisation dans un cadre professionnel. Pour ces raisons, Skype fait face à de sérieux concurrents, notamment ceux se reposant sur le protocole SIP (rfc 3261)
- Skype ne propose aucun service de téléphonie classique comme répondeur, numéros d'appel d'urgence, renseignements... Notons que l'architecture même du système rend difficile le déploiement de ce genre de service sans passer par un serveur central (informations de localisation, stockage des messages ...).
- De plus, la phase d'authentification étant centralisée, le serveur d'authentification semble donc être le point de vulnérabilité du système. Si un utilisateur malveillant arrive à mettre hors service ce serveur, aucun nouvel utilisateur ne pourra plus se connecter.

Skype, du fait de sa gratuité, de sa simplicité d'utilisation ainsi que de ses très bonnes performances, démocratise la voix sur IP. Cette technologie s'appuie sur les réseaux communautaires pour palier l'absence de qualité de service de l'internet actuel. De part ses partenariats engagés avec le monde des télécoms, Skype s'ouvre sur des perspectives d'évolution intéressantes (SkypeOut, SkypeIn).

2.5 Problèmes des réseaux P2P

Le modèle P2P n'a pas que des avantages. Parmi les problèmes des systèmes P2P on trouve :

- **Interopérabilité :** dans les réseaux P2P, Les applications utilisent différentes technologies réseaux ainsi que différentes plateformes. Le fait que plusieurs plateformes puissent cohabiter au sein de celles-ci, avec différents systèmes de sécurité, rend l'interopérabilité difficile.
- **La performance des réseaux :** La latence dans les réseaux de communications à cette échelle est très importante, surtout lorsque les messages sont relayés entre réseaux de nature différente. Les systèmes de cache améliorent le temps d'accès aux données, mais introduisent de nouveaux problèmes de cohérence sur ces données.
- **La gestion de la cohérence :** Les systèmes distribués à plus petite échelle peuvent appliquer des modèles de cohérence relativement fort avec la possibilité de synchroniser tous les nœuds pour réaliser les mises à jour. Si un nombre important de copies de la donnée est à mettre à jour, la latence au travers du réseau peut rendre cette opération particulièrement prohibitive.
- **La sécurité :** La sécurité des données est souvent réalisée au niveau applicatif par encryptage des données. Certains systèmes prennent en compte la possibilité de présence d'un pair malicieux dans leurs protocoles de cohérence, par exemple dans OceanStore. D'autres systèmes s'attachent à assurer l'anonymat des utilisateurs. Ces contraintes vont à l'encontre des objectifs de performance et sont à la base des mécanismes mis en œuvre dans un système tel que Freenet.
- **Problème des firewall :** Un problème majeur pour la communication entre pairs est l'existence de pare-feu (firewall). Celui-ci empêche la création d'une communication vers les machines qu'il protège si c'est à l'initiative d'une machine extérieure. Une conséquence est que si deux machines d'un système pair à pair qui veulent communiquer sont toutes deux placées derrière des pare-feu, il leur est impossible d'ouvrir un canal de communication.
- **Problème de localisation et de routage :** L'utilisation du modèle P2P, avec ses caractéristiques de décentralisation et de dynamicité pose un problème très simple : comment découvrir et accéder à des ressources dans un tel contexte ? En effet, étant donné le comportement dynamique des pairs, il est très difficile d'obtenir une vue globale de l'ensemble des ressources d'une communauté P2P mais surtout de savoir où se situe une ressource particulière.

2.6 Conclusion

Dans ce chapitre, nous avons présenté les réseaux Pair-à-Pair. Différents niveaux de décentralisation permettent de scinder ce modèle de réseaux en trois sous-modèles qui sont le modèle P2P pur, hybride et centralisé. Le modèle pur est constitué de pairs strictement équivalents. Le modèle hybride utilise des super-pairs qui présentent des fonctions avancées. Enfin le modèle centralisé repose sur un serveur dédié qui assure les fonctions de découverte et localisation.

Après, nous avons vu aussi que ces réseaux peuvent être classifiés en réseaux non structurés ou structurés. Ces dernier utilisent des protocoles basés sur les DHT, come le protocole Chord. L'utilisation d'un réseau structuré ou non dépend finalement de l'application recherchée. Il est donc souhaitable de s'intéresser à l'implémentation de mécanismes de routage dans ces deux types de réseaux.

Ensuite, nous avons présenté Skype, le premier protocole de la VoIP basé sur la technologie P2P, trois facteurs essentiels le rend populaire :

1. Il fournit une bonne qualité de voix.
2. Il peut fonctionner pratiquement derrière tous les NATs et les firwalls.
3. Il est facile à installer et à utiliser.

Malgré la croissance de la popularité de Skype, il présente quelques inconvénients, par exemple la centralisation de l'opération de login (*skype.com*), car si ce serveur tombe en panne, alors tout le réseau Skype devient infonctionnel.

Enfin, nous avons abordé des problématiques de sécurité. Actuellement, la sécurité des réseaux Pair-à-Pair s'oriente principalement autour de l'anonymat et de la gestion de groupes fermés. Certains travaux ont également été publiés concernant la disponibilité et l'authentification dans des réseaux structurés.

Dans la suite de ce rapport, nous nous intéressons au travaux existant sur SIP-P2P, ainsi qu'à la présentation de notre solution d'optimisation du routage SIP, basée sur le protocole Chord.

Chapitre 3

Optimisation du routage de la signalisation SIP

3.1 Introduction

Actuellement, les tables de hachage distribuées sont intégrées dans de nombreuses applications P2P. Elles sont en effet parfaitement adaptées pour assurer la localisation et l'accès à des ressources dans un environnement dynamique et distribué. Dans le chapitre précédent, nous avons présenté quelques études comparatives de DHTs qui identifient leurs différences.

Notre contribution n'a pas pour objectif de comparer les différentes propositions de DHTs, mais plutôt de proposer une amélioration d'un protocole existant basé sur les DHTs, orienté vers l'optimisation du routage de la signalisation SIP. Le protocole que nous avons choisi est Chord. Beaucoup de travaux ont été basé sur sur ce protocole mais présentent quelques inconvénients. Par la suite, nous allons présenté le premier travail de K. Singh and H. Schulzrinne décrit dans [SSc05] et baptisé SIP-P2P, comme référence des travaux existants sur la téléphonie Internet basée sur SIP en mode P2P. Le deuxième travail, que nous allons présenté est celui de David A. Bryan *et al.*, qui proposent dans [4] une approche entièrement Pair-à-Pair de SIP, baptisée SOSIMPLE.

3.2 La téléphonie IP basée sur le modèle client-seveur

A la différence du P2P, la téléphonie existante basée sur SIP a une architecture client/ serveur [34]. Comme il est montré sur la figure 3.1. Quand un utilisateur Bob lance l'application SIP client sur son ordinateur ou sur son téléphone IP ou sur un autre appareil qui supporte la téléphonie IP, il s'enregistre dans le serveur SIP en indiquant l'adresse IP de son

appareil. Le serveur enregistre l'identificateur de l'utilisateur et son adresse IP pour pouvoir faire la correspondance. Quand un autre utilisateur Alice appel Bob, il envoie son appel au serveur, ce dernier s'occupe de l'acheminement de l'appel vers la bonne destination.

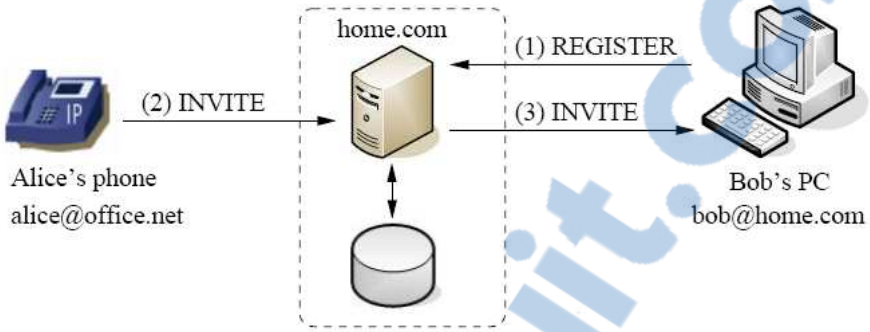


FIG. 3.1 – Établissement d'appel SIP en utilisant des serveurs proxy

Dans cet exemple, il est clair qu'un serveur simple peut devenir un goulot d'étranglement ou une cause de défaillance qui empêche le fonctionnement de système. Il peut être amélioré en ayant des multiples serveurs redondants. Deux solutions ont été proposées dans [34].

1. La première solution illustrée par la *figure 3.2.(a)* consiste à dupliquer les serveurs et leurs contenus. Lors de la première phase, les utilisateurs s'enregistrent dans tous les serveurs. Quand un autre utilisateur veut les contacter, il envoie sa requête à l'un des serveurs. L'inconvénient de cette solution est que l'enregistrement et le départ des nœuds ne se font pas en temps réels. Par consequent, si par exemple un utilisateur A quitte le système avant que les serveurs mettent à jour cette information, il se peut qu'un autre utilisateur B contacte l'utilisateur A alors que ce dernier a déjà quitté le système et donc il y a une information inconsistante.

2. La deuxième solution décrite sur la *figure 3.2.(b)* consiste à dupliquer les serveurs et

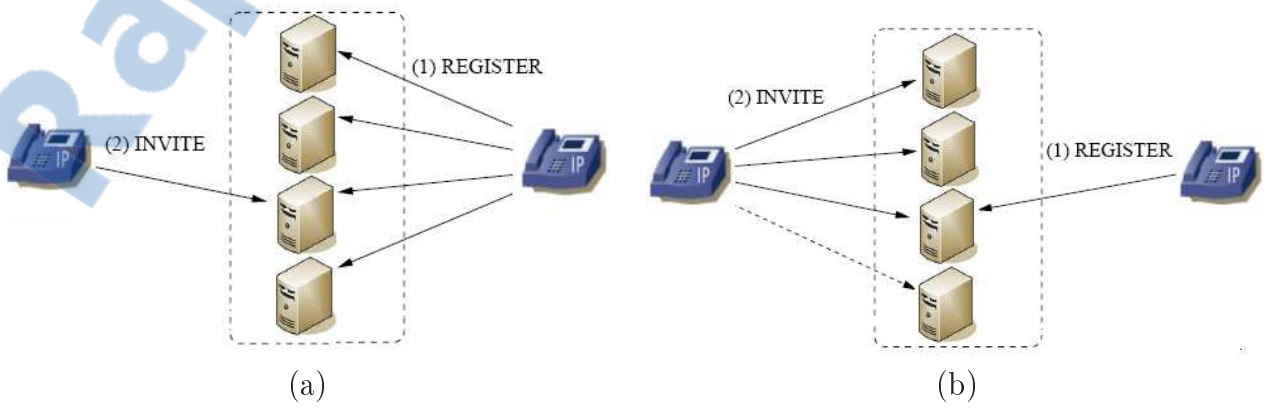


FIG. 3.2 – La recherche et l'enregistrement des utilisateurs

leurs contenus. Lors de la première phase, les utilisateurs s'enregistrent dans l'un des serveurs. Quand un autre utilisateur veut les contacter, il envoie sa requête à tous les serveurs. Celui dans lequel l'interlocuteur est enregistré s'occupe de l'acheminement des requêtes. L'inconvénient de cette solution est que le nombre de messages pour chaque nouvelle recherche est élevé, ainsi que l'enregistrement dans un seul serveur ne garantit pas que les utilisateurs restent connectés au réseau en cas de défaillance de serveur sur lequel il sont connectés.

La téléphonie IP basée sur SIP peut être traitée comme un système P2P avec un ensemble statique de super-nœuds (serveurs SIP) où la recherche est basée sur le DNS au lieu d'une clef de hachage. Cependant, employer une architecture P2P pure au lieu d'un ensemble statique de serveurs SIP améliore la fiabilité et permet au système de s'adapter dynamiquement aux défaillances de nœud.

3.3 La téléphonie IP basée sur une architecture P2P

La téléphonie Internet en mode P2P en utilisant le protocole d'initiation de session (P2P-SIP) [34, 4, 35] a été proposé pour minimiser le coût de maintenance et de configuration des serveurs dans l'architecture SIP, ainsi pour éviter les pannes catastrophiques de ces serveurs. Beaucoup de Solution sont en cours de développement sur ce sujet. Parmi ces solutions, deux approches ont été proposés par K.Singh dans [35] pour combiner SIP et P2P : remplacez le serveur de localisation SIP par un protocole P2P (SIP-utilisant-P2P) [34], ou bien, implémenter le protocole P2P lui-même en utilisant l'échange de messages SIP (P2P au-dessus de SIP).

3.3.1 P2P-SIP

Dans [34], H.Schulzrinne and K.Singh ont proposé une architecture pour les systèmes de téléphonie IP basé sur SIP. cette architecture P2P-SIP supporte l'enregistrement des utilisateurs et l'établissement des appels, ainsi que des services avancés tel que la délivrance des messages off-line, les messages voix/vidéo et les conférences multipartie. Cette solution est détaillée dans [35], où l'auteur propose une solution de téléphonie IP basée sur une architecture P2P selon deux configurations possibles. Dans la première configuration présentée par la figure 3.3(a), les super nœuds forment le réseau P2P, alors que les nœuds ordinaires utilisent ce réseau à travers le super nœud auquel ils sont attachés. Dans cette architecture, chaque

nœud peut être un super nœud ou un nœud ordinaire suivant ses capacités. Le coût de la recherche dans les deux types d'architecture est $O(\log n)$. La première solution est coûteuse en maintenance de stabilité s'il y a fréquemment des nouveaux super nœuds qui arrivent ou quittent. Alors que la deuxième solution présente un problème pour les nœuds de capacités limités.

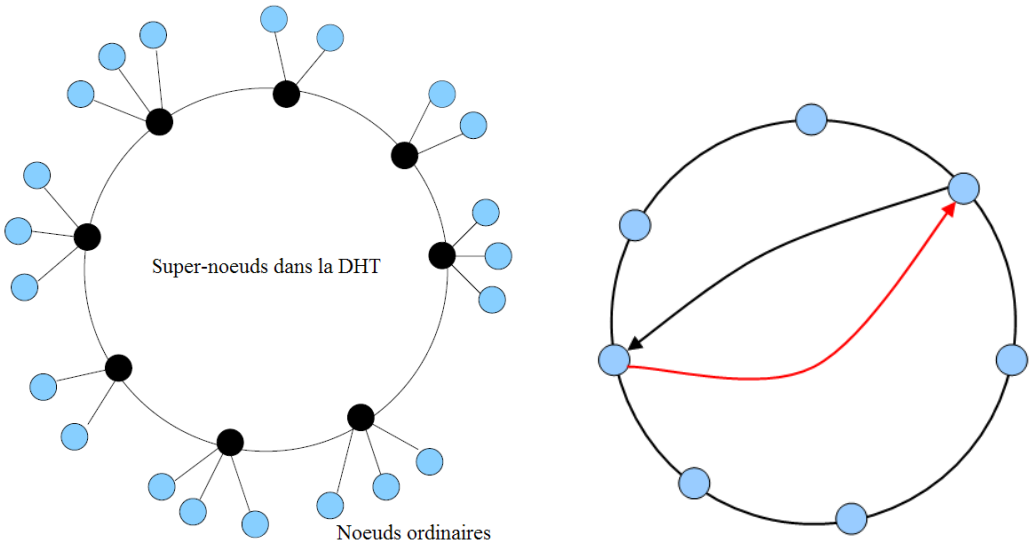


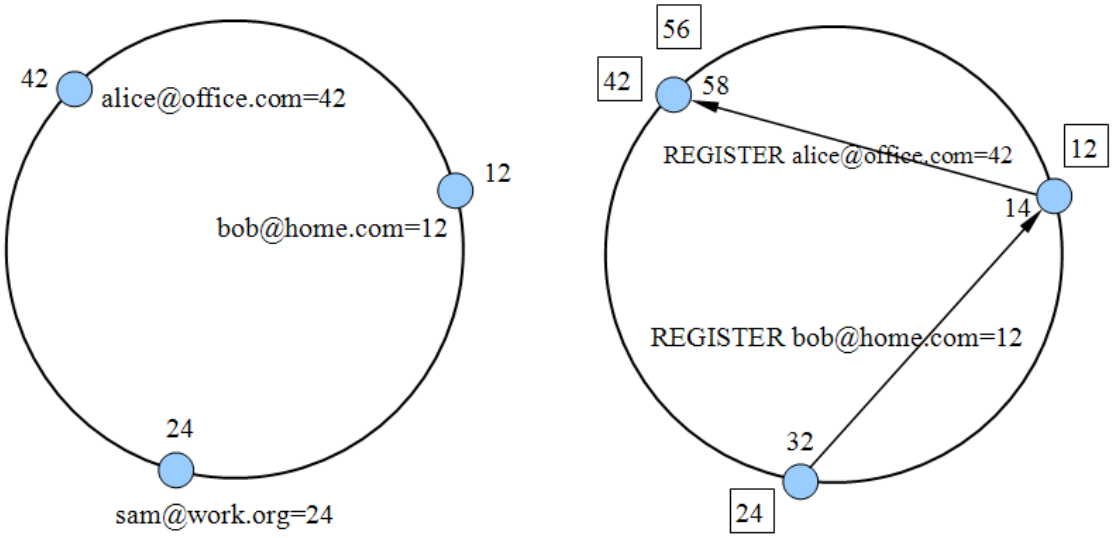
FIG. 3.3 – La téléphonie IP en mode P2P.

Dans la deuxième architecture (figure 3.3(b)) le problème qui se présente est que quelques nœuds ont par exemple un faible CPU, une petite mémoire, une petite bande passante ou ils sont derrière un NAT ou un Firewall, donc ils ne peuvent pas accomplir leurs fonctions dans le réseau P2P. Ce problème peut être résolu en adoptant un réseau P2P hybride, avec des super nœuds. Les nœuds qui ont une grande capacité (*bande passante, mémoire, CPU, ainsi que des adresses IP publiques*), constituent des super nœuds. Un nœud ordinaire est connecté à un des super nœuds. La décision de devenir un super nœud ou un nœud ordinaire est locale. Quand un nouveau nœud rejoint le réseau, il devient un nœud ordinaire. Quand il détecte qu'il a des capacités lui permettant d'être un super nœud, il peut transiter vers un super nœud.

3.3.1.1 L'enregistrement d'un nouveau utilisateur

A l'arrivée d'un nouveau utilisateur (*nœud*), il indique son nom (*alice@office.com*). En utilisant les tables de hachage distribuées, il obtient une clé à partir non pas de son adresse

IP, mais à partir de son nom, ensuite il se place dans le réseau P2P (*La clé de Alice est 42 dans la figure 3.4.(a)*). Quand un autre utilisateur qui possède l'adresse de *Alice* veut parler



(a) (b)

FIG. 3.4 – La recherche et l'enregistrement des utilisateurs

avec lui (*Bob qui a la clé 12*), il calcule sa clé par la même fonction de hachage en utilisant son nom et il lance la requête *rechercher(clé 42)*. Après établissement de la connexion, les deux utilisateurs peuvent parler entre eux. Cette architecture ne contient pas de registration et par conséquent elle ne permet pas la messagerie off-line (*si Alice n'est pas présente alors Bob ne peut pas lui envoyer des messages*). Dans la deuxième architecture(figure 3.4.(b)) la clé du nœud et celle de l'utilisateur sont calculées séparément. Un message SIP REGISTER est utilisé à la fois pour l'insertion d'un nouveau nœud dans le réseau, ainsi que pour la registration de ce dernier dans le même réseau.

Quand *Alice* lance son application client, le nœud utilise son adresse IP pour calculer sa clé, par exemple 14 et il prend place dans le réseau, ensuite, il calcule la clé d'utilisateur en utilisant le nom d'utilisateur et publie cette clé par un message REGISTER de clé 42. Le nœud 58 qui est responsable de la clé 42 accepte l'enregistrement et maintient dans sa table de routage que la clé 58 se trouve dans le nœud 42. Si *Bob* n'est pas présent alors *Alice* peut lui envoyer des messages off-line avec la clé 56, qui vont être délivrés à *Bob* quand il devient on-line.

3.3.1.2 Architecture de P2P-SIP

La figure 3.5 montre un diagramme d'un nœud P2P-SIP, il est composé de différents blocs lui permettant de s'enregistrer dans le réseau P2P, de localiser d'autres utilisateurs, de transmettre et de recevoir des messages off-line et de détecter les Firewalls et les NATs derrière lesquels il est, ...etc.

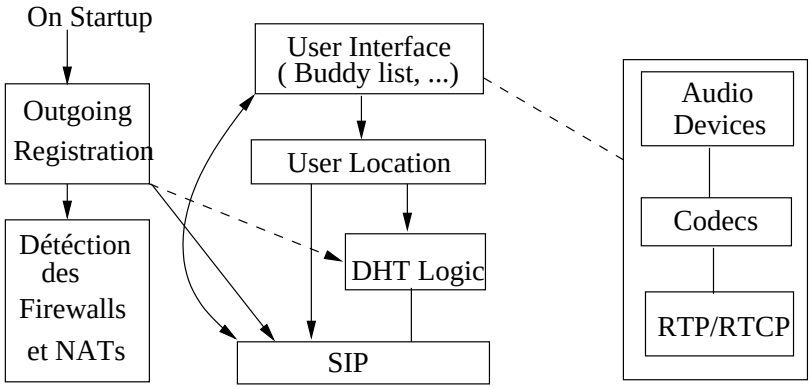


FIG. 3.5 – Diagramme d'un nœud P2P-SIP

3.3.1.3 L'enregistrement d'utilisateur

Pour qu'un nœud rejoint le réseau, il doit s'enregistrer par son adresse IP pour pouvoir être localisé dans le réseau P2P et par son nom pour qu'il puisse transmettre et recevoir des messages off-line. La figure 3.6 montre comment un nœud s'enregistre dans le réseau.

3.3.1.4 Défaillance d'un nœud

Pour qu'un nœud ordinaire quitte le système (figure 3.7), il n'a qu'à envoyer un message REGISTER au super nœud avec lequel il est attaché, ce dernier fait propager ce message vers le super nœud qui est responsable de ce nœud (*qui stocke sa clé*). La défaillance d'un nœud ordinaire n'influe pas sur le reste du système. Dans ce cas, le super nœud sur lequel le nœud défaillant était attaché peut détecter la défaillance de ce dernier par l'absence de réponses sur les messages de rafraîchissement périodique, il peut le confirmer par l'envoi d'un message OPTIONS au nœud défaillant, s'il ne reçoit pas de réponse, le nœud est défaillant.

Quand un super nœud quitte le système, ce dernier doit faire quelques mises à jour qui concernent les nœuds ordinaires qui ont été attachés à ce super nœud, ainsi que ses voisins.

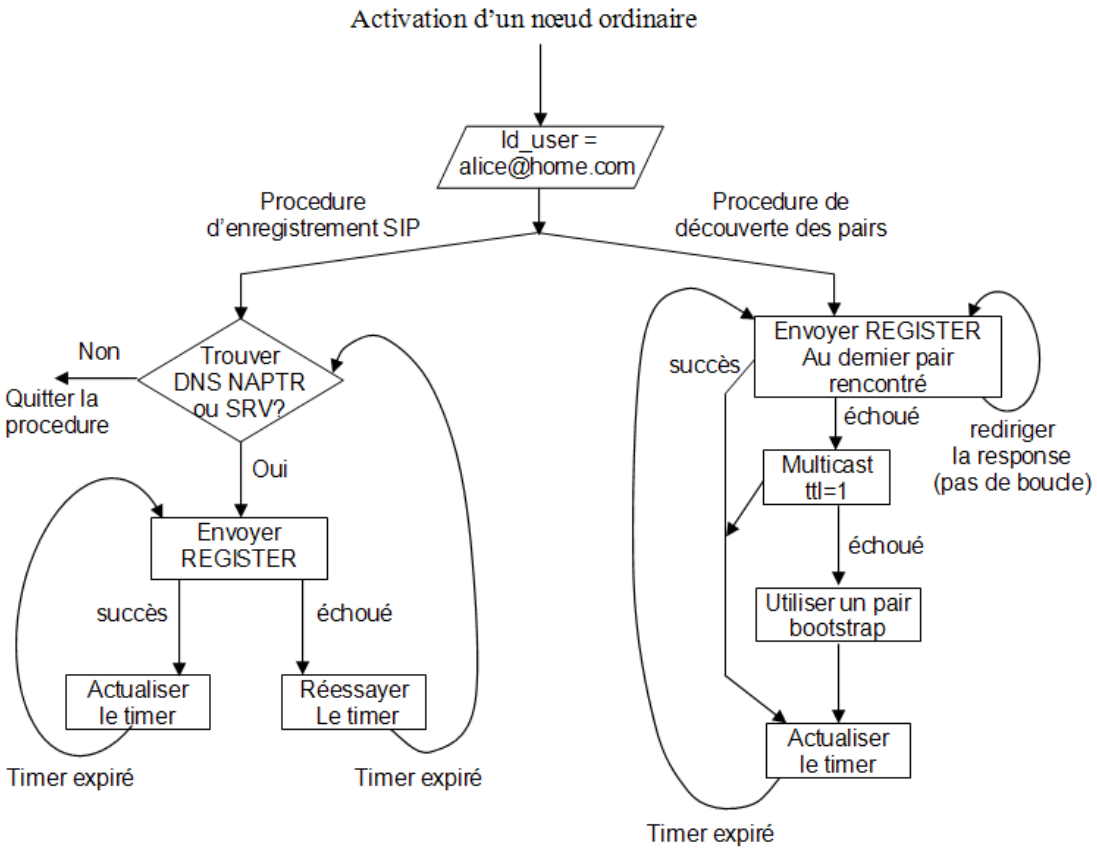


FIG. 3.6 – Node startup and outgoing registration

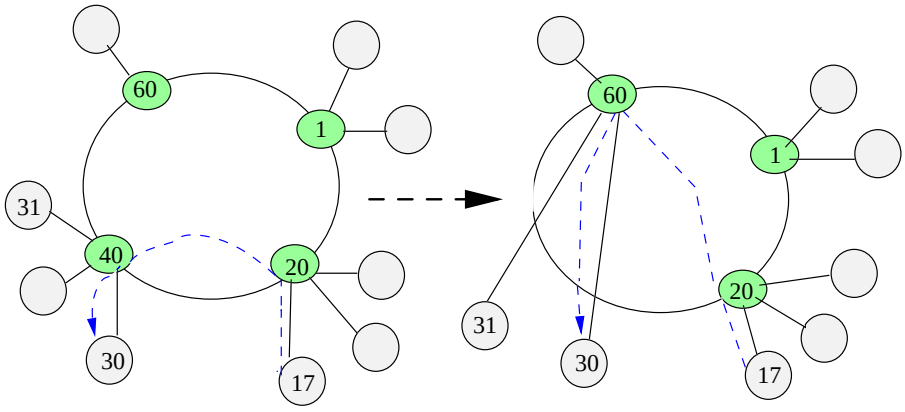


FIG. 3.7 – Défaillance d'un super nœud dans le DHT

Dans la défaillance d'un super nœud, les nœuds ordinaires qui ont été attachés à ce dernier seront ré-attachés à un autre super nœud. La défaillance d'un super nœud est détectée par leurs voisins.

3.3.2 SoSimple

Dans [4], David A. Bryan *et al.* proposent une approche entièrement Pair-à-Pair de SIP, baptisée SOSIMPLE et depuis projet P2PSIP de l'IETF. L'idée est de mettre en relation les participants via un réseau Pair-à-Pair et non via des serveurs. SOSIMPLE utilise pour cela un réseau structuré. Lors de son insertion, un nœud A insère une ressource à l'emplacement *h(nameA)* de la DHT, contenant son IP réelle. Lorsque B veut contacter A, il effectue une recherche dans la DHT de *h(nameA)*, obtient l'adresse IP de A et initie la connexion avec A. La localisation du destinataire est réalisée grâce à la DHT et non plus grâce à un serveur.

3.3.2.1 Arrivée d'un nœud

Quand un nouveau nœud souhaite joindre le réseau, il doit d'abord localiser un nœud déjà dans le reseau, qui lui sert de bootstrap. Actuellement ce nœud est localisé via des mécanismes hors-bande. Le nœud se joignant calcule son ID-Nœud, dans notre exemple 503 (voir figure 3.8), et l'envoie dans un message REGISTER au nœud bootstrap, qui a l'ID-Nœud égale à 023 (1). Supposant que le bootstrap n'est pas le nœud actuellement responsable de ce ID-Nœud, il répond avec les informations du nœuds le plus près où le nœud se joignant sera placé dans le réseau, dans notre exemple, c'est le nœud B, avec ID-Nœud égale à 445. Cette information est passée en en-têtes dans une réponse SIP 302 (Moved Temporarily) (2). Le nœud se joignant répète le processus, en utilisant ce nœud plus proche comme nouveau bootstrap (3-4).

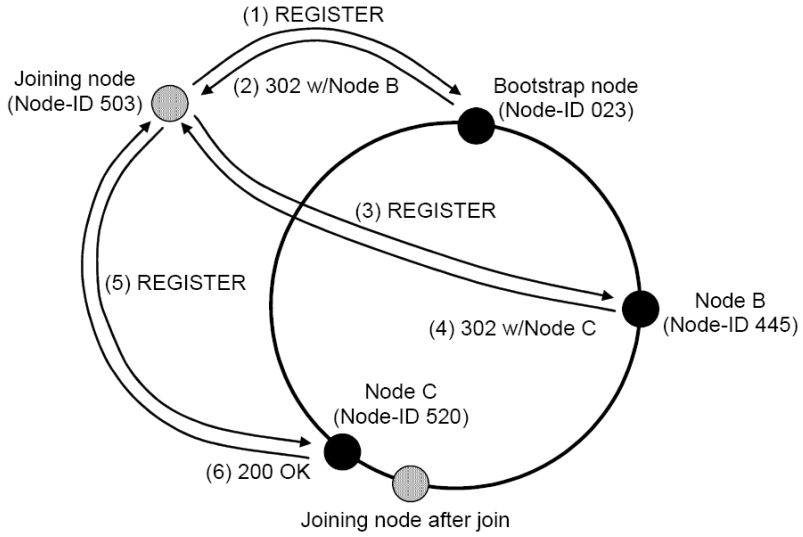


FIG. 3.8 – Extrait de [4]. Un exemple d'un nouveau nœud avec ID-nœud=503 joignant le réseau.

Enfin, le nœud se joignant atteint le nœud qui est actuellement responsable de son ID dans le réseau, dans ce cas-ci le nœud C, avec ID-Nœud 520. Le nœud C répond avec une réponse SIP 200 (OK) comprenant des informations détaillées dans l'en-tête du message sur les voisins (5-6), permettant au nœud se joignant de s'insérer lui-même dans le reseau.

3.3.2.2 Localisation d'un nœud

Quand un nœud souhaite trouver le nœud responsable d'un utilisateur particulier il commence par le hachage du nom d'utilisateur pour produire un ID-Ressource. Puisque le nœud de l'utilisateur est déjà placé correctement dans le réseau (l'enregistrement du nœud est déjà établi), le nœud a un certain nombre d'entrées dans la table des raccourcis qui pointe vers les nœuds dans le réseau. Dans l'exemple présenté par la figure 3.9, Alice lance la rechercher d'une ressource (Bob dans l'exemple). Le nœud de Alice cherche dans sa table de raccourcis le nœud qui a un ID-Nœud plus proche de ID-Ressource à chercher (dans ce cas le nœud A). Le nœud d'Alice envoie un message au nœud A (1). Le nœud A n'est pas responsable de cet ID-Ressource, ainsi il envoie une réponse SIP 302 (Moved Temporarily), y compris le nœud qu'il pense la plus proche, le nœud B, dans l'en-têtes (2). Le nœud d'Alice essaye le nœud B et reçoit encore une réponse avec un autre nœud, dans ce cas le nœud C (3-4). Enfin, le nœud d'Alice essaye le nœud C, qui est responsable de cette ressource (5).

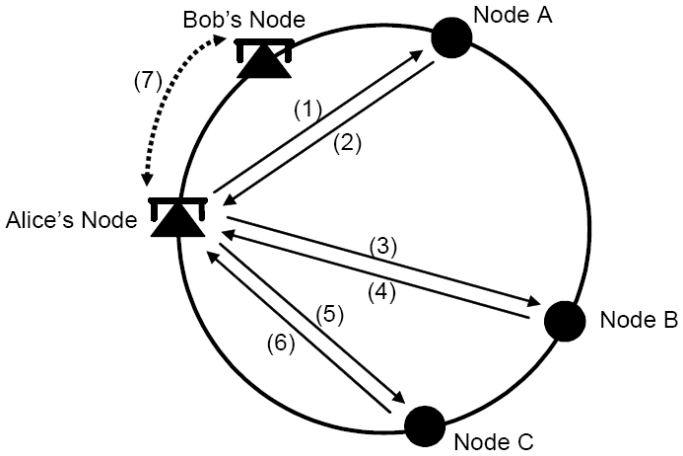


FIG. 3.9 – Extrait de [4]. Alice localise l'utilisateur Bob et établit une session de communication.

Une fois Alice a l'adresse IP de l'UA de Bob, un appel entre ces UAs peut être établi directement entre les nœuds en utilisant les mécanismes conventionnels de SIP sans impliquer le réseau (7). Le nœud d'Alice cache l'information reçue du nœud C et peut les employer pour des futures communications avec Bob. Une fois que cette information de localisation

est passée vers le UA, l'établissement d'appel SIP n'exige aucune fonction du P2P. Comme dans le SIP traditionnel, les médias coule également directement entre les points finaux.

3.3.3 Problèmes soulevés des solutions existante

La première solution P2P-SIP de H.Schulzrinne and K.Singh est basée sur une architecture P2P hybride, là où il existe toujours une notion de client (nœuds ordinaire) et serveur (super nœuds). Les auteurs dans leurs solution, ont traité le cas de défaillance des nœuds ordinaires, mais pour les super-nœuds, il n'ont pas fait. Cependant, ce point est un problème qui s'est posé par cette solution. Ainsi, cette solution reste exposé aux problèmes du modèle client-serveur, tel que la défaillance des serveurs, qui sont des éléments critiques dans ce modèle.

La deuxième solution proposé par David A. Bryan *et al.* dans [4], est basée sur un routage *itérative* [9] : le nœud initiateur d'une requête contacte, d'après sa table de routage, un nœud plus proche de la clé à atteindre. Le nœud contacté consulte alors sa table de routage et retourne au nœud initateur l'identifiant d'un nœud plus proche. Le nœud initiateur contacte alors ce dernier nœud, et ainsi de suite. L'exemple donné par la figure 3.9 illustre le cas d'un nœud qui désire accéder à une ressource représentée par une clé. On voit que d'après la méthode itérative, le nœud initiateur va contacter chacun des nœuds sur le chemin menant à la clé recherché et attend une réponse de chacun d'eux pour connaître l'identifiant du prochain saut. Cette méthode est assez coûteuse en termes de nombre de messages à échanger, et donc pas optimale pour le routage.

3.4 Proposition

Vue que le routage dans les réseaux P2P est un routage au niveau applicatif et par consequent, le nombre des sauts correspondant dans la couche réseau sera plus grand. L'objectif dans le routage P2P est de diminuer au maximum possible le nombre de sauts.

Étant donné les solutions exposées dans la section 3.3, nous nous sommes intéressés à la manière dont il serait possible d'optimiser le nombre des messages échanger entre les nœuds dans la DHT. Pour cela, nous allons défini des unités de travail qui représentent les opérations possible implantées par toutes les DHTs actuelles. Parmi ces opérations, nous nous sommes particulièrement intéressés à celle qui est relative au routage de la signalisation SIP, ainsi qu'à

la localisation des ressources. Nous avons choisi Chord car c'est une proposition reconnue et standard dans la communauté des DHTs. En outre, la simplicité de ses concepts en font un bon choix dans le cadre de notre sujet de téléphonie IP basée sur SIP-P2P.

3.4.1 Définition des unités de travail

Les DHT reposent sur des modèles topologiques différents et sont constituées de services, dont le fonctionnement diffère d'une infrastructure à une autre. Néanmoins, d'une manière générale, une DHT présente toujours quatre processus, qui sont :

- **Un processus de localisation :** qui assure la localisation des ressources à travers le routage des requêtes vers le nœud racine de leur identifiant ;
- **Un processus de maintenance :** qui met à jour les données inscrites dans les tables de routage des pairs pour assurer la cohérence de la DHT ;
- **Un processus d'insertion pour les pairs et les clés :** qui permet l'insertion d'un nœud ou d'une clé par la recherche du nœud racine ;
- **Un processus de retrait pour les pairs et les clés :** qui permet d'informer les nœuds impliqués du retrait d'un nœud ou d'une clé.

Ces quatre processus sont déployés par toutes les propositions de DHTs actuelles, mais leur fonctionnement varie d'une implantation à une autre. Dans le cadre de notre sujet d'optimisation du routage de la signalisation SIP au dessus d'une DHT, nous avons pris en compte ces quatre processus.

Dans le cadre de SIP, le processus d'enregistrement des utilisateurs est effectivement semblable au processus d'insertion d'un pair. Par la suite, nous décrivons la manière dont nous allons modéliser l'exécution des deux processus de localisation et d'insertion qui sont les processus auxquels nous nous sommes particulièrement intéressés.

3.4.2 Processus de localisation et de routage

Le processus de localisation et de routage est le processus qui représente la finalité d'une DHT. Afin de bien expliquer notre solution, nous introduisons deux termes qui caractérisent le rôle d'un nœud dans le processus de localisation. Nous appelons nœud *source*, un nœud qui initie une requête de localisation de ressource, et nœud *routeur*, un nœud qui participe au routage d'une requête initiée par un nœud source.

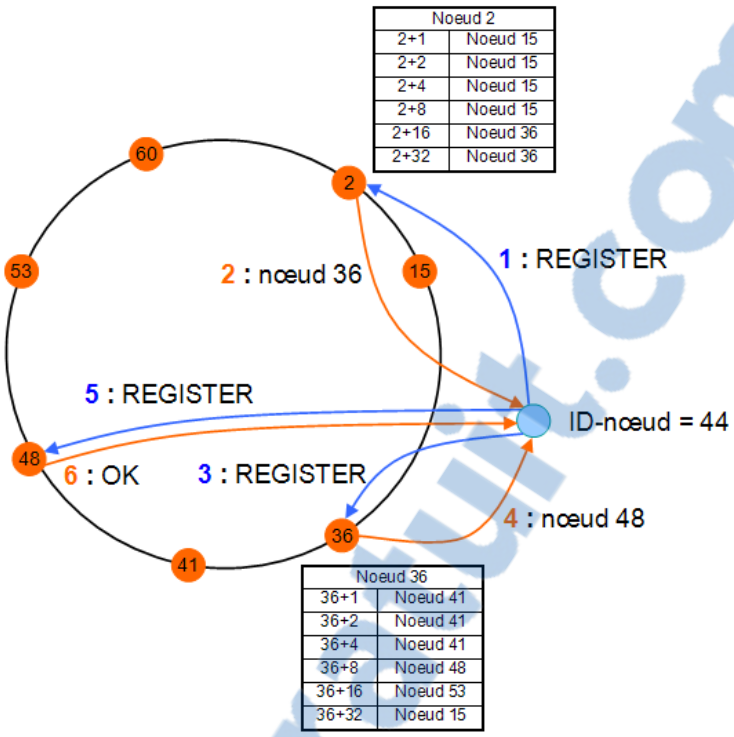


FIG. 3.10 – Enregistrement d'un nouveau utilisateur avec routage itérative (approche SO-SIMPLE).

Le processus de localisation et de routage d'une requête vers une clé donnée s'effectue par approches successives : un nœud source ou routeur d'une requête tente toujours de trouver un nœud plus proche que lui de la clé à atteindre. Pour effectuer cette approche, nous allons choisi une méthode *récursive* [9] pour le routage d'une requête au lieu d'une méthode itérative, méthode coûteuse en nombre de messages échangés (voir l'exemple de la figure 3.10). Cette méthode récursive fait passer la requête initiée par un nœud source de nœud routeur en nœud routeur jusqu'à atteindre la clé requise. Une fois la clé est trouvée le nœud responsable de celle ci va répondre au nœud source de la requête.

3.4.3 Enregistrement d'un nouveau nœud

A l'arrivée d'un nouveau nœud, un message REGISTER est envoyé par ce dernier vers un nœud sur l'anneau. Ce premier nœud routeur est choisi aléatoirement. Nous illustrons l'arrivé d'un nœud par l'exemple de la figure 3.11.

Dans notre premier exemple (figure 3.11), lorsqu'une requête REGISTER pour la clé 44 est reçue par le nœud 2, celui-ci va tout d'abord vérifier dans son cache local s'il n'est pas la racine de cette clé. Si oui, il retourne un pointeur vers la ressource au nœud source.

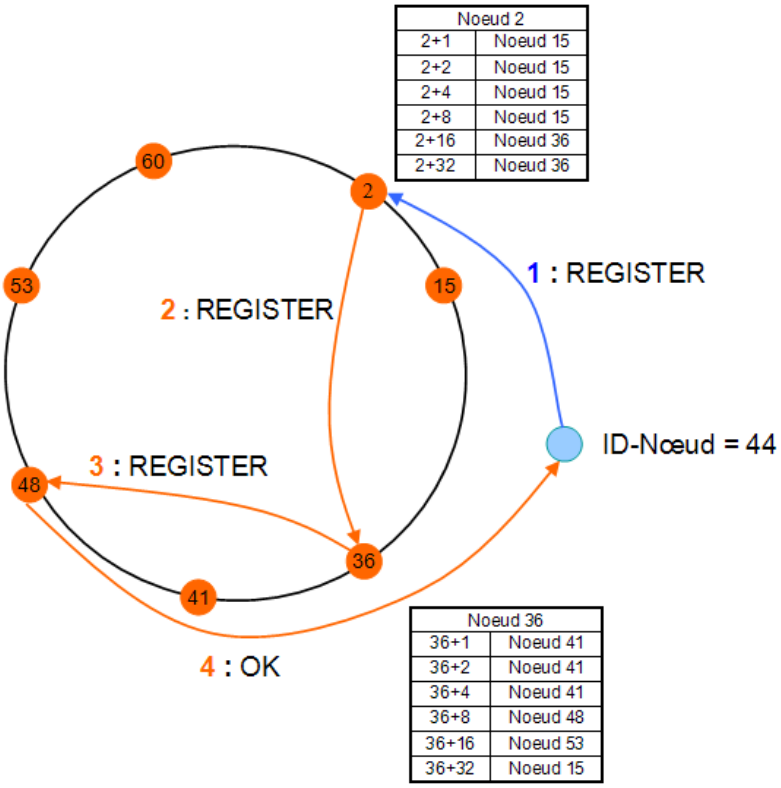


FIG. 3.11 – Enregistrement d'un nœud avec nombre de message réduit (approche récursive).

Sinon, le nœud consulte sa table de raccourcis pour trouver un nœud plus proche de la clé requise. Si aucun nœud n'est trouvé, une réponse d'échec est fournie au nœud source. Dans notre exemple, la requête est transférée au nœud 36,ensuite au nœud 48. Ce dernier est la racine de la clé 44 et il va donc répondre au nœud source par un message 0K. Notons que le nœud source va attendre la réponse à la requête qu'il a soumise, par contre le nœud routeur va simplement faire suivre la requête sans attendre de réponse. Pour finir, le nœud source termine le processus de localisation lorsqu'une réponse positive ou négative est reçue. Bien que dans le cadre d'une implantation, un *timeout* est mis en place pour éviter une attente infinie.

3.4.4 Établissement d'une session

Lorsque le nœud d'un nouveau utilisateur est bien placé sur l'anneau (après l'enregistrement avec le message REGISTER), il peut établir une session avec les autres utilisateurs déjà enregistrés et placés sur le réseau. L'établissement d'une session et réalisé par l'envoi du message de signalisation INVITE à un utilisateur choisi aléatoirement, comme illustré par la figure 3.12(a). Dans notre exemple, le nœud d'Alice et placé sur l'anneau, Alice veut

communiquer avec l'utilisateur Bob. Elle va tout d'abord haché l'adresse de Bob pour obtenir son identificateur, et elle va lancer la recherche de Bob à partir de son identificateur (ID-User de Bob = 1).

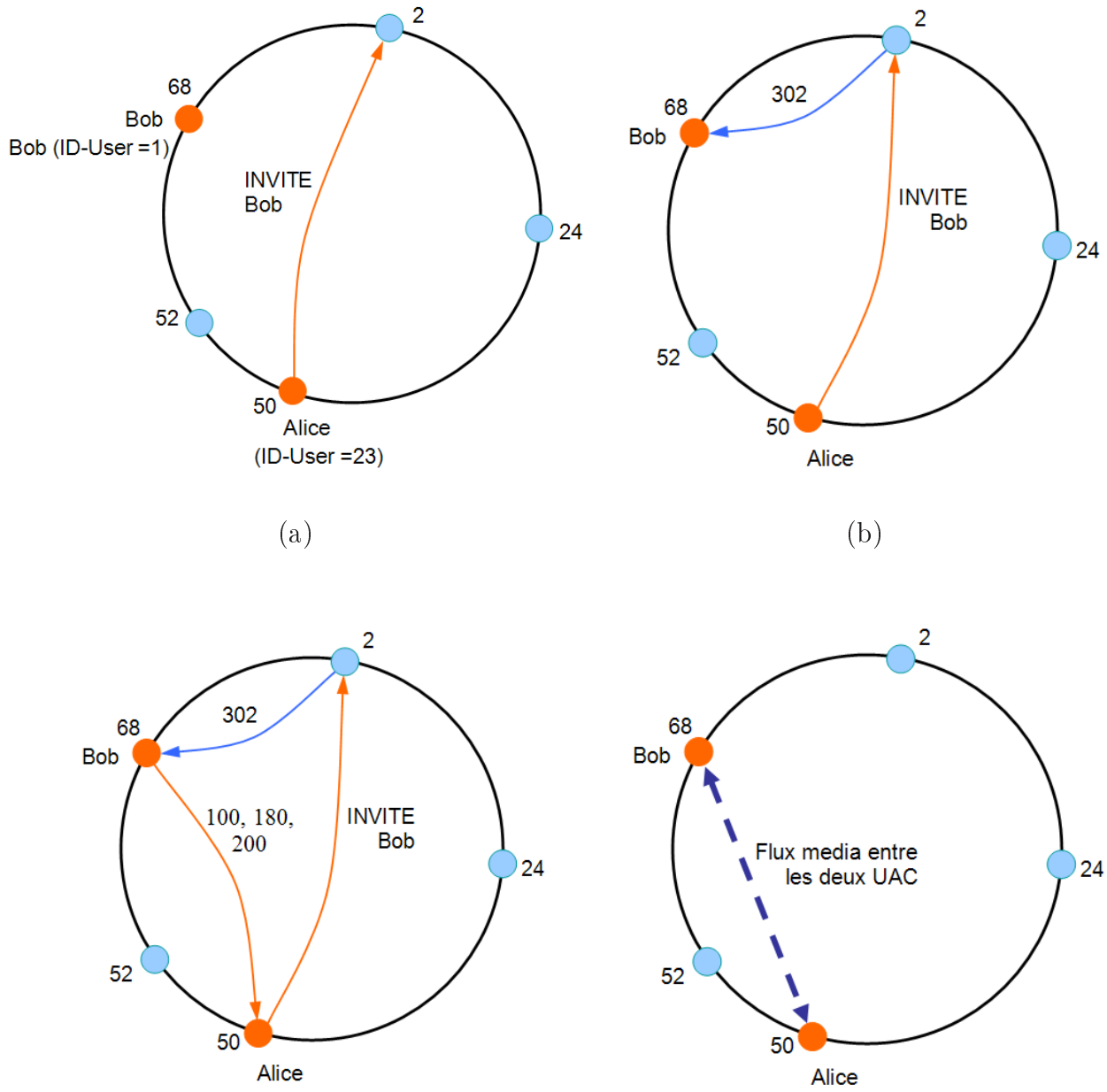


FIG. 3.12 – Établissement d'une session entre Alice et Bob, en réduisant le nombre de messages échangés

Puisque notre solution est basé sur une approche récursive, donc lorsque le nœud d'identifiant 2 reçoit le message INVITE, et aprés qu'il trouve dans sa table de raccourci que le nœud d'identifiant 68 est la racine de la clé 1 de l'utilisateur Bob, il va directement renvoyé le message avec le code 302 (Moved temporarily) au nœud d'identifiant 68 (figure 3.12 (b)).

L'UA de bob reçoit donc le message d'invite et répond directement l'UA d'Alice par des messages avec les codes 100, 180, 200 (Trying, Ringing et OK, respectivement).

Bob a accepté donc, de communiquer avec Alice (figure 3.12 (c)), et les flux média sont échangés tout au long de la communication, en directe entre le nœud d'Alice et celui de Bob sans passer par les autres nœuds du réseau (figure 3.12 (d)).

3.4.5 Définition des variables

Les variables que nous avons définies sont les suivantes :

Nb_{MAX} : la plus grande valeur numérique correspondante à un identifiant pour les pairs et les clés. La fonction de hachage de Chord génère des identifiants compris dans l'intervalle [0,Nb_{MAX}[et toutes les opérations sont effectuées *modulo* Nb_{MAX} ;

n_i : un nœud d'identifiant i ;

N : l'ensemble des nœuds n_i qui sont actuellement présents et joignables dans un anneau, avec $N = \{n_i\}$;

N_T : le nombre total de nœuds dans un anneau, avec N_T = *Card(N)* ;

k_i : une clé d'identifiant i ;

K_i : le nombre de clés dont le pair N_i est responsable, avec $K_i = Card(\{k_j\})$ et $id(pred(n_i)) \leq j \leq i$;

K_T : le nombre total de clés référencées dans un anneau, avec $K_T = \sum_{i \in N} K_i$;

$\overline{K}$: le nombre moyen de clés par pair, avec $\overline{K} = \frac{K_T}{N_T}$.

3.4.6 Propriétés de la solution

En plus des buts explicites de notre contribution, il y a quelques avantages implicites de scalability et de fiabilité dans notre solution de P2P/SIP comparé à l'architecture SIP Client/Serveur. Effectivement, les propriétés de notre solution peuvent être résumé dans les points qui suit.

3.4.6.1 Interopérabilité

Notre proposition de téléphonie P2P est basé sur le protocole SIP. nous avons choisis SIP comme protocole de signalisation pour assurer l'interopérabilité. Cette dernière, devrait être facilement intégrer avec les protocoles existants et l'infrastructure de la téléphonie IP.

3.4.6.2 Optimisation du nombre de messages échangés

Dans l'exemple de la figure 3.10, qui présente la solution SOSIMPLE, le nombre de message échangés lors du routage de la requête d'enregistrement (REGISTER) est égale à 6. Par contre dans notre proposition illustrée par la figure 3.11, il est égale à 4. Donc, nous avons réduit le nombre de message de cet exemple de 2 message. Étant donné un réseau qui contient un grand nombre de nœuds et supposant que la requête va traversé N nœuds routeur, dans ce cas là notre proposition réduit le nombre de message de *N-1* messages, par rapport à la méthode itérative.

3.4.6.3 Décentralisation

Notre solution basée sur le protocole Chord, est purement distribuée, donc aucun nœud n'est important que les autres. Cependant, le système devrait être capable de se configurer automatiquement, par exemple, en détectant les NAT et les paramètres des firewall, en découvrant des pairs voisins et en effectuant l'enregistrement initial.

3.4.6.4 Recherche efficace

Une recherche aveugle basée sur l'inondation est inefficace. Notre solution était basée sur un DHT fondamental pour optimiser la recherche. Nous avons choisis Chord comme DHT fondamental pour notre système en raison de sa robustesse et efficacité dans le cas où le nœud concurrent se joint et part.

3.4.6.5 Équilibre des clés

Contrairement au modèle client-serveur qui centralise ses ressources, le modèle P2P les distribue parmi l'ensemble des pairs participants. Les DHTs reposent sur des modèles qui appuient leurs propriétés de performance de localisation sur une répartition équilibrée des clés. La fonction de hachage qui génère les identifiants de nœuds et de clés garantit avec une très forte probabilité que les clés seront réparties de manière équitable parmi les nœuds présents. Néanmoins, dans le cas d'un déploiement réel d'une DHT, mettant en oeuvre des pairs et des clés qui vont et viennent de manière imprévisible, il est impossible de savoir si cet équilibre sera toujours assuré. Or, un déséquilibre de cette répartition peut fortement dégrader les performances du processus de localisation. A l'extrême, si seuls quelques pairs devenaient responsables de la majorité des clés, la DHT ne fonctionnerait plus selon le modèle P2P mais selon le modèle client/serveur, et elle ne pourrait plus garantir aucune propriété en termes de passage à l'échelle, de tolérance aux fautes et d'équilibre de la charge et du trafic.

3.5 Conclusion

Plusieurs travaux ont été réalisés autour la téléphonie IP en mode client-serveur, mais comme on a vue dans le deuxième chapitre, le réseau Internet est en changement permanent de l'architecture client-serveur vers le mode P2P. Pour cela, les recherches en téléphonie IP commencent à se diriger vers le mode P2P, peu de travaux qui ont été faits dans ce domaine.

Le routage dans les réseaux P2P est un routage au niveau applicatif et par consequent, le nombre des sauts correspondant dans la couche réseau sera plus grand. L'objectif dans le routage P2P est de diminuer au maximum possible le nombre de sauts.

Nous avons proposés dans ce domaine, une solution intéressante, en fait c'est une amélioration des solutions existantes de *la téléphonie IP en mode P2P en utilisant le protocole de signalisation SIP*, l'avantage principale de notre solution est qu'elle optimise le nombre de messages de signalisation échangés lors d'une communication.

Notre solution est basée sur une architecture P2P pure, ainsi sur le protocole de signalisation SIP, donc elle rassemble et combine les avantages des architectures P2P pures (scalabilité, fiabilité...) et ceux du protocole SIP (simplécité, adaptation aux application de VoIP...).

Conclusion générale et perspective

LA voix et la vidéo sur IP prennent des dimensions de plus en plus importantes depuis quelques années. D'autre part, la téléphonie entre PCs via l'Internet commence à prendre une part importante dans le monde des télécommunications. Dans un avenir proche, l'utilisation coûteuse du réseau de téléphonie fixe ne sera plus nécessaire, surtout avec la possibilité de transférer la voix , la vidéo et les données sur le même support via l'internet. D'où la nécessité d'évoluer vers des solutions IP ce qui provoque l'émergence de nouveaux standards Par rapport à la téléphonie classique, la téléphonie IP offre non seulement des perspectives intéressantes de simplification d'architecture et d'administration des équipements, mais aussi des nouveaux services enrichis, intégrant des applications informatiques. Le déploiement de cette technologie permet de franchir un premier pas dans la convergence des réseaux voix et données.

À L'heure actuelle, SIP se présente comme le protocole de signalisation le plus adéquat aux applications de voix et vidéo sur IP. Sa simplicité relative par rapport au standard H.323, le rend de plus en plus populaire dans ce domaine. En effet, des études comparatives de ces deux protocoles, ont fait ressortir les forces de chacun des deux standards et ont montré que SIP se présente comme concurrent principal du standard H.323 dans ce domaine de la voix et de la vidéo sur IP.

Dans le contexte de notre travail, nous avons introduit les concepts de la téléphonie sur IP, et nous avons définit les protocoles et concepts de mise en oeuvre et de signalisation. Nous avons proposés dans ce rapport une amélioration des travaux existants sur la téléphonie IP basé sur SIP-P2P. Notre solution s'inspire du protocole Chord qui est basé sur le principe des DHT au sein d'un réseau Pair-à-Pair. L'utilisation de Chord nous assure une identification unique des personnes, cette identification étant nécessaire pour empêcher un adversaire de

contrôler trop de ressources dans le réseau Pair-à-Pair. De plus, nous avons proposé une solution qui optimise le nombre de sauts d'une requête, ceci en se basant sur une méthode de routage récursive.

Au final, on peut conclure que les applications pair à pair ne sont pas déployées dans les réseaux ad hoc. La plupart des applications pair à pair existantes sont réellement déployé dans les réseaux traditionnels. Cependant, la majorité des réseaux ad hoc se fondent sur le paradigme pair à pair, pour la raison simple qu'ils sont sans infrastructure. Leurs différents noeuds ont généralement des rôles symétriques.

En perspectives, nous allons évaluer les performance de notre solution en utilisant la méthode de simulation en utilisant le simulateur OPNET ou NS2, et en comparant la solution que nous avons proposé avec les autres solutions existante de P2PSIP.

Une autre perspective est de pouvoir améliorer notre solution pour sécuriser les communications téléphoniques effectuées depuis un réseau P2P vers le réseau fixe pour assurer la sécurisation de bout en bout de l'appel.

Bibliographie

[1] S. Androutsellis-Theotokis and D.Spinellis. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys*, 4(36) :335–371, Décembre 2004.

[2] S. Baset and H. Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. *IEEE INFOCOM 2006,*, Avril 2006.

[3] Carole Bassil. *Secure Voice over IP Simple Protocol.* PhD thesis, École Nationale Supérieure des Télécommunications, Décembre 2005.

[4] David A. Bryan, Bruce B. Lowekamp, and Cullen Jennings. SOSIMPLE : A Serverless, Standards-based, P2P SIP Communication System. In *First International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications (AAA-IDEA'05), IEEE*, pages 42–49, Washington, DC, USA, 2005. IEEE Computer Society.

[5] Gonzalo Camarillo. *SIP Demystified.* McGraw-Hill Professional, 2001.

[6] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and Dan S.Wallach. Secure routing for structured peer-to-peer overlay networks. *SIGOPS Oper. Syst. Rev.*, 36 :299–314, Décembre 2002.

[7] I. Dalgic and H. Fang. Comparison of H.323 and SIP for IP telephony signaling. *Photonics East, (Boston, Massachusetts), SPIE*, Septembre 1999.

[8] M. Datar. Butterflies and peer-to-peer networks. In R. Möhring and R. Raman, editors, *Proceedings of 10th Annual European Symposium on Algorithms - ESA'02, number 2461 in LNCS*, pages 310–322, London, UK, 2002. Springer-Verlag.

[9] Guillaume Doyen. *Supervision des réseaux et services pair à pair.* PhD thesis, Université Henri Poincaré-Nancy 1, décembre 2005.

[10] M. Stiemerling et al. SIP : Protocol Overview. 2001.

[11] P. Fraigniaud and P. Gauron. An overview of the content-addressable network D2B. In *Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing - PODC'03*, pages 151–151, New York, NY, USA, 2003. ACM Press.

[12] K. Gummadi, S. Ratnasamy, I. Stoica, R. Gummadi, S. Gribble, and S. Shenker. The impact of DHT routing geometry on resilience and proximity. In *proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications - SIGCOMM'03*, pages 381–394, New York, NY, USA, 2003. ACM Press.

[13] M. Handley and V. Jacobson. SDP : session description protocol. *RFC 2327, Internet Engineering Task Force*, Avril 1998.

[14] O. Hersent, D. Gurle, and J.-P. Petit. *IP telephony : Voice-over-IP Protocols.* John Wiley & Sons, March 2005.

[15] M. Holredge and P. Srisureh. Protocol Complications with the IP Network Address Translator. *RFC3027 - Internet Engineering Task Force*, Janvier 2001.

[16] J. Stribling J. Li, T.M. Gil, R.Morris, and M. F. Kaashoek. Comparing the performance of distributed hash tables under churn. In G. M. Voelker and S. Shenker, editors, *proceedings of the 3rd International Workshop on Peer-to-Peer Systems (IPTPS'04)*, volume 3279, San Diego, CA, February 2004.

[17] Yuh-Jzer Joung and Jiaw-Chang Wang. Chord2 : A two-layer Chord for reducing maintenance overhead via heterogeneity. July 2006.

[18] D. Richard Kuhn, Thomas J. Walsh, and Steffen Fries. Security Considerations for Voice Over IP Systems.

[19] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the evolution of peer-to-peer systems. In *the 21st annual symposium on Principles Of Distributed Computing - PODC'02*, pages 233–242, New York, NY, USA, July 2002. ACM Press.

[20] Dahlia Malkhi, Moni Naor, and David Ratajczak. Viceroy : A scalable and dynamic emulation of the butterfly. *21st Annual ACM Symposium on Principles of Distributed Computing (ACM PODC'02) Monterey, California, USA*, pages 183–192, Juillet 2002.

[21] J. Mischke and B. Stiller. Peer-to-peer overlay network management through Agile. In G. Goldszmidt and J. Schönwälder, editors, *the 8th symposium on Integrated Network Management - IM'03*, pages 337–350. Kluwer Academic, 2003.

[22] M. Naor and U. Wieder. Novel architectures for P2P applications : the continuous-discrete approach. In *the 15th annual ACM symposium on Parallel Algorithms and Architectures - SPAA'03*, pages 50–59. ACM Press, 2003.

[23] C.Greg Plaxton, Rajmohan Rajaraman, and Andrea W. Richa. Accessing nearby copies of replicated Objects in a distributed environnement. In *the 9th Annual ACM Symposium on parallel algorithms and architectures (SPAA)*, pages 311–320, 1997.

[24] D. Ratajczak and J. M. Hellerstein. Deconstructing DHTs. Technical report, IRB-TR-03- 042, Intel Reasarch Berkeley, 2003.

[25] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A Scalable Content-Addressable Network. In *the ACM Conference on Applications, Technologies, Architectures and Protocols for Computer Communication - SIGCOMM'01*, pages 161–172. ACM Press, 2001.

[26] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a DHT. In *proceedings of the USENIX Annual Technical Conference*, pages 10–10, Berkeley, CA, USA, June 2004. USENIX Association.

[27] J. Rosenberg. Interactive Connectivity Establishment (ICE) : A Methodology for Network Address Translator (NAT) Traversal for the Session Initiation Protocol (SIP) . June 2003.

[28] J. Rosenberg and H. Schulzrinne. An offer/answer model with session description protocol (SDP). *RFC 3264, Internet Engineering Task Force*, Juin 2002.

[29] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP : session initiation protocol. *RFC 3261, Internet Engineering Task Force*, Juin 2002.

[30] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - simple traversal of user datagram protocol (UDP) through network address translators (NATs). *RFC 3489, Internet Engineering Task Force*, Mars 2003.

[31] A. Rowstron and P. Druschel. Pastry : Scalable, distributed object location and routing for large-scale peer-to-peer systems. *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), (Heidelberg, Germany)*, pages 329–350, Novembre 2001.

[32] Rüdiger Schollmeier. A Definition of Peer-to-Peer Networking for the Classification of Peer-to- Peer Architectures and Applications. In *the First International Conference on Peer-to-Peer Computing (P2P.01)*, 2002.

[33] H. Schulzrinne and J. Rosenberg. Internet telephony : Architecture and protocols - an IETF perspective. *Computer Networks and ISDN Systems*, 31, Février 1999.

[34] K. Singh and H. Schulzrinne. Peer-to-peer internet telephony using SIP. *NOSSDAV 2005, (Skamania, Washington)*, Juin 2005.

[35] Kundan Narendra Singh. *Reliable, Scalable and Interoperable Internet Telephony.* PhD thesis, COLUMBIA UNIVERSITY, 2006.

[36] M. P. Singh. Being Interactive : Peering at Peer-to-Peer Computing. *IEEE Internet Computing*, 2001.

[37] E. Sit and R. Morris. Security considerations for peer-to-peer distributed hash tables. In *the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), (Cambridge, MA, USA), IEEE*, Mars 2002.

[38] William Stallings. The Internet Protocol Journal, The Session Initiation Protocol. page 20, Mars 2003.

[39] T.Berners-Lee, R. Fielding, and L. Masinter. Uniform resource identifiers (URI) : generic syntax. http ://www.ietf.org/rfc/rfc2396.txt, Août 1998. Network Working Group, The Internet Society Internet Society.