ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE À L'OBTENTION
DE LA MAÎTRISE EN GÉNIE ÉLECTRIQUE
M. ING.

PAR
MOURAD EL ALLIA

DÉVELOPPEMENT D'UN ENVIRONNEMENT DE COMMUNICATION MULTIMÉDIA (VOIX ET VIDÉO) SUR INTERNET

MONTRÉAL, LE 16 OCTOBRE 2002

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

- Mme. Rita Noumeir, directeure de mémoire
  Département de génie électrique à
  l'École de Technologie Supérieure

- M. Abdellatif Obaid, codirecteur
  Laboratoire de recherche : Systèmes informatiques répartis,
  Département d'informatique
  Université du Québec à Montréal

- M. Mohamed Cheriet, professeur
  Département de génie de la production automatisée à
  l'École de Technologie Supérieure

- M. Cristopher Fuhrman, professeur
  Département de génie électrique à
  l'École de Technologie Supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET UN PUBLIC

LE 20 SEPTEMBRE 2002

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

DÉVELOPPEMENT D'UN ENVIRONNEMENT DE COMMUNICATION MULTIMÉDIA (VOIX ET VIDÉO) SUR INTERNET

Mourad El Allia

SOMMAIRE

De nos jours, la voix et la vidéo sur IP occupe une place privilégiée dans le monde des télécommunications. L'avantage incontesté de cette technologie est sa possibilité d'intégrer la voix, la vidéo et les données sur une même infrastructure Internet existante déjà. Grâce à cette technologie les coûts des communications interurbaines ont chuté de manière considérable ce qui laisse croire qu'elle a encore de beaux jours devant elle. Différents standards et protocoles de communications ont été élaborés pour rendre possible cette communication numérique. Parmi ces protocoles notons les protocoles de signalisation SIP et H.323 ainsi que les protocoles de transport en temps réel RTP et RTCP que nous avons étudiés et comparés dans ce mémoire.

Nous avons également modélisé, en utilisant l'approche par objet, le protocole SIP. Sa mise en œuvre, a été réalisée selon une architecture client-serveur à l'aide des librairies multimédia du langage de programmation Java pour transmettre et recevoir la voix et la vidéo via Internet.

DEVELOPMENT OF A MULTIMEDIA COMMUNICATION ENVIRONMENT (VOICE AND VIDEO) OVER INTERNET

Mourad El Allia

ABSTRACT

Nowadays, voice and video over IP occupy a privileged place in the telecommunication world. The uncontested advantage of this technology is its capabilities to mix voice, video and data using the same existing internet infrastructure. Another advantage is the cost of the long-distance communications which has decreased considerably, and let us suppose that this technology has a great future. Many standards and communications protocols have been developed to support such technology. Among them there are signaling protocols like SIP and H.323, and real-time protocols like RTP and RTCP that we have studied and compared in this work.

We propose an object oriented solution that models the SIP protocol. Based on a client-server architecture, we have also succeeded in implementing the SIP protocol, by using the Java Media Framework in order to transmit and receive voice and video on the Internet.

AVANT-PROPOS ET REMERCIEMENTS

Je remercie sincèrement ma directrice de mémoire Rita Noumeir professeure au département de génie électrique à l'ÉTS (École de Technologie Supérieure) et aussi mon co-directeur Abdellatif Obaid, directeur du laboratoire des systèmes informatiques répartis de l'université du Québec à Montréal, sans l'initiative desquels ce projet n'aurait pas été possible. Je tiens à leur exprimer toute ma reconnaissance pour leur dévouement, la confiance qu'ils m'ont accordée, leur rigueur et la qualité des commentaires et suggestions dont ils m'ont fait part.

Je remercie particulièrement ma chère épouse, qui ma soutenue et encouragé durant toute la période de mes études, et surtout mes chers parents, mes frères et ma sœur, pour leur encouragement et leur soutien moral malgré la distance qui nous sépare, sans oublier mes collègues du laboratoire des systèmes informatiques répartis de l'UQAM et tous mes amis d'ici et d'ailleurs.

TABLE DES MATIÈRES

Page

SOMMAIRE i

ABSTRACT ii

AVANT-PROPOS ET REMERCIEMENTS iii

LISTE DES TABLEAUX vi

LISTE DES FIGURES vii

LISTE DES ABRÉVIATIONS ET DES SIGLES viii

INTRODUCTION 1

1. Problématique 2
2. Objectif du mémoire 3
3. Motivation et méthodologie 4
4. Plan du mémoire 5

CHAPITRE 1 LA TÉLÉPHONIE SUR IP ET PROTOCOLES 7

1.1 La téléphonie sur IP 7
1.1.1 Principe 7
1.1.2 Normalisation de la téléphonie sur IP 9
1.1.3 Qualité de service et capacité 10
1.2 La téléconférence multimédia 10
1.2.1 Les modes de communication 10
1.3 Protocoles 12
1.3.1 Protocole IP 12
1.3.2 Protocole TCP 13
1.3.3 Protocole UDP 13
1.4 Les protocoles de transport temps réel 14
1.4.1 Le protocole RTP 14
1.4.2 Le protocole RTCP 16
1.5 Conclusion 19

CHAPITRE 2 LE STANDARD H.323 20

2.1 Introduction 20
2.2 Historique du standard H323 20
2.3 Les éléments du H323 21
2.3.1 Terminaux 21
2.3.2 Gardes-barrières 21
2.3.3 Passerelles 21
2.3.4 Les unités de contrôle multipoint (MCUs) 22
2.4 Protocoles et procédures 22

2.5 La pile H323 24
2.5.1 Les codecs Audio 24
2.5.2 Les codecs Vidéo 25
2.5.3 Conférence de données 25
2.5.4 Mécanismes de contrôle et de signalisation 25
2.5.5 La signalisation 26
2.6 Conclusion 27

CHAPITRE 3 LE PROTOCOLE D'INITIATION DE SESSION 28

3.1 Introduction 28
3.2 Architecture de SIP 28
3.3 Modèle client-serveur 30
3.4 URL SIP 31
3.5 Les méthodes 32
3.6 Les codes d'état 33
3.7 Description des messages SIP 34
3.8 Diagramme des séquences 35
3.9 Discussion 37
3.10 Conclusion 39

CHAPITRE 4 CONCEPTION ET ARCHITECTURE 40

4.1 Modèle de dialogue 40
4.2 Architecture 43
4.2.1 Les composantes de l'architecture 44
4.3 Conclusion 48

CHAPITRE 5 MISE EN ŒUVRE ET TESTS 50

5.1 Le langage Java 50
5.2 Java et le modèle client-serveur 51
5.3 Architecture des hiérarchies de classes 51
5.3.1 Identification des classes 53
5.4 Tests et résultats 72
5.4.1 Initiation d'appels 72
5.4.2 Enregistrement des clients 73
5.4.3 Invitation 74
5.4.4 Communication audio et vidéo 78
5.4.5 Statistiques de transmission et de réception 80
5.4.6 Terminaison d'appel SIP 82
5.5 Difficultés rencontrées 82
5.6 Conclusion 82

CONCLUSION 84

RÉFÉRENCES BIBLIOGRAPHIQUES 87

LISTE DES TABLEAUX

Page

Tableau I En-tête du paquet RTP 15
Tableau II Messages RTCP 17
Tableau III En-tête des paquets RTCP 18
Tableau IV Recommandations du H323 23
Tableau V Récapitulatif des codes 33
Tableau VI Message SIP 34

LISTE DES FIGURES

Page

Figure 1	Téléphonie de PC à PC	8
Figure 2	Téléphonie entre PC et poste téléphonique	8
Figure 3	Téléphonie entre postes téléphoniques	9
Figure 4	Le mode point à point	10
Figure 5	Le mode multipoint pleinement interconnectés	11
Figure 6	Le mode multipoint via un MCU	11
Figure 7	Architecture RTP	14
Figure 8	Le pile protocole du terminal H323	24
Figure 9	L'architecture de SIP	29
Figure 10	Établissement d'appel	35
Figure 11	Terminaison d'appel	36
Figure 12	Modèle de dialogue	41
Figure 13	Architecture de l'application	43
Figure 14	Tâches du serveur SIP	45
Figure 15	Transmission audio et vidéo	47
Figure 16	Réception audio et vidéo	48
Figure 17	Principales classes	52
Figure 18	Boîte de dialogue du client	72
Figure 19	Message Register	73
Figure 20	Message INVITE	74
Figure 21	Message INVITE	75
Figure 22	Message RINGING	75
Figure 23	Message OK	76
Figure 24	Message ACK	77
Figure 25	Message «connection établie»	77
Figure 26	Conférence à trois	79
Figure 27	Statistiques RTCP vidéo	80
Figure 28	Statistiques RTCP audio	81

LISTE DES ABRÉVIATIONS ET DES SIGLES

ATM	Asynchronous Transfer Mode
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISDN	Union Internationale de télécommunication
ITU-T	International Telecommunication Union
JMF	Java Media Framework
LAN	Local Area Network
LS	Location Server
MCU	Multipoint Control Unit
OSI	Open Systems Interconnection
PS	Proxy Server
RAS	Registration, Admission and Status
RFC	Request For Comment
RG	Registrar
RS	Redirect Server
RSVP	Resource Reservation Protocol
RTP	Real-Time Transport Protocol
RTCP	Real-Time Control Protocol
RTT	Round Trip Time
SAP	Session Announcement Protocol
SDP	Session Description Protocol
SIP	Session Initiation Protocol
TCP	Transmission Control Protocol
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
URI	Uniform Resource Indicator
URL	Uniform Resource Locator

INTRODUCTION

Le développement rapide de l'internet et l'utilisation croissante des réseaux fondés sur le protocole Internet (IP) pour les services de communications, y compris pour les applications telles que la téléphonie, sont devenus des domaines importants pour l'industrie des télécommunications. La possibilité d'acheminer du trafic vocal et de la vidéo sur des réseaux IP et les avantages offerts, notamment au niveau de l'intégration voix-données constituent un point de convergence entre deux technologies : la commutation de circuits et la commutation de paquets.

L'apparition récente de la transmission de la voix et de la vidéo sur IP représente une avancée technologique importante dans le domaine du multimédia et offre un service conçu pour permettre aux compagnies d'utiliser leurs réseaux Internet pour y faire passer leur trafic de la voix sans nécessiter de changement des équipements ou réseaux existants. En d'autre terme, l'ajout de quelques équipements tels que les passerelles permettent de garder les mêmes supports, utilisés auparavant pour acheminer les communications téléphoniques, pour véhiculer la voix, la vidéo et les données. Cette technologie exige des protocoles spécialisés dédiés à ce genre d'applications, comme le protocole de transport en temps réel RTP [12, 28] utilisé en parallèle avec d'autres protocoles qui concernent surtout la signalisation, la demande de réservation de ressources, la négociation de capacité comme le standard H323 et le protocole d'Initiation de sessions (SIP) [1, 9, 23].

Aujourd'hui, la technologie de la téléphonie sur IP a produit plusieurs services basés sur les différents scénarios de communication (téléphonie PC à PC, téléphonie entre un PC et un poste téléphonique et téléphonie entre postes téléphoniques ou fax). En conséquence, cette technologie est devenue un outil de communication mutimédia basé sur le réseau internet, intégrant des outils d'interfaces avec les réseaux téléphoniques traditionnels.

Les analystes spécialistes des questions techniques annoncent depuis plusieurs années que toutes les formes de communications fusionneront tôt ou tard en une plate-

forme unique et, depuis quelques années, il semble évident qu'avec la technologie IP adoptée, qu'elle soit bien la plate-forme unificatrice de tous les réseaux de communications sur internet [15]. De même, le marché de la voix et de la vidéo sur IP a ouvert plusieurs perspectives en ce qui a trait à la téléphonie sur IP, téléconférence, transferts de données etc. et a contribué à la réduction des prix des communications internationales grâce à la concurrence. L'importance de cette technologie et l'avenir qui lui est réservé nous a encouragés à s'impliquer dans ce domaine avec enthousiasme.

Notre objectif majeur derrière ce travail est la modélisation orientée objet du protocole de signalisation SIP, sa mise en œuvre et l'implémentation d'un modèle de communication PC à PC.

Dans ce mémoire, nous présentons une revue bibliographique sur le protocole H323 et le protocole d'initiation de session SIP. Ces deux protocoles sont utilisés dans des applications de voix et vidéo conférence. Nous passerons en revue aussi les protocoles de transmission et de contrôle en temps réel RTP et RTCP qui sont mis en jeu dans la transmission et la réception des données en temps réel et le contrôle des flux. Nous illustrerons les résultats des expériences réalisées.

1. Problématique

Les sessions multimédia sur Internet sont devenues de plus en plus populaires. Parmi les problèmes pour établir une session multimédia entre utilisateurs nous citons : comment trouver tous les utilisateurs? comment les informer et les inviter à participer à une communication? comment peuvent ils obtenir des informations sur les paramètres d'une session? comment et par quels moyens logiciels véhiculer du trafic audio et vidéo? quels sont les protocoles de communications mis en jeu dans telle ou telle application multimédia? et enfin comment mettre en œuvre et implémenter un modèle de communication basé sur les protocoles de signalisation SIP ou H323 et sur les protocoles de transport et de contrôle en temps réel RTP et RTCP via Internet?.

Notre contribution dans ce travail est de répondre à toutes ces questions posées. Nous avons, entre autre à étudier les différents protocoles de signalisation, à faire le choix du protocole de signalisation qui répond à nos besoins, à étudier et à utiliser le langage Java et l'API de ce langage dédiée à ce type de communication multimédia. Cette API est l'outil logiciel qui nous permettra de véhiculer des données soumises à des contraintes de temps réel. Enfin nous avons aussi à mettre en œuvre cette application qui servira à d'autres travaux de l'équipe de notre laboratoire.

Nous avons donc conçu deux architectures différentes et indépendantes. Une architecture concerne la communication multimédia, responsable de la transmission et la réception des données audio et vidéo. Une autre concerne la signalisation d'appels. En dernière étape nous avons rassemblé ces deux architectures pour en faire une plus globale avec gestion des paramètres et ports de communication.

2. Objectif du mémoire

L'objectif de notre mémoire est d'implanter un modèle de signalisation et de communication multimédia voix et vidéo en temps réel en utilisant Internet et les standards correspondants, dont H.323 ou SIP. En effet, il existe des logiciels tels que Netmeeting de Microsoft qui font de la communication multimédia. Cependant notre objectif s'inscrit dans l'utilisation et la manipulation des standards développés par les organismes de télécommunications tels que l'ITU-T et l'IETF, ce qui aidera à faciliter l'interopérabilité des équipements et des applications dans ce domaine. Pour cela, une recherche bibliographique préalable est nécessaire afin de comprendre les différents protocoles et permettre de choisir le protocole de signalisation qui s'adapte le mieux à notre application.

Cette application est implantée avec une approche orientée objet en utilisant le langage de programmation Java pour des raisons de portabilité et de réutilisation. L'outil utilisé est Jbuilder3 de Borland.

Il s'agit donc de développer une application basée sur le modèle client-serveur, d'analyser et traiter les messages échangés entre différentes entités. Un autre volet de notre application consiste utiliser les protocoles spécifiques qui nous permettent d'acheminer la voix et de la vidéo sur Internet.

3. Motivation et méthodologie

Le domaine de la communication multimédia est en plein essor (coûts des communications interurbaines, croissance du marché, augmentation du nombre des utilisateurs d'Internet dans le monde, intégration de technologies dans le réseau Internet...). Les entreprises qui œuvrent dans ce domaine sont à la recherche de méthodes, de standards et de prototypes qui pourront aider à implanter ces technologies.

La méthodologie à suivre consiste en :

- Étude et comparaison des standards de la communication multimédia (H.323 et SIP).
- Modélisation et architecture de l'application
- Conception orientée objet (représentation UML, diagramme des classes).
- Implantation orientée objet en utilisant le langage de programmation Java et l'API Java Média Framework.
- Test et résultats de l'application.

Les éléments à considérer dans cette implantation sont :

- Le protocole de signalisation SIP : En effet, bien que H.323 soit omniprésent dans les communications en temps réel sur IP, les avantages de SIP sur le H323 sont certains. Notre choix s'est porté sur le protocole SIP au lieu de H.323 pour les raisons suivantes :

  1. La description de SIP est beaucoup plus simple que celle d'H323; il est plus léger et donc facile à mettre en œuvre, sans être moins complet pour autant. Par contre, H.323 est lourd, complexe et gourmand en ressources. De plus, l'implantation d'une tâche complète même pour une simple application dans H.323

nécessite plusieurs protocoles en même temps qui complète le service (RAS H225.0, H.245, H.225).

2. SIP est un protocole indépendant de la couche transport. Il peut aussi bien s'utiliser avec TCP que UDP. De plus, sa mise en œuvre est plus facile grâce au langage de programmation de haut niveau. Enfin une des caractéristiques du SIP est que c'est un protocole plus rapide. Dans les chapitres suivants nous allons détailler notre application et sa mise en œuvre.

Au chapitre 3 nous allons donner une comparaison entre les deux protocoles.

- Le protocoles de transport en temps réel (RTP) et le protocole de contrôle en temps réel (RTCP). Il s'agit d'inclure des implantations existantes dans l'interface Java Media Framework (JMF).

4. Plan du mémoire

Après une introduction sommaire de la problématique et des objectifs de ce mémoire dans le présent chapitre, le plan des chapitres est comme suit :

Le chapitre 1 présente une introduction générale à la téléphonie sur Internet et aux conférences multimédia. Dans ce chapitre, nous donnons un petit aperçu sur la téléphonie sur IP et les différents modes de communications. Nous donnons également des définitions des importants protocoles de communication tels que les protocoles de la couche transport TCP, UDP et les protocoles de transport et de contrôle temps-réel RTP et RTCP.

Le chapitre 2 est consacré à une brève étude sur le standard H323 et des protocoles qu'il enveloppe. On présentera également l'architecture H323 et les différents composantes physiques et logicielles qui la constituent ainsi que la signalisation des appels de H323.

Le chapitre 3 concerne le protocole de signalisation de session (SIP), où on décrira de manière très brève ce protocole et la signalisation des appels. Nous allons décrire ses

composantes, ses caractéristiques et ses fondements et on fera aussi une comparaison avec le standard H.323. Dans notre application, nous avons implanté des fonctions de ce protocole que nous détaillerons dans les autres chapitres.

Le chapitre 4 est consacré à l'architecture et au modèle de dialogue de notre application. Nous allons détailler les différentes composantes qui constituent toute l'architecture du système et leurs interactions.

Dans le chapitre 5 nous présentons la mise en œuvre de l'application, les principales classes qui constituent notre programme et des saisies d'écrans qu'on a effectuées lors de l'exécution du programme.
Nous finirons ce rapport par une conclusion générale, de commentaires et suggestions sur ce travail et d'une bibliographie.

CHAPITRE 1

LA TÉLÉPHONIE SUR IP ET PROTOCOLES

1.1 La téléphonie sur IP

1.1.1 Principe

La téléphonie sur IP est la transmission de la voix sur les réseaux Internet. Le principe de base de cette transmission consiste en [22, 54] :

1. La voix à transmettre est échantillonnée et numérisée par un convertisseur analogique numérique.
2. Le signal numérique obtenu est comprimé et encodé grâce à des algorithmes de compression spécifiques .
3. Le signal est découpé en paquets.
4. Les paquets sont transmis à travers le réseau Internet

À la réception, les paquets sont ré-assemblés, le signal de données ainsi obtenu est décomprimé puis convertis en signal analogique sonore.

Selon le type de terminal utilisé , on distingue trois scénarios possibles de téléphonie sur IP [22, 54].

1. Téléphonie de PC à PC

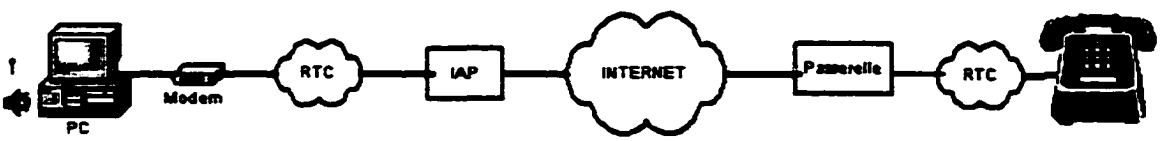
   Dans ce scénario , les deux correspondants utilisent un PC rattaché au réseau Internet par l'intermédiaire d'un fournisseur d'accès Internet. Cette technique nécessite des participants à la communication d'avoir un PC muni d'un modem, d'une carte réseau, d'un microphone, d'un haut-parleur et d'un logiciel de téléphonie IP compatible de chaque côté. La voix est comprimée et décomprimée par un logiciel de compression. 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, comme nous allons

l'expliquer dans les prochains chapitres, des protocoles de signalisations ont été élaborés pour pallier à ce genre de problème.

RTC:Réseau téléphonique commuté
IAP:Fournisseur d'accès à Internet

Figure 1 Téléphonie de PC à PC

2. Téléphonie entre PC et poste téléphonique et vis-versa :
   Dans ce scénario, l'un des correspondants utilise un PC rattaché au réseau Internet par un fournisseur d'accès Internet, l'autre correspondant utilise un téléphone rattaché au réseau téléphonique commuté. Une passerelle est nécessaire ente les deux réseaux pour rendre possible cette technique et faire la conversion entre réseaux (dans ce cas elle fait la conversion Internet-RTC et vis-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.



RTC:Réseau téléphonique commuté
IAP:Fournisseur d'accès à Internet

Figure 2 Téléphonie entre PC et poste téléphonique

3. Téléphonie entre postes téléphoniques

   Dans ce cas 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 IP en voix et vis-versa. Le réseau Internet est utilisé pour la connexion longue distance.

RTC:Réseau téléphonique commuté
IAP:Fournisseur d'accès à Internet

Figure 3 Téléphonie entre postes téléphoniques

1.1.2 Normalisation de la téléphonie sur IP

La plupart des téléphones sont encore, et seront encore pendant plusieurs années, connectés aux réseaux téléphoniques traditionnels à commutation de circuits. Les services de téléphonie IP doivent donc pouvoir accepter tout trafic émanant de ces réseaux et assurer la terminaison d'une communication.

La normalisation technique de la téléphonie 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) [15].

Un exemple de normalisation dans le cadre de l'IUT est la série de recommandations H323 pour les champs suivants : 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 par l'IETF pour la conférence, la téléphonie, la notification d'événements et la messagerie instantanée. Ces protocoles seront détaillés aux chapitres suivants.

1.1.3 Qualité de service et capacité

La qualité de service est un facteur déterminant dans la téléphonie et, à ce titre, occupe le centre du débat sur la téléphonie IP. La qualité peut se définir sur plusieurs plans : fiabilité, débit, sécurité.
Néanmoins, c'est parce que la qualité de transmission de la téléphonie caractérisant actuellement l'Internet public est perçue comme médiocre que la téléphonie sur Internet est rarement considérée comme un service de qualité. En règle générale, pour améliorer la qualité, on peut soit mettre en application des critères de qualité de service, soit accroître la capacité disponible [15].

1.2 La téléconférence multimédia

La téléconférence multimédia désigne l'ensemble des moyens et des services accessibles par des terminaux spécifiques permettant la tenue et l'animation de réunions entre deux ou plusieurs groupes de personnes éloignés.
Toutes les applications de téléconférence utilisent un ou plusieurs médias : la voix, l'image fixe ou animée et/ou des données en temps réel [6].

1.2.1 Les modes de communication

1. Le mode point à point
   Le mode point à point correspond à une communication entre deux participants. C'est le mode de fonctionnement le plus simple à mettre en œuvre.

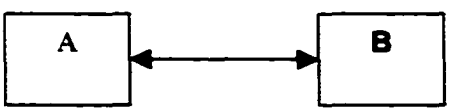


Figure 4 Le mode point à point

2. Le mode multipoint pleinement interconnectés

   Ce mode correspond au cas où une téléconférence met en relation plus de deux participants simultanément. Ces participants sont interconnectés en multicast.

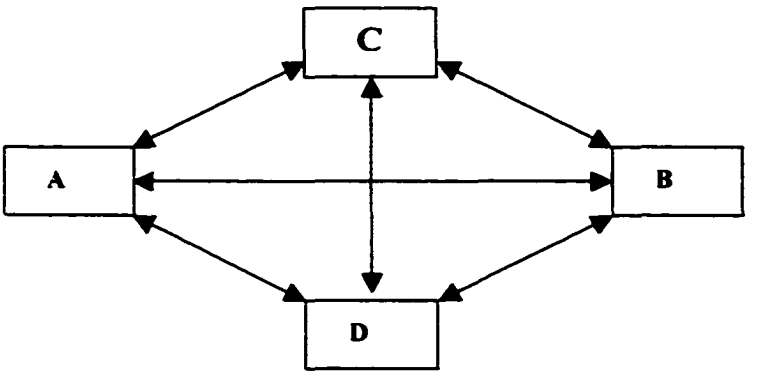


Figure 5 Le mode multipoint pleinement interconnectés

3. Le mode multipoint via un MCU :

   Dans ce mode plusieurs participants sont connectés via une unité de contrôle MCU (Multipoint Control Unit). Celle-ci gère les procédures d'appels entrants et sortants (entre le MCU et les terminaux).

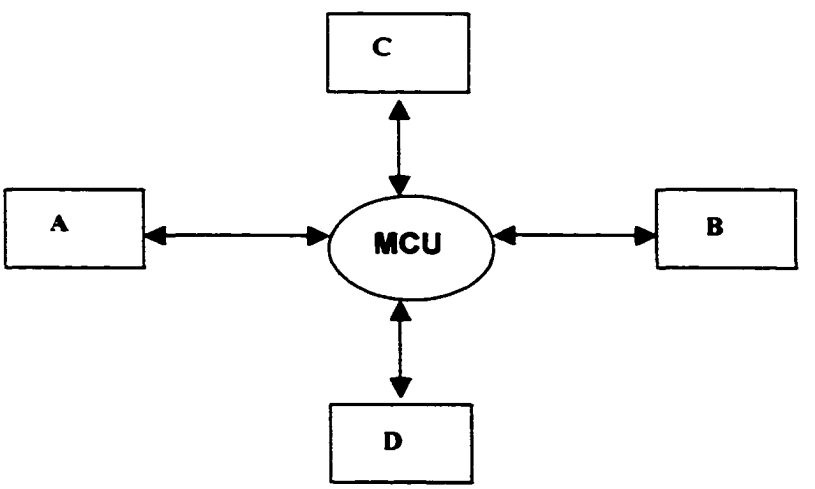


Figure 6 Le mode multipoint via un MCU

1.3 Protocoles

Le transfert de données sur Internet s'effectue par paquets de données. Cette structure repose sur l'utilisation de protocoles TCP/IP (Transport Control Protocol/ Internet Protocol).

Chaque document, qu'il s'agisse de texte, image ou voix, est numérisé puis réparti en paquets. Chacun de ces paquets est alors envoyé sur Internet indépendamment des autres et essaie de prendre le chemin le plus rapide pour parvenir à sa destination. Ceci est réalisé en fonction de l'encombrement d'une partie ou de l'autre du réseau au moment où le paquet est expédié. La segmentation de l'information permet une plus grande flexibilité dans l'utilisation des ressources puisque la communication ne monopolise pas une ligne donnée. Dans le reste de cette section, nous ferons un bref descriptif des protocoles de transport utilisés:

1.3.1 Protocole IP

Le protocole IP est au centre du fonctionnement de l'Internet. Il fait partie de la couche Internet de la suite de protocoles TCP/IP. Il assure sans connexion un service non fiable de délivrance de paquets IP. Le service est non fiable car il n'existe aucune garantie pour que les paquets IP arrivent à destination. Certains paquets peuvent être perdus, dupliqués ou remis en désordre. On parle de remise au mieux. Le protocole IP permet aux paquets de se déplacer sur le réseau Internet, indépendamment les uns des autres, sans liaison dédiée. Chacun d'entre eux, envoyé sur le réseau, se voit attribuer une adresse IP. Cette dernière est un en-tête accolé à chaque paquet et contenant certaines informations, notamment, l'adresse destinataire, sa durée de vie, le type de service désiré, etc. Le protocole IP actuellement utilisé en est à la version 4 et la nouvelle version Ipv6 est déjà prête à prendre le relais .

1.3.2 Protocole TCP

Le protocole TCP est un protocole de contrôle de transmission, il fait partie de la couche transport du modèle OSI. Il est orienté connexion, c'est à dire, il assure un circuit virtuel entre les applications utilisateurs. Le protocole TCP établit un mécanisme d'acquittement et de re-émission de paquets manquants. Ainsi, lorsqu'un paquet se perd et ne parvient pas au destinataire, TCP permet de prévenir l'expéditeur et lui réclame de renvoyer les informations non parvenues. Il assure d'autre part un contrôle de flux en gérant une fenêtre de congestion qui module le débit d'émission des paquets. Il permet donc de garantir une certaine fiabilité des transmissions. TCP assure un service fiable et est orienté connexion [33,34], cependant il ne convient pas à des applications temps réel à cause des longs délais engendrés par le mécanisme d'acquittement et de retransmission [1].

1.3.3 Protocole UDP

Le protocole de datagramme utilisateur (UDP) est le protocole de transport sans confirmation. UDP est un protocole simple qui permet aux applications d'échanger des datagrammes sans accusé de réception ni remise garantie [33,34]. Le traitement des erreurs et la retransmission doivent être effectués par d'autres protocoles. UDP n'utilise ni fenêtrage, ni accusés de réception, il ne reséquence pas les messages, et ne met en place aucun contrôle de flux. Par conséquent, la fiabilité doit être assurée par les protocoles de couche application. Les messages UDP peuvent être perdus, dupliqués, remis hors séquence ou arriver trop tôt pour être traiter lors de leurs réception. UDP est un protocole particulièrement simple conçu pour des applications qui n'ont pas à assembler des séquences de segments. Son avantage est un temps d'exécution court qui permet de tenir compte des contraintes de temps réel ou de limitation d'espace mémoire sur un processeur, contraintes qui ne permettent pas l'implémentation de protocoles beaucoup plus lourds comme TCP.
Dans des applications temps-réel, UDP est le plus approprié, cependant il présente des faiblesses dues au manque de fiabilité. Des protocoles de transport et de contrôle temps-réel sont utilisés au dessus du protocole UDP pour remédier à ses faiblesses et

assurer sa fiabilité [1]. Ces protocoles sont RTP et RTCP et sont détaillés dans le paragraphe suivant.

1.4 Les protocoles de transport temps réel

1.4.1 Le protocole RTP

Le groupe de l'IETF a développé en 1993 le protocole de transport en temps réel (RTP, RFC 1889) dont le but est de transmettre sur Internet des données qui ont des propriétés temps réel (audio, vidéo ...) [1,55]. C'est un protocole de la couche application du modèle OSI et utilise les protocoles de transport TCP ou UDP, mais, généralement, il utilise UDP qui est mieux approprié à ce genre de transmission.
La figure 7 représente l'architecture du protocole RTP.

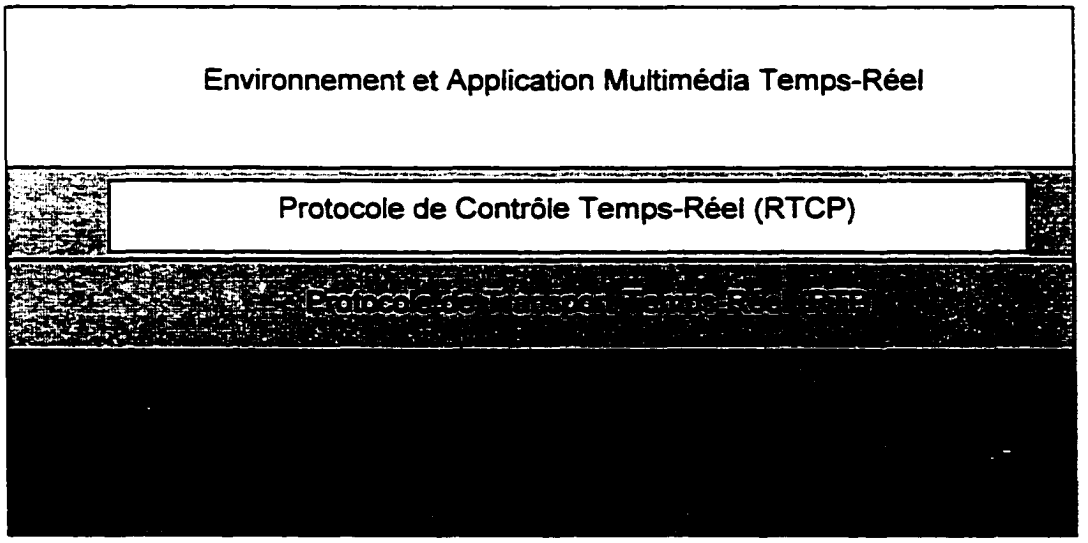


Figure 7 Architecture RTP

Le rôle principal de RTP consiste à mettre en œuvre des numéros de séquences de paquets IP et des mécanismes d'horodatages (timestamp) pour permettre de reconstituer les informations de voix ou vidéo. Plus généralement, RTP permet

- D'identifier le type de l'information transportée;

- D'ajouter des indicateurs de temps (horodater) et des numéros de séquence à l'information transportée;
- De contrôler l'arrivée à destination des paquets.

L'en-tête RTP (Tableau I) indique le type, la source, et les caractéristiques de contrôle du temps des données encodées. Ce standard inclut en effet un horodateur (Timestamp) des paquets permettant au destinataire d'utiliser la numérotation des séquences pour reconstituer l'ordre original des paquets et de détecter les paquets perdus.

Tableau I

En-tête du paquet RTP

V	P	X	CC	M	PT	Numéro de Séquence
Horodatage (Timestamp)						
Identificateur de source de synchronisation (SSRC)						
Identification des sources contributrices (CSRC)						
En-tête supplémentaires						
Données						

- Le champ *V* (2 bits) : Version RTP, actuellement V=2.
- Le champ *P* (1 bit) : Si P=1 le paquet contient des octets additionnels de bourrage (padding) pour finir le dernier paquet.
- Le champ Extension X (1 bit) : Si X=1, l'en-tête est suivie d'un paquet d'extension.
- Le champ CSRC count CC (4 bits) : Il contient le nombre de CSRC qui suivent l'en-tête.
- Le champ *M* (1 bit) : Son interprétation est définie par un profil d'application (Profile).
- Le champ *Données* type PT (7 bits) : Ce champ identifie le type de données (audio, vidéo, image, texte...).

- Le champ *numéro de séquence* (16 bits) : Sa valeur initiale est aléatoire et il s'incrémente de 1 à chaque paquet envoyé, il peut servir à détecter des paquets perdus.
- Le champ *horodatage* (32 bits) :Ce champ reflète l'instant où le premier octet du paquet RTP a été échantillonné.
- Le champ *SSRC* (32 bits) :Ce champ identifie de manière unique la source de synchronisation. Sa valeur est choisie de manière aléatoire par l'application.
- Le champ *CSRC* (32 bits) : Ce champ identifie les sources participantes.

RTP supporte différents types de codage et de compression des données. Beaucoup de formats standardisés sont acceptés (GSM, G.723.1, G.729 pour l'audio et MPEG, H261 pour la vidéo).

1.4.2 Le protocole RTCP

RTCP (Real Time Control Protocol, RFC 1889) est un protocole de contrôle utilisé conjointement avec RTP pour contrôler les flux de données et la gestion de la bande passante. [1,55].
RTCP permet de contrôler le flux RTP, et de véhiculer périodiquement des informations de bout en bout pour renseigner sur la qualité de service de la session de chaque participant à la session. Des quantités telles que le délai, la gigue, les paquets reçus et perdus sont très important pour évaluer la qualité de service de toute transmission et réception temps réelles.
C'est le protocole sous-jacent (UDP par exemple) qui permet grâce à des numéros de ports différents et consécutifs (port pair pour RTP et port impair immédiatement supérieure pour RTCP) le multiplexage des paquets de données RTP et des paquets de contrôle RTCP. Le protocole RTCP remplit trois fonctions :

- L'information sur la qualité de service : RTCP fournit, en rétroaction des informations sur la qualité de réception des données transmises dans les paquets RTP. Cette information est utilisé par la source émettrice pour adapter le type de codage au niveau des ressources disponibles .

- L'identification permanente : RTCP transporte un identificateur original de la source RTP c'est à dire la provenance du flux, appelé *CNAME* (Canonical name). Cet identificateur permet une identification permanente de chacun des flux multimédia entrants.
- Le calibrage de la fréquence d'émission : La réception des feed-back et la connaissance du nom permanent servent à ajuster la fréquence d'envoi des paquets à la bande passante mise à la disponibilité de l'utilisateur situé à l'autre extrémité.

Chaque paquet RTCP contient un rapport émetteur ou récepteur suivi d'une description de la source (source description).
Il existe cinq types de paquets de contrôle (Tableau II) . Chaque paquet commence par un en-tête fixe suivi d'éléments structurés qui peuvent être de longueur variable selon le type de paquet

Tableau II

Messages RTCP

RR	Rapport Émetteur
SR	Rapport Récepteur
SDES	Description de Source
BYE	Fin de session
APP	Nouveau champ

- *RR :* contient le rapport de la qualité de la livraison des données des participants passifs (récepteur) incluant le nombre de paquets reçus, le nombre de paquets perdus, la gique et l'horodatage qui permet le calcul du délai total de transmission entre les deux parties.

- *SR* :contient le rapport de la qualité de livraison des données des participants actifs (émetteurs). Il contient les champs du *RR*, et des informations sur

l'émetteur, la synchronisation (pour synchroniser deux sources de données), un compteur cumulatif de paquets et le nombre d'octets envoyés.

- *SDES* : contient l'information concernant les émetteurs comme le nom canonique (CNAME), et le nom d'usager (NAME), le numéro de téléphone (PHONE) et d'autres informations concernant les participants.

- *BYE* : indique la fin de la participation d'une des parties.

- *APP* : Dans le cas des applications spécifiques, les données peuvent être transmises dans des paquets spécifiques de type APP.

Le tableau III détaille l'en-tête ce message commun à tous les paquets RTCP :

Tableau III

En-tête des paquets RTCP

V	P	RC	PC	Longueur
Rapport (s)				

Avec

- Le champ *V* (2 bits) : Indique la version de RTP, actuellement V=2.
- Le champ P (1 bit) : Si P=1 le paquet contient des octets additionnels de bourrage (padding) pour finir le dernier paquet.
- Le champ *RC* (5 bits) : Il contient le nombre de rapports contenus dans le paquet (un paquet pour chaque source)
- Le champ *PT* (8 bits) : Il donne le type de rapport du paquet (PT=SR=200, PT=RR=201, PT=SDES=202, PT=BYE=203, PT=APP=204).
- Le champ *Longueur* (16 bits) : Il indique la longueur du paquet.

1.5 Conclusion

La voix et la vidéo sur IP procurent des avantages certains aux entreprises et aux utilisateurs. Un seul réseau est utilisé pour la voix, la vidéo et les données, ce qui a pour conséquence de réduire les frais d'exploitations. D'autre part, les utilisateurs peuvent éviter les frais dus aux appels interurbains et payer uniquement les frais de connexions locales. D'autres applications de la voix et la vidéo sur IP permettent la tenue de réunions et de conférences multimédia.

Dans ce chapitre, nous avons présenté le principe de la téléphonie sur IP et des modes de communications des téléconférences multimédia. Nous avons également étudié les protocoles qui permettent ce genre d'applications. Parmi ces protocoles, nous avons présenté les protocoles de transports et de contrôles en temps réel RTP et RTCP qui utilisent le protocole UDP pour le transport de la voix et de la vidéo.

Dans le chapitre prochain, nous allons étudier le standard H.323 qui permet entre autre la signalisation des appels.

CHAPITRE 2

LE STANDARD H.323

2.1 Introduction

Le standard H323, développé par l'ITU–T, est défini comme étant le standard spécifiant les éléments, les protocoles et les procédures pour réaliser des communications audio, vidéo et autres données en temps réel sur les réseaux de paquets.
Il décrit également les terminaux, équipements, et services nécessaires à l'établissement d'une communication multimédia sur un réseau local ne garantissant pas une qualité de service optimale.
Il spécifie aussi les protocoles, les méthodes et les éléments du réseaux qui sont nécessaire à l'établissement de connections multimédia point à point, multipoint et de conférences multimédia entre trois parties et plus.

2.2 Historique du standard H323

La première version a été approuvée en octobre 1996. Elle définit les standards pour les transmissions multimédias au dessus des réseaux IP. Cependant, cette version présentait des faiblesses telles que l'absence de la qualité de service. La deuxième version de H323, approuvée en janvier 1998, permet une interopérabilité entre différents réseaux. De plus, Il s'adapte facilement avec d'autres technologies de réseaux de paquets.
La troisième version de H323, approuvée le 30 septembre 1999, introduit quelques nouvelles fonctionnalités (identification de l'appelant, communication entre gardes-barrières ...). La quatrième version, a été approuvée en novembre 2000 [39].

2.3 Les éléments du H323

Les éléments de base du standard H323 sont les terminaux, les gardes-barrières, les passerelles et les unités de contrôle multipoint MCUs. Les MCUs sont cités séparément, mais en pratique il font souvent partie des gardes-barrières ou des ordinateurs rapides qui servent plusieurs utilisateurs [1,9].

2.3.1 Terminaux

Un terminal est un périphérique qui supporte les communications multimédia bidirectionnelles en temps réel . Tous les terminaux H323 doivent supporter H.245, RAS (Registration Admission Status) et RTP (Real Time Transport Protocol).
Un terminal H323 fournit les services suivants :

- Services H245 : l'échange de capacités et la gestion des canaux logiques.
- Services H225 : gestion de la signalisation des appels.
- Services RAS : l'enregistrement auprès du garde-barrière.
- Services RTP / RTCP : Séquencement des paquets audio et vidéo.

2.3.2 Gardes-barrières

Le garde-barrière est un élément vital dans un système H323. Il joue le rôle de contrôleur pour tous les appels à l'intérieur de la zone H323 (une zone H323 est une agrégation de garde-barrière et de tous les autres éléments terminaux et MCU qui sont enregistré auprès de lui). Il fournit les services aux éléments qui sont enregistrés auprès de lui tel que la conversion des adresses, le contrôle d'admission, la gestion de la bande passante et la capacité de routage.

2.3.3 Passerelles

Une passerelle H323 est un élément du réseau qui assure la conversion voulue entre les formats de transmission (par exemple, du format H.225.0 au format H.221 ou vice versa) ainsi qu'entre les protocoles de communication (par exemple du protocole

H.245 au protocole H.242 ou vice versa). La passerelle assure aussi l'établissement et la libération des communications tant du côté réseau à commutations de paquets que du côté réseau à commutation de circuits.

2.3.4 Les unités de contrôle multipoint (MCUs)

Le MCU est un élément du réseau qui fournit les capacités à plusieurs terminaux et passerelles pour participer à une conférence multipoint. En d'autres termes, un MCU gère les ressources de la conférence et les échanges de capacités. Le MCU se compose de deux parties : un contrôleur multipoint obligatoire qui assure la gestion d'au moins trois terminaux participants à une conférence multipoint. Le contrôleur multipoint permet de négocier avec tous les terminaux les moyens à mettre en œuvre pour parvenir à établir des communications multimédia. Il peut également exercer un contrôle au niveau des ressources de la conférence pour déterminer par exemple l'entité qui transmet en muticast. La deuxième partie est le processeur multipoint facultatif qui assure le traitement centralisé des flux de données dans une conférence multipoint.

2.4 Protocoles et procédures

La recommandation H323 enveloppe d'autres recommandations pour permettre les communications en temps réel.
Le tableau IV résume quelques unes d'entre elles :

Tableau IV

Recommandations du H323

Recommandation	Aperçu
Codecs Audio	
G.711	Encode le signal selon les lois A ou μ en 64 Kbit/s
G.723.1	Encode et compresse le signal vocal en 5.3 et 6.4 Kbit/s
G.729	Encode et compresse le signal vocal en 8 et 13 Kbits/s
Codecs Vidéo	
H.261	Encode et compresse la vidéo en 64 kb/s
H.263	Encode et compresse à faible taux de compression
Communication de données	
T.120	Protocole de données pour les conférences multimédia
Contrôle	
H.245	Protocole de contrôle pour les communications de données
H.225.0	Protocole de signalisation des appels
Transport temps réel	
RTP / RTCP	Protocoles de transport et de contrôle en temps réel
Sécurité	
H.235	Sécurité, cryptage pour les terminaux des séries H32x
Services supplémentaires	
H.450.1	Protocole pour les services supplémentaires
H.450.2 et 450.3	Transfert d'appels et autres services supplémentaires

2.5 La pile H323

La figure 8 montre la pile des protocoles spécifiés par le standard H323

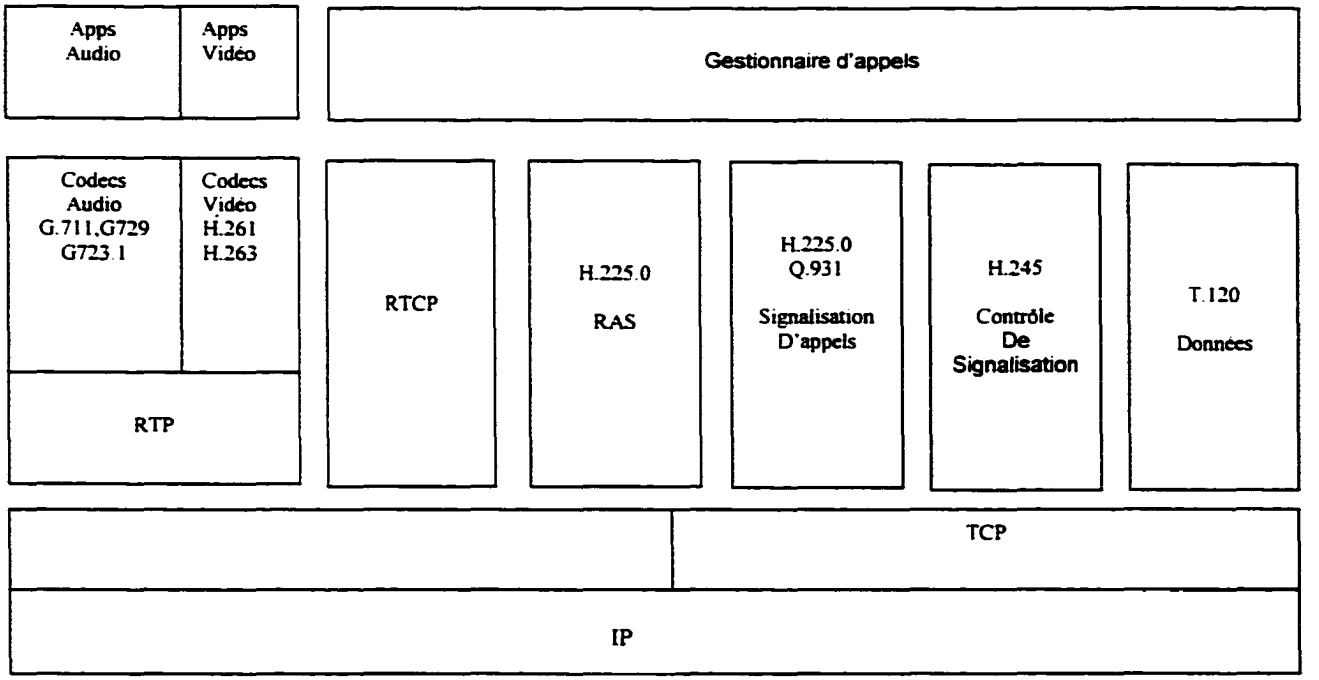


Figure 8 Le pile protocole du terminal H323

Cette pile est indépendante des réseaux et des protocoles de transport utilisés.
Si le protocole IP est utilisé (ce qui est le plus souvent le cas) alors les paquets audio, vidéo et H.225.0 RAS utilisent UDP comme protocole de transport alors que les paquets de contrôle (H.245 et H.225.0 call signaling) utilisent TCP.
La pile H323 est constituée des éléments décrits ci-dessous :

2.5.1 Les codecs Audio

H323 spécifie une série de codecs audio classés par débits allant de 5.3 à 64 kb/s. G.711 est le codec le plus populaire conçu pour les réseaux de téléphonie. Aujourd'hui, les terminaux H323 supportent le codec G.723.1 qui est plus efficace et produit une

meilleure qualité audio à 5.3 kb/s et 6.3 kb/s. Les codecs G.729 utilisent la quantification à prédiction linéaire pour produire une qualité supérieure à des taux de 16 kb/s et 8 kb/s.

2.5.2 Les codecs Vidéo

La communication vidéo nécessite une bande passante importante, d'où l'intérêt d'avoir des techniques de compression et de décompression performante.
H323 spécifie deux codecs vidéos : H.261 et H.263.
- Les codecs H.261 produisent la transmission vidéo pour des canaux avec une bande passante de p x 64 kb/s (p est une constante qui varie de 1 à 30)
- Les codecs H.263 sont conçus pour des transmissions à faible débit sans perte de qualité.

2.5.3 Conférence de données

Les capacités des conférence de données en temps réel sont requises pour des activités telles que le partage d'applications, le transfert de fichiers, la transmission de fax, la messagerie instantanée.
La recommandation T.120 fournit ces capacités optionnelles au H323.

2.5.4 Mécanismes de contrôle et de signalisation

Le flux d'informations dans les réseaux H323 est un mixage de paquets audio, vidéo, données et de contrôle. L'information de contrôle est essentielle pour l'établissement et la rupture des appels, l'échanges et la négociation des capacités. H323 utilise trois protocoles de contrôles : Contrôle multimédia H.245, signalisation d'appel H.225/ et H.225.0 RAS.

2.5.5 La signalisation

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.
Il existe actuellement deux protocoles de signalisation pour les réseaux IP, la signalisation H.225 qui fait partie du standard H323 et le récent protocole SIP.

2.5.5.1 Signalisation des appels H.225

La signalisation des appels est importante pour établir et rompre une connexion entre deux entités. Q.931 a été développé initialement pour la signalisation dans les Réseaux Numériques à Intégration de Service (ISDN). H.225.0 a adopté la signalisation Q.931 en l'incluant dans le format de ses messages
Deux entités désirant établir une connexion doivent ouvrir un canal de signalisation.. La signalisation d'appels H.225.0 est envoyé directement entre les entités périphériques quand aucun garde-barrière n'est utilisé. Si un garde-barrière est utilisé alors la signalisation d'appels H.225.0 doit être router à travers ce garde-barrière.

2.5.5.2 H.225.0 RAS

Les messages H.225.0 RAS (registration, admission, status) définissent une communication entre les terminaux et un garde-barrière. H.225.0 RAS s'occupe de la communication entre le garde-barrière et les différents terminaux. Elle gère les opérations suivantes : l'inscription, le contrôle d'admission, la gestion de la bande passante. Un canal de signalisation est utilisé afin de transporter les différents messages RAS.

2.5.5.3 Le protocole de contrôle de signalisation H.245

La flexibilité de H.323 nécessite que les différends terminaux négocient les capacités avant que les liens de la communication audio, vidéo et/ou données ne soit établit. H.245 utilise les messages de contrôle et de commandes qui sont échangés durant l'appel. Ces messages sont classés en quatre catégories :
Messages d'échanges de capacités

1. Messages pour la gestion des canaux logiques
2. Messages pour la gestion des flux de contrôle
3. Commandes générales et indications

2.6 Conclusion

Le standard H.323 est omniprésent dans les communications multimédia en temps réel et il est utilisé par plusieurs compagnies telles que Intel, Microsoft, Cisco, IBM, etc. Son indépendance des plates formes permet le concevoir des applications multimédia sans changer l'infrastructure des réseaux.

Dans le chapitre suivant nous allons étudié un autre protocole de signalisation concurrent du H.323 appelé protocole d'initiation de session. Nous présenterons aussi une comparaison entre ces deux protocoles.

CHAPITRE 3

LE PROTOCOLE D'INITIATION DE SESSION

3.1 Introduction

Développé par l'IETF et présenté dans le RFC 2543, le protocole d'initiation de session (SIP) est un protocole de signalisation appartenant à la couche application du modèle OSI. Son rôle est d'ouvrir, modifier et libérer les sessions entre un ou plusieurs utilisateurs. L'ouverture de ces sessions permet de réaliser de l'audio ou vidéoconférence, de l'enseignement à distance, de la voix (téléphonie) et de la diffusion multimédia sur IP. Notons qu'avec SIP, les utilisateurs qui ouvrent une session peuvent communiquer en mode multicast, en mode point à point ou dans un mode combinant ceux-ci.

Pour ouvrir une session, l'utilisateur émet une invitation transportant un descripteur de session permettant aux utilisateurs souhaitant communiquer de s'accorder sur la compatibilité de leur média. Ainsi SIP peut relier des stations mobiles en transmettant ou redirigeant les requêtes vers la position courante de la station appelée.

Notons que SIP possède l'avantage de ne pas être attaché à un médium particulier et est sensé être indépendant du protocole de transport. De plus il peut être étendu et s'adapter aux évolutions futures [1,23].

3.2 Architecture de SIP

L'architecture de SIP est présenté à la figure 9

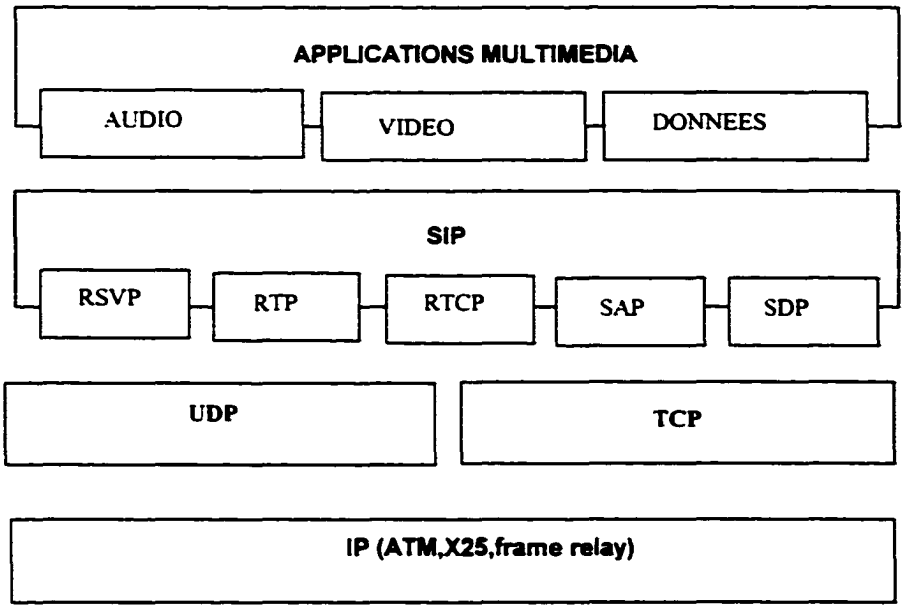


Figure 9 L'architecture de SIP

A chacune des couches de l'architecture SIP sont associés des protocoles tels que :

- *RSVP* (Resource Reservation Protocol) : protocole utilisé pour réserver les ressources réseaux sur IP.
- *RTP* (Real Time Transport protocol) : protocole de transport des données en temps réel
- *RTCP* (Real Time Control Protocol) : protocole qui assure le contrôle de flux des données multimédia
- *SAP* (Session Annoucement Protocol) : protocole pour annoncer si les sessions multimédia ouvertes sont en multicast
- *SDP* (Session Description Protocol) : protocole de description des sessions multimédia

3.3 Modèle client-serveur

Pour établir une communication, l'appelant, que l'on désignera par client, adressera sa requête à un serveur SIP, qui lui donnera les moyens de communiquer. Il existe cinq types de serveurs :

- l'UAS (User Agent Server) : c'est l'application du terminal d'abonné qui reçoit les requêtes. L'UAC (User Agent Client) est l'application cliente qui émet les requêtes;
- le relais mandataire ou PS (Proxy Server) : auquel est relié un terminal fixe ou mobile agit à la fois comme client et serveur. Un tel serveur peut interpréter et modifier les messages qu'il reçoit avant de les transmettre;
- le RS (Redirect Server) : réalise simplement une association d'adresses vers une ou plusieurs adresse (lorsqu'un client appelle un terminal mobile - redirection vers le PS le plus proche - ou en multicast, le message émis est redirigé vers toutes les sorties auxquelles sont reliés les destinataires);
- le LS (Location Server) fournit la position courante des utilisateurs dont la communication traverse les RS et PS auxquels ils sont rattachés : Cette fonction est assurée par le service de localisation;
- le RG (Registrar) est un serveur qui accepte les requêtes REGISTER et offre également un service de localisation comme le LS. Chaque PS ou LS est généralement relié à un Registrar.

Pour établir la communication entre l'UAC (Agent utilisateur client) du terminal appelant et l'UAS (Agent utilisateur serveur) du terminal appelé, ceux-ci doivent être identifiés. Ainsi, chaque utilisateur et sa machine est identifié par une adresse appelée URL SIP qui se présente comme une URL Mailto ou Telnet :

utilisateur@machine

La partie utilisateur est composé d'un nom ou un numéro de téléphone alors que la partie machine est un nom de domaine ou une adresse IP.

Si le client souhaite communiquer avec son destinataire, il envoie une requête URI directement vers le port d'entrée de l'UAS du destinataire de la requête. Le protocole utilisé (TCP ou UDP), l'adresse IP et le port d'entrée du serveur du destinataire doivent être précisés, lors de l'émission d'une requête URI.
Une fois le client connecté à un serveur SIP distant, il peut lui adresser une ou plusieurs requêtes SIP et recevoir une ou plusieurs réponses de ce serveur. Les réponses contiennent certains champs identiquement remplis à ceux des requêtes : ce sont les champs *Call-ID* dont le rôle est de définir de façon unique une invitation particulière, le champ *Cseq* qui est utilisé pour identifier un message faisant partie d'une négociation d'invitation, le champ *Via* qui indique le chemin pris par la requête pour arriver jusqu'à destination et les url To et From qui contiennent les adresses sources et destination des clients.
Les échanges client-serveur se font à l'aide de requêtes (INVITE, ACK, BYE, REGISTER, OPTION, CANCEL). Celles-ci sont définies dans les paragraphes ultérieurs.

3.4 URL SIP

Il existe cinq types de format de nommage universel SIP (URL SIP) qui sont : (FROM, COURANTE, TO, CONTACT et EXTERNAL).

- URL TO : contient l'adresse du destinataire de la requête SIP.
- URL FROM: contient l'adresse source du client qui émet la requête SIP.
- URL COURANTE ou requête URI: précise la destination de la requête REGISTER c'est à dire son domaine d'enregistrement. La requête REGISTER est transmise de serveur en serveur jusqu'à ce qu'elle ait atteint le serveur dont le domaine correspond à celui listé dans la requête URI.
- URL CONTACT : les requêtes autres que REGISTER à destination de l'URL TO sont redirigés aux adresses données dans l'URL CONTACT.
- URL EXTERNAL : réservé à des applications futures.

La structure de l'URL est la suivante :

sip :informations_utilisateur@domaine paramètres en-têtes

avec

informations_utilisateur = (nom de l'utilisateur :mot de passe) ou (numéro de téléphone si user = phone)

domaine = nom de domaine ou adresse IP : port

paramètres = ;transport = udp ou tcp; user = phone ou IP
; method = INVITE, ACK, OPTIONS, BYE, CANCEL, REGISTER
; ttl = 0 à 255
; maddr = adresse IP de multicast
; tag = compteur en héxadécimal

en-tête = ?hname= hvalue & hname = hvalue ....

Seules les URL CONTACT et EXTERNAL contiennent les paramètres transport, method, ttl, maddr et des en-têtes.

Nous illustrons par un exemple une requête SIP par l'exemple d'une personne nommée Jacques dont l'url est Jacques@uqam.ca qui invite une personne Bill ayant pour url Bill@ele.ets.ca . La requête INVITE sera la suivante :

INVITE sip :ele.ets.ca
Via: SIP/2.0/UDP ets.ca
From: sip:Jacques@uqam.ca
To: sip:Bill@ele.ets.ca
CallID: 1468@ets.ca
Cseq: 1 INVITE

3.5 Les méthodes

Les échanges client-serveur se font à l'aide de requêtes INVITE, ACK, BYE, REGISTER, OPTIONS et CANCEL.

Étudions en détail ces six méthodes qui permettent les échanges entre client et serveur :

- INVITE : indique que l'application ou utilisateur est invité à participer à une session.
- ACK :confirme que le client a reçu une réponse positive à une requête INVITE.
- OPTIONS : un PS en mesure de contacter l'UAS appelé, doit répondre à une requête OPTIONS précisant ses capacités à contacter l'UAS.
- BYE : est utilisée par l'UAS de l'appelé pour signaler au PS local qu'il ne souhaite plus participer à la session.
- CANCEL : la requête CANCEL est envoyée par un UAC ou un PS pour annuler une requête non validé par une réponse finale d'état.
- REGISTER : cette méthode est utilisée par le client pour enregistrer l'adresse dans l'URL TO par le serveur auquel il est relié. L'adresse d'enregistrement est du type Utilsateur@Domaine

3.6 Les codes d'état

Une réponse à une requête est caractérisée par un code appelé code d'état. Le tableau V suivant récapitule ces codes :

Tableau V

Récapitulatif des codes

Code	Signification
1xx	Information
2xx	Succès
3xx	Réacheminement
4xx	Erreur requêtes
5xx	Erreur serveur
6xx	Erreur globale

3.7 Description des messages SIP

Un message SIP peut être une requête d'un client vers un serveur ou bien une réponse d'un serveur vers un client. Ces deux types de messages SIP utilisent le format suivant :

Tableau VI

Message SIP

Début de ligne
En tête
CRLF
Corps du message (optionnel)

avec :

Début de ligne = ligne de requête ou ligne d'état

En-tête = *en-tête générale* ou *en-tête de requête* ou *en-tête de réponse* ou *en-tête d'entité.*

Le champs *en-tête générale* s'applique à la fois aux messages de requêtes et de réponses. Le champs *en-tête d'entité* définit le type d'informations contenues dans le corps du message. Le champs *en-tête de requête* autorise le client à ajouter des informations concernant sa requête et le champs *en-tête de réponse* autorise le serveur à ajouter des informations concernant sa réponse.
La description des 36 noms d'en-têtes est donnée en détail dans le document normalisé RFC 2543 [23].

CRLF = indique la fin de ligne et le début du corps du message (optionnel)

Corps du message = selon la méthode il indique des informations sur la progression de la requête, description de la session, description de destinations ou services intermédiaires, causes d'erreurs.

3.8 Diagramme des séquences

Dans la section précédente on a présenté le format des messages SIP échangés entre différentes entités SIP lors de la signalisation des appels. Dans cette section, on procède à la modélisation orienté objet de ce procédé. On présente le comportement dynamique du protocole SIP durant l'établissement et la fermeture d'appel.

Établissement d'appel :

Dans ce cas deux clients établissent l'appel par l'intermédiaire d'un serveur proxy en mode point à point. Le diagramme de séquences suivant illustre ce cas de figure :

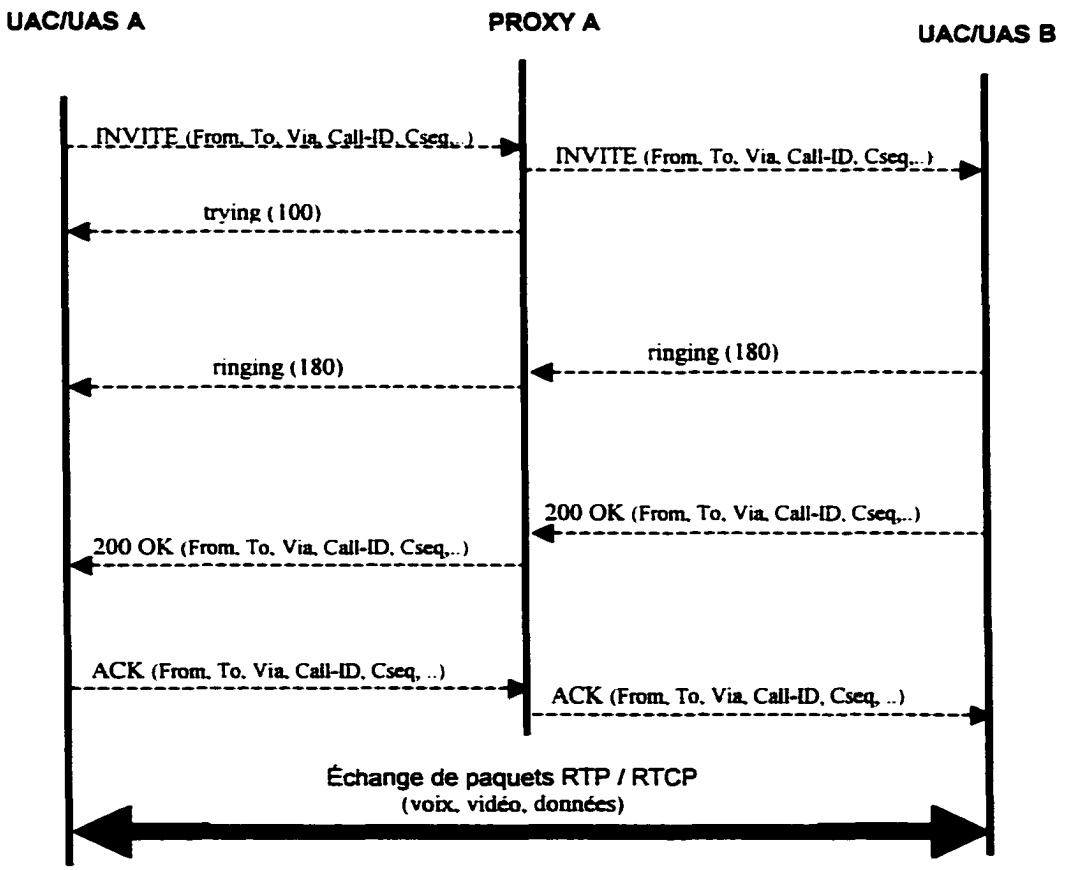


Figure 10 Établissement d'appel

Terminaison d'appel :

La figure 11 présente la fermeture de l'appel. L'entité, qui désire mettre fin à cet appel, envoie un message BYE et l'autre répond par un OK.

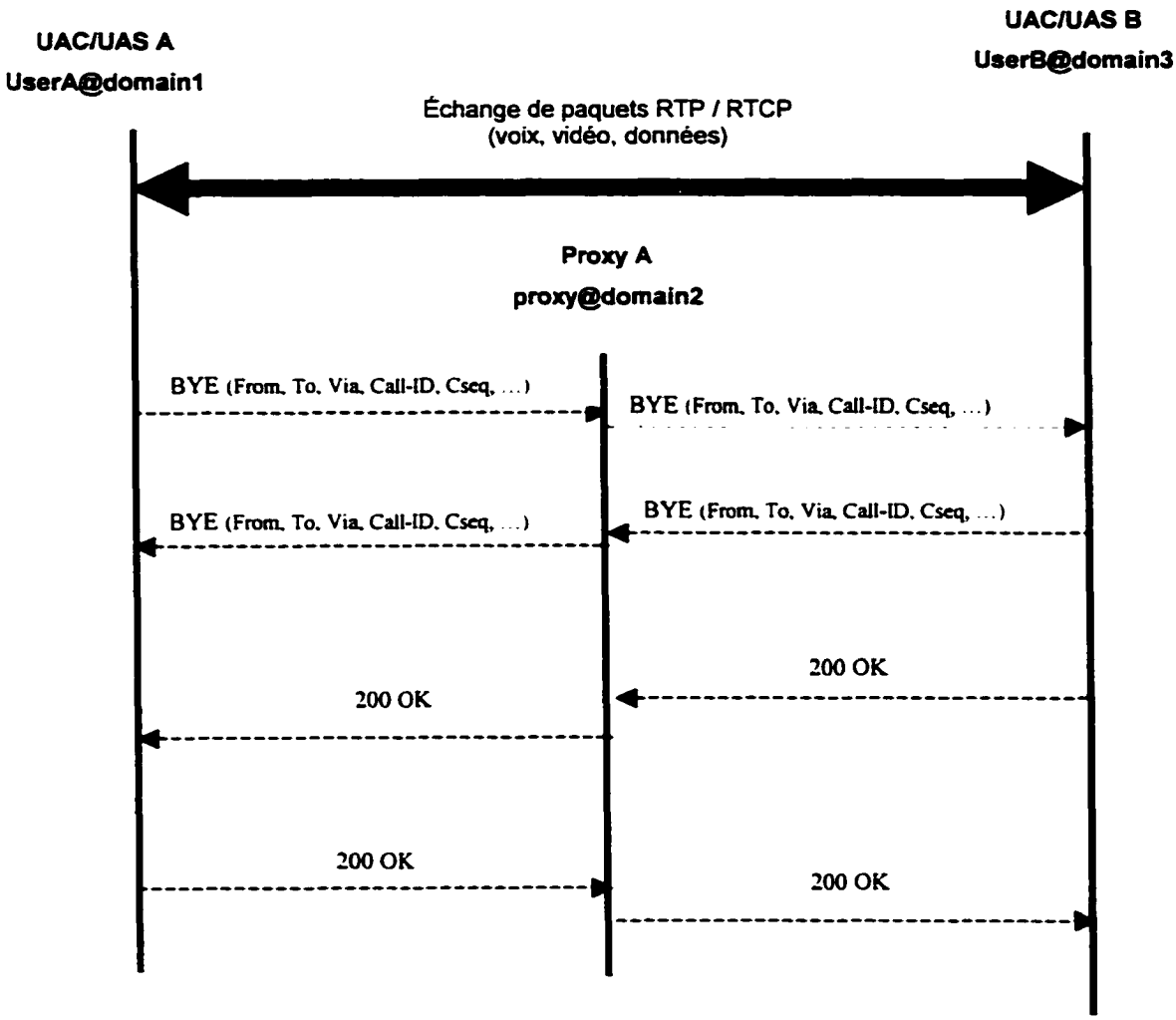


Figure 11 Terminaison d'appel

3.9 Discussion

Nous avons choisi d'utiliser le protocole SIP pour ses caractéristiques suivantes :

- **Fiabilité :** SIP peut fonctionner aussi bien avec TCP qu'avec UDP. Vu que UDP est un protocole non fiable, SIP doit fournir sa propre fiabilité faite de retransmission. Le protocole TCP par contre est fiable; une fois que la connexion est établie les données arriveront à destination.

- **Services :** SIP permet des services similaires aux services de la téléphonie classique. Il fournit les éléments de construction de ces services à l'aide des en-têtes et des méthodes. Parmi ses services :
  - *Le transfert d'appel* : permet à un utilisateur de transférer un appel établi à une tierce partie. SIP le réalise par l'intermédiaire de l'en-tête *Contact* .
  - *Le transfert de communication* : permet à l'utilisateur appelé de faire suivre un appel à une autre adresse en utilisant les en-têtes *Contact, Call-ID*...
  - *Invitation d'un groupe :* Contrairement aux autres protocoles de signalisations, SIP permet des appels qui atteignent une personne ou bien un groupe de personnes. Ceci est rendu possible grâce à l'utilisation d'UDP qui permet le multicast.
  - *Mobilité de personnes :* La mobilité d'une personne correspond à la capacité de créer et de recevoir un appel à n'importe quel terminal et dans n'importe quel site. SIP permet de joindre la personne appelée, même en cas de changement de terminal. Ceci est possible grâce au serveur de redirection dont le rôle est de rediriger les appels vers une autre destination que la personne appelée a choisi.

- **Flexibilité Topologique :** SIP permet des conférences en mode multicast, multipoint pleinement interconnectés ou bien en mode multipoints via un MCU.

De même, ISIP et H323 sont conçus pour répondre aux besoins de contrôle de session et de fonctions de signalisation dans une architecture de contrôle d'appel distribuée.

Dans ce qui suit nous allons comparer ces deux protocoles par rapport à certains critères [30]:

1. Simplicité

H.323 utilise une représentation pour ses messages, basée sur le standard ASN.1 ce qui complique les procédures d'analyse des messages. En revanche, SIP code ses messages sous forme textuel (comme HTTP). Ce format textuel tend à faciliter la mise au point et l'analyse.

2. Délai d'initiation d'appel

Dépendamment si le garde barrière est utilisé ou non, établir un appel H.323 nécessite l'échange de plusieurs paquets et un intervalle de temps de 6 à 7 RTT (Round-trip times). En revanche, pour établir un appel SIP via UDP cela prend 1.5 RTT et un échange de 4 paquets seulement (INVITE, 200 OK,ACK, BYE).

3. Neutralité du protocole de transport

SIP peut utiliser aussi bien TCP que UDP, incluant ATM,AAL5,IPX,X.25 sans changement au protocole. H.323 nécessite un protocole de transport fiable.

4. Du fait de l'utilisation de UDP, SIP peut facilement envoyer les requêtes via le multicast. H.323 a une spécification différente (appelée H.332) pour supporter le multicast.

5. Complexité

Les spécifications SIP sont nettement plus courtes que les spécifications Q.931 et H.245 combinées [30].

6. Description de contenu

H.323 utilise toujours H.245 pour négocier le type de média dans une conférence; SIP peut utiliser n'importe quel format de description. SIP ne possède pas toute la flexibilité de négociation du H.245, cependant le H245 est restreint seulement aux codecs ITU-T.

3.10 Conclusion

Dans ce chapitre nous avons étudié le protocole SIP et l'avons comparé au standard H.323. Malgré sa relative jeunesse par rapport au H.323, le protocole SIP, grâce à ses qualités, a provoqué un vif intérêt, tant dans le milieu de la recherche que dans l'industrie. Plusieurs compagnies telles que Cisco, 3Com, oeuvrent dans ce domaine pour développer des produits SIP comme des serveurs proxy, de redirection, etc.

Le protocole SIP, répond parfaitement aux besoins de notre application. Nous présenterons dans le chapitre prochain le modèle de dialogue et l'architecture de notre système.

CHAPITRE 4

CONCEPTION ET ARCHITECTURE

Dans les chapitres précédents nous avons décrit brièvement les différents protocoles, mis en jeu dans la réalisation d'une application multimédia, relatifs à la signalisation des appels et au transport des données (audio et vidéo) en temps réel. Dans ce chapitre, nous allons décrire le modèle de dialogue propre au SIP et à RTP et l'architecture système de notre application. Nous décrirons aussi les différents composants qui constituent cette architecture et leur fonctionnement.

4.1 Modèle de dialogue

SIP permet l'utilisation de deux architectures : une architecture impliquant un proxy utilisé dans le cas de postes de travail fixes et une deuxième impliquant en plus du proxy, un serveur de redirection utilisée dans le cas de terminaux mobiles.
Dans ce qui suit nous procédons à la modélisation générale d'un dialogue dans le cadre d'un SIP avec proxy qui s'adapte bien à notre application.

Notre application est de type client-serveur. Elle permet de réaliser la signalisation d'appels SIP et le transport multimédia des données voix et vidéo.

La figure 12 représente le modèle de dialogue propre au SIP et à l'échange des paquets RTP et RTCP de notre application entre deux agents utilisateurs client-serveur (*UAC/UAS*) A et B via un proxy:

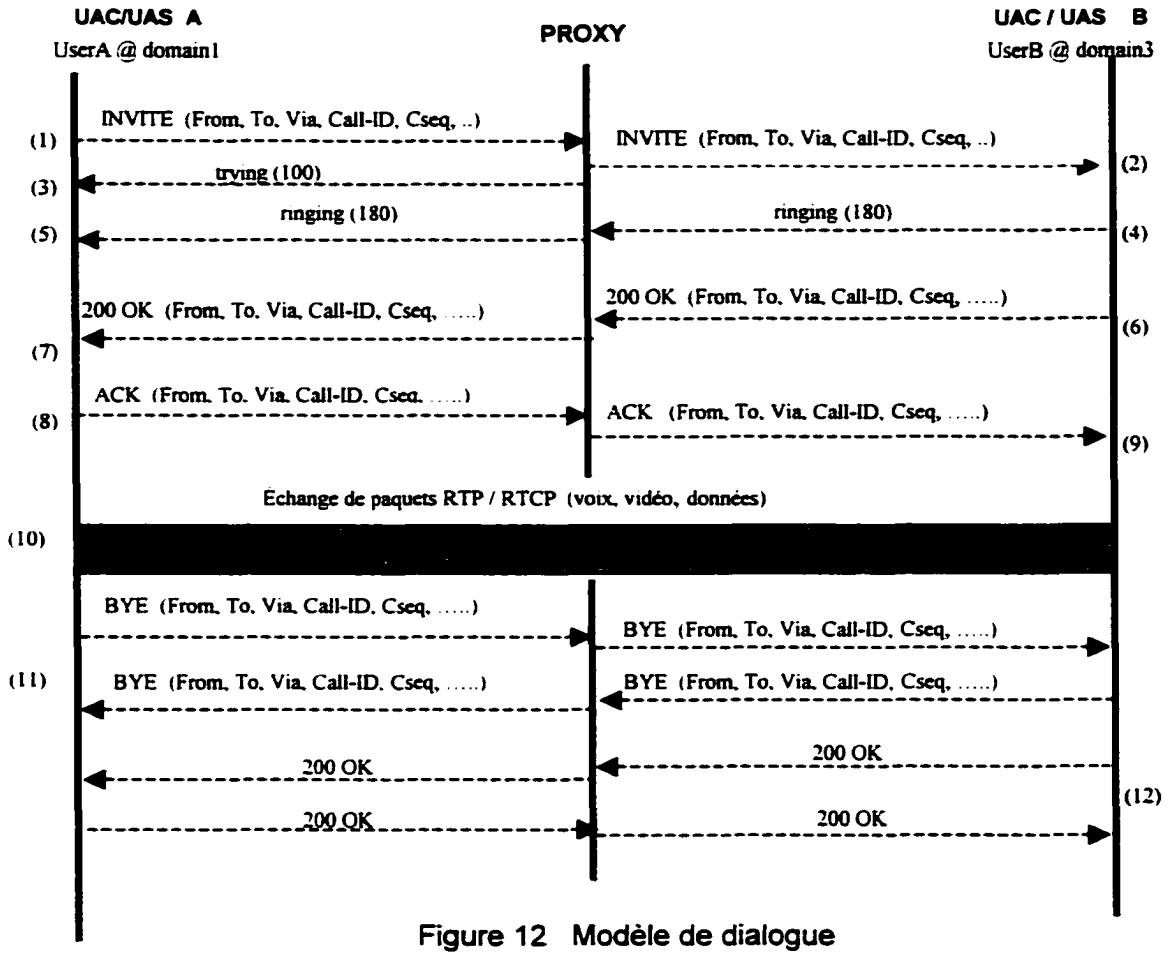


Figure 12 Modèle de dialogue

L'établissement d'une communication entre clients se fait en unicast (point à point). Les étapes successives sont les suivantes :

1. Le client A qui désire initier une communication émet une requête INVITE contenant les URL From,To et Via. Celle ci est envoyée au serveur proxy.
2. Le proxy reçoit la requête INVITE du client, il en extrait l'URL To, consulte sa base de données pour vérifier si le destinataire existe et en fonction de cette vérification il renvoie un message d'état au client source le notifiant que le destinataire n'est pas trouvé dans le cas où celui-ci n'existe pas. Dans le cas contraire il achemine la requête au destinataire à l'adresse IP contenue dans l'URL To pour signaler au destinataire une invitation à une communication.

3. En même temps que le proxy achemine la requête au destinataire, il envoie aussi un message d'état au client pour le notifier qu'il essaie d'établir un contact avec le destinataire.
4. Dès la réception de la requête INVITE par le client destinataire, un message d'invitation s'affiche à son écran lui indiquant le nom du client qui désire établir une communication. Un message d'état est envoyé de manière automatique au proxy qui l'acheminera vers le client source pour l'informer que le destinataire a reçu son invitation avec succès.
5. Le proxy reçoit la réponse du destinataire, il extrait l'URL *From* du message et de la même manière qu'à l'étape 2, il achemine celle-ci vers le client source. Un message *ringing* est affiché à l'écran de ce dernier.
6. Le client destinataire répond par une message *Ok* au client source et accepte ainsi son invitation à communiquer. Cette réponse est envoyée au proxy.
7. Le proxy reçoit la réponse Ok, il l'achemine au client source de la même manière qu'à l'étape 5.
8. À la réception de la réponse Ok, un message d'acquittement est envoyé au proxy automatiquement contenant les mêmes URL From et To que les messages échangés précédemment.
9. De la même manière qu'à l'étape 2, le proxy achemine ce message d'acquittement vers le destinataire.
10. La communication est établie dès la réception des messages d'acquittement *Ack*. A cette étape les paquets RTP et RTCP sont échangés entre les clients en multicast. Les ports RTP et RTCP sont attribués et gérés par le serveur proxy. Chaque fois qu'une communication est établie, le serveur proxy attribue à chaque client un port RTP pair et un port RTCP impair immédiatement supérieur.
11. Si un client désire mettre fin à sa participation à la conférence multimédia, il envoie une requête *Bye* au proxy qui l'achemine à son tour aux autres clients pour les informer du départ de celui-ci.
12. Les clients répondent par un message *Ok* au client qui désire terminer sa connexion via le proxy. Celui-ci libère les ports RTP et RTCP qu'il a attribué à ce client.

4.2 Architecture

La figure 13 illustre l'architecture de notre logiciel proposé et les interactions entre les différentes composantes durant le processus d'échange des requêtes SIP et de la visioconférence. Pour rendre l'architecture plus lisible et moins encombrée, nous avons représenté un cas simple de deux clients et d'un serveur proxy.

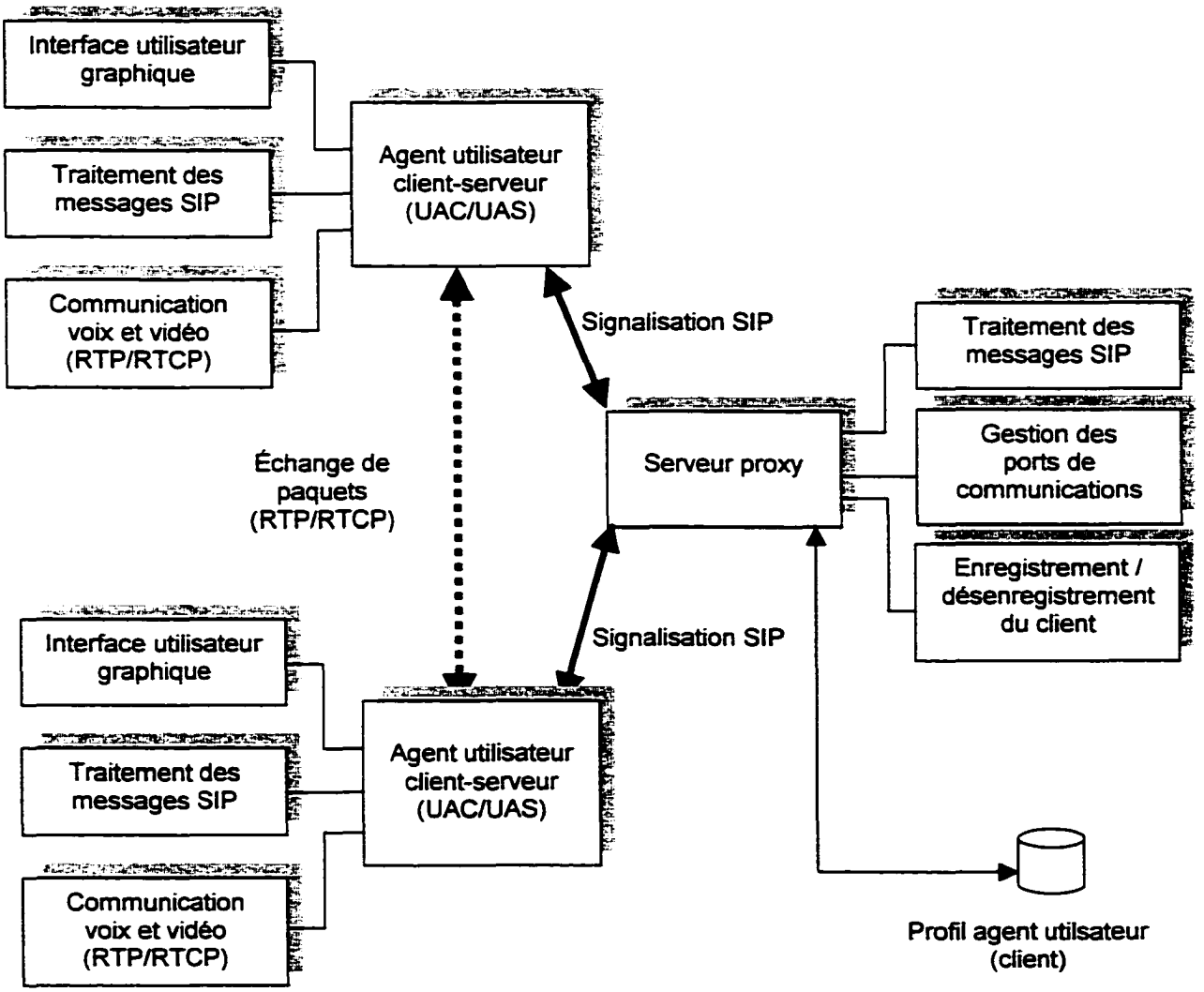


Figure 13 Architecture de l'application

4.2.1 Les composantes de l'architecture

Les étapes que nous avons suivies pour l'implémentation de notre application ont consisté à développer les différentes composantes suivantes :

4.2.1.1 Enregistrement des clients

La première étape pour participer à une session SIP est de s'enregistrer auprès du serveur registraire. Pour cela, on a crée une base de données dynamique, qui enregistre toutes les données concernant le profile usager (client) . Pour mettre à jour cette base de données nous avons jugé utile de la rafraîchir à chaque nouvel enregistrement ou désenrégistrement d'un client donné, cela nous permettra d'éviter de faire des mises à jours automatiques à chaque fois.

Un client qui désire inviter d'autres clients à une session multimédia peut avoir accès à cette liste, le serveur se charge de lui envoyer la dernière mise à jour à sa demande. Chaque client qui s'enregistre auprès du serveur se voit attribuer un numéro de port pour pouvoir communiquer avec lui. Cette liste est consultée en permanence par le proxy, pour traiter les requêtes qu'il reçoit des clients.

4.2.1.2 Agent utilisateur client-serveur

Le client SIP est conçu pour exécuter une session de conversation sur Internet. Il se charge de la création et de la terminaison d'une conférence.

En premier lieu, le client SIP interagit avec un usager à l'aide de l'interface usager qui est généralement une interface graphique (GUI). Cela permet l'invocation de requêtes à partir de boutons du fichier déroulant, à l'aide d'une souris ou du clavier. Il permet de saisir dans les champs les URL sources et destinations qui seront envoyés au proxy. L'état des échanges de requêtes SIP est affiché à tout instant. L'état « CONNECTÉ » permet au client de savoir que la conférence a été lancée. D'autres fenêtres s'affichent à l'écran montrant les participants invités à la conférence. Si un client désire quitter la

conférence, il choisit dans l'interface graphique le bouton BYE qui met fin à sa participation. Cette application client est un serveur en même temps, celui-ci traite les réponses SIP qui proviennent du proxy, et génère d'autres messages tels que le message *ringing* , et les acquittements.

4.2.1.3 Proxy

Ce serveur a été conçu pour procurer un service au client, pour analyser et interpréter des invitations de sessions basées sur le protocole SIP et pour générer des réponses aux clients. Il permet aussi aux utilisateurs de s'enregistrer dans sa liste. La figure 14 illustre les tâches successives du proxy dans un processus de signalisation SIP qui consistent en la réception des paquets UDP, l'enregistrement du client dans le cas où celui-ci se connecte pour la première fois, ensuite les paquets sont traités par le proxy pour décider de la réponse à renvoyer au client source ou leurs acheminements vers le destinataire :

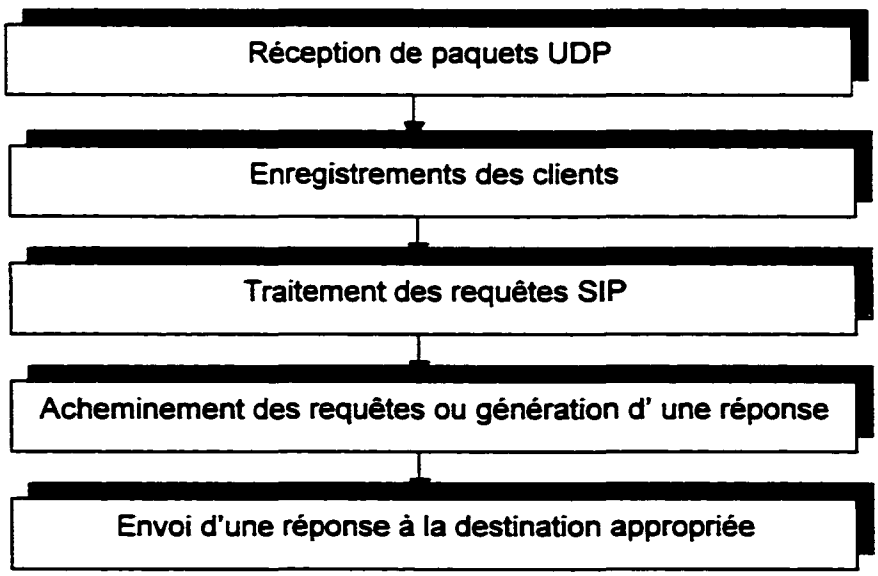


Figure 14 Tâches du serveur SIP

Ce serveur gère aussi les ports RTP et RTCP dans une communication audio et vidéo. Il attribue à chaque client des ports RTP pairs et des ports RTCP impairs immédiatement supérieurs.

4.2.1.4 Transmission et réception audio et vidéo

L'API (Java Media Framework) [20] est une interface de programmation développée par Sun dont le but est de faciliter le développement des applications multimédia en Java. Il comporte les fonctionnalités nécessaires à la capture, au stockage et au contrôle des données multimédia. Il est basé sur plusieurs éléments: Le *datasource*, le *player*, le *processor*, le *manager*...

Il fonctionne de manière similaire au système audio-visuel, de la vie courante, qui utilise des dispositifs de capture et de restitution du son et de l'image (caméra, haut-parleur, écran TV...), une cassette vidéo, et un lecteur VCR. Le *dataSource*, tout comme une cassette vidéo, est un objet qui permet d'encapsuler les flux de données multimédia et de gérer leurs transfert. Le *player* et le *processor* sont des objets dont le rôle est de fournir les mécanismes de contrôle, de capture et de restitution des données en temps réel. Le *player* permet seulement de présenter les données alors que le *processor* permet en plus d'effectuer des traitements. Ces deux éléments jouent le rôle d'un lecteur VCR. Le gestionnaire de session RTP maintient l'état de la session, des participants et les statistiques sur la transmission et la réception des données (RTCP). Le gestionnaire ou *manager* permet la construction des objets *player, processor, et dataSource* [35].

JMF fournit une architecture et un protocole d'échange pour gérer l'acquisition, le traitement et la restitution de données multimédia en temps réel. Parmi les formats disponibles, nous citons entre autres GSM, WAV, [20] pour l'audio et les formats JPEG [52], MPEG [53],AVI [20] pour la vidéo.
Dans notre application nous avons choisi d'utiliser le format GSM pour la transmission et la réception audio. Ce format présente une faible qualité de son mais nécessite moins de bande passante et de capacité CPU. Pour la vidéo nous avons choisi le

format JPEG qui présente une qualité supérieure mais qui est plus gourmand en terme de bande passante et de capacité CPU [20].

Transmission audio et vidéo :

A la transmission un objet *processor* est créé, il va lire la voix et la vidéo sur des périphériques de capture à l'aide de méthodes propres à l'API JMF et les passe au d*ataSource*. La voix et la vidéo sont alors codées, compressées, paquetisées puis passées au gestionnaire de session RTP ou *SessionManager* qui envoie les données à l'adresse indiquée [20]. La figure 15 illustre ces étapes de transmission :

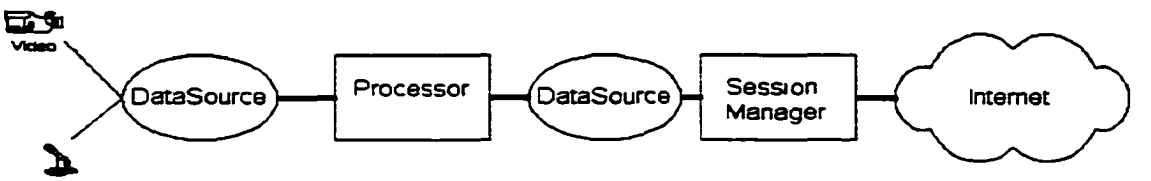


Figure 15 Transmission audio et vidéo

La procédure à suivre pour créer un canal d'émission est la suivante [20] :

- Créer, initialiser et démarrer un *sessionManager* pour la session RTP.
- Construire un *processor* utilisant le d*ataSource* de capture.
- Choisir un format de sortie du *processor* spécifique au RTP.
- Récupérer le flux de sortie du *procesor* dans un d*ataSource*
- Appeler la méthode CreateSendStream du s*essionManager* avec le *dataSource* en paramètre

Réception audio et vidéo :

A la réception un gestionnaire de session RTP est créé et se met en attente. Lorsqu'il détecte les premières données reçues du réseau IP, il crée un objet *player* de

l'interface JMF qui dépaquetise, décompresse et décode les données, puis les passe au d*ataSource*.

C'est le *player* qui se charge de restituer le contenu du flux entrant. Pour recevoir et restituer un flux provenant d'une session RTP, il faut utiliser un objet *MédiaLocator* qui spécifie les paramètres de construction du *player* [20]. La figure 16 montre les étapes de réception :

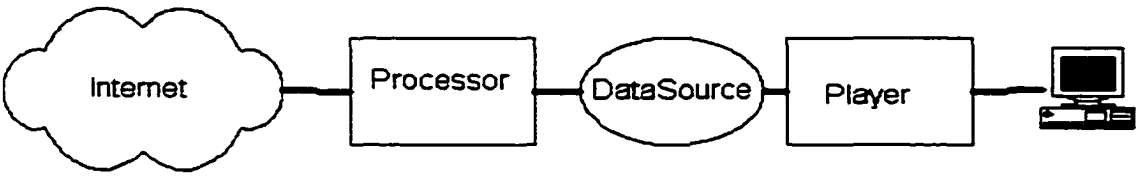


Figure 16 Réception audio et vidéo

La procédure à suivre pour la réception de flux de données est la suivante [20] :

- Créer un objet *sessionManager*
- Appeler la méthode *addReceiveStreamListener* pour s'identifier comme listener
- Initialiser la session RTP
- Commencer la session
- Appeler la méthode *update* de la classe *ReceiveSreamListener* qui indique qu'un nouveau flux de données est recu grâce à NewReceiveStreamEvent et appeler la méthode *getReceiveStream* qui cherche ce nouveau flux.
- Appeler la méthode *getDataSource* sur ce *ReceiveStream*.
- Créer un player avec comme paramètre le *DataSource*.

4.3 Conclusion

Pour mettre en œuvre l'architecture qui a été présenté dans ce chapitre, nous avons implanté notre application avec une approche orientée objet en utilisant le langage Java parce qu'il est simple et complet. De plus, Sun microsystem a développé l'API JMF (Java Mdia Framework) qui inclut les protocoles RTP et RTCP et facilite l'implémentation de la partie transport des données.

Pour cela, nous avons utilisé un kit de développement JDK1.3 de java. L'application a été testée sur les machines Pentium 3 avec 128 Méga-octets de mémoire vive mis à notre disposition, avec le système d'exploitation Windows 2000. Pour la partie conférence voix et vidéo, du matériel multimédia a été mis à notre disposition qui consiste en microphones, haut-parleurs et caméras (*webcam*). Certaines de ces caméras sont supportées par l'API JMF, et ce qu'on a trouvé sur le marché est la *webcam* de la compagnie Logitech (*QuickCam Express* et *QuickCam Home*).

Dans le prochain chapitre nous présentons en détail cette mise en œuvre et les tests qui ont été effectués sur cette application.

CHAPITRE 5

MISE EN ŒUVRE ET TESTS

Dans le chapitre précédent, nous avons présenté le modèle de dialogue et l'architecture générale de notre application. Dans ce chapitre nous allons décrire de manière général le modèle client-serveur en Java qui constitue la base de notre architecture répartie SIP. Nous détaillerons aussi les différentes classes java qui constituent le programme de notre application. Enfin, nous présenterons les résultats et tests effectués.

5.1 Le langage Java

Le langage Java est un langage de programmation orienté objet mis au point par Sun Microsystems est basé sur la réutilisabilité et la simplicité de mise en œuvre. Java permet à la fois de créer des programmes classiques et des applets intégrables à des pages Web. L'applet est téléchargé et exécuté par le navigateur web, offrant ainsi une dimension dynamique à la page HTML.

Les caractéristiques du langage Java sont :

- Il est orienté objets
- Il Intègre l'encapsulation, l'héritage, la liaison dynamique, possède d'origine une librairie de classes standards (qui intègre notamment des classes gérant l'interface graphique utilisateur)
- Il facilite la portabilité car il est indépendant de la machine sur laquelle on exécute le programme
- Une source Java pré-compilée donne des "bytes codes",une sorte de pseudo-assembleur adresse à la machine virtuelle Java (JVM). Cette JVM interprète ensuite ce byte code pour exécuter le programme. C'est le byte code qui est portable ; le tout est donc d'écrire un JVM propre à la machine. Les logiciels de navigation compatibles Java intègrent l'interpréteur de bytes codes qui simule cette machine virtuelle. Les applications ou applets Java, sous forme de bytes codes, sont donc indépendantes de la machine physique.

- Il permet la programmation multi-fils. Un programme Java peut lancer plusieurs fils qui s'exécutent simultanément. Java gère aussi le mécanisme des moniteurs, qui permet de synchroniser les fils.
- Il intègre dès la conception plusieurs mécanismes de sécurité, visant à rendre les programmes fiables et à éliminer les risques de virus.

5.2 Java et le modèle client-serveur

L'une des forces de Java est de pouvoir travailler sans peine en réseau. Les concepteurs de la bibliothèque de Java en ont fait quelque chose d'équivalent à la lecture et l'écriture de fichiers, avec la différence que les fichiers résident sur une machine distante et que c'est elle qui décide de ce qu'elle doit faire au sujet des informations qui lui sont demandées ou envoyées.

Java utilise des sockets qui sont des abstractions de programmation représentant les extrémités d'une connexion entre deux machines. Java fournit des sockets de séquences d'octets (stream) et des sockets de datagrammes. Dans notre application nous avons choisi d'utiliser le protocole de transport non fiable UDP, bien qu'il soit aussi possible d'utiliser TCP pour les échanges SIP. Nous avons donc utilisé les sockets de datagramme, avec lesquelles il est possible de transmettre des paquets individuels de données. La mise en réseau est illustrée dans les classes client et serveur de la section suivante.

5.3 Architecture des hiérarchies de classes

La figure 17 illustre les principales classes de note application :

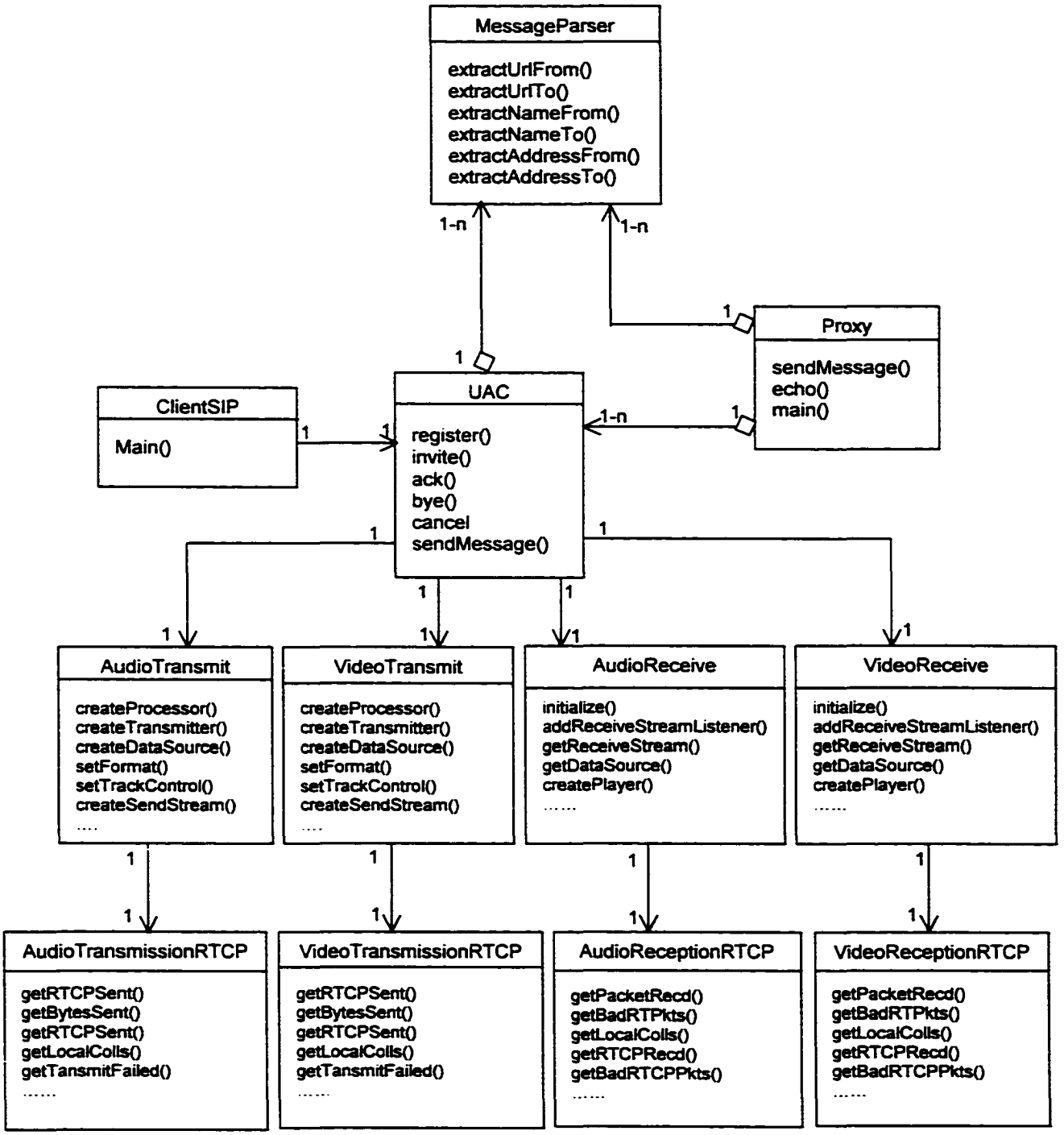


Figure 17 Principales classes

5.3.1 Identification des classes

Dans la figure 17, nous avons représenté toutes les classes de notre application. La partie signalisation SIP est représentée par les classes ClientSIP, UAC, Proxy et MessageParser. La partie RTP qui concerne l'échange de données voix et vidéo est représentée par les classes AudioTransmit, VideoTansmit, AudioReceive, et VideoReceive. La partie contrôle RTCP est représentée par les classes AudioTransmissionRTCP, VideoTransmissionRTCP, AudioReceptionRTCP et VideoReceptionRTCP. Dans les deux dernière partie RTP et RTCP nous avons utilisé les méthodes de l'API JMF pour implanter cette application.
Dans cette section nous allons décrire chaque classe avec ses principales composantes :

5.3.1.1 Classe UAC

La classe UAC est la partie cliente de l'application SIP et est utilisée pour créer et envoyer des requêtes SIP échangées avec le serveur proxy. Elle permet l'ouverture d'une interface graphique (GUI) qui interagit avec le serveur. Elle contient les méthodes SIP qui permettent d'initier et de fermer une session multimédia. Parmi ces principales méthodes, on a :

invite()
Cette méthode crée une requête INVITE lors d'un processus de signalisation d'appel et l'envoie au serveur. Elle contient l'information dans l'en-tête SIP, qui identifie entre autres, l'entité appelée et appelante, le Call-ID, le numéro de séquence. En fait, elle nous informe qu'un appel est entrain de s'initier.

ack()
Cette méthode indique que l'entité appelante acquitte une requête INVITE qui a été acceptée avec succès par le code 200. ACK indique que l'appelant a reçu une confirmation à une requête INVITE .

bye()

Cette méthode indique qu'un client qui participe à une conférence veut mettre fin à une communication. Le flux de données est terminé quelque soit la réponse de l'autre entité.

cancel()

Cette méthode annule une requête en cours, mais n'a aucun effet sur l'appel déjà établi quand aucune requête n'est en cours.

register()

Cette méthode est utilisé par un UAC pour s'enregistrer auprès d'un serveur SIP. L'UAC doit s'enregistrer avec le Registrar lors du démarrage en envoyant une requête REGISTER.

sendMessage(String IPDest, int PortDest, String message)

Cette méthode utilise les sockets pour envoyer les paquets de messages d'une station vers une autre ayant pour adresse IP IPDest et port PortDest. Elle est sollicité par toutes les méthodes citées précédemment.

```
static void register()
 {
                        String message = "Register "+ProxyDomain+"Via:"+
              via+  "From: "+ from +    "To: "+from+Call_ID+"CSeq:
   "+n+ "REGISTER";
   Sendmessage(message ) ;
 }
static void invite()
{
                    String message = "INVITE " + ProxyDomain+"Via: " + via
         +"From: "+ from+"To:    "+Dest.getText()+CallID+"CSeq: "+n
         +"INVITE";
```

```
        Sendmessage(message ) ;
        }
        static void ok()
        {
         String  message = "OK "+ ProxyDomain+"Via: "+via+"From: "+from+   "To: "
                                    +inviteFrom+CallID+"CSeq: "+n+" OK" ;


     if(inviteFrom != null) {
     Sendmessage(message) ;
     inviteFrom= null; }
     else  {
              JOptionPane.showMessageDialog(null, "Client not found, check the URL
              please!", " Error!",JOptionPane.PLAIN_MESSAGE);
              }
 }

static void bye()
 {
     String message = "BYE "+ ProxyDomain+"Via: "+via+"From: "+from+"To: "+to
     +CallID+"CSeq: "+n+" BYE" ;
     Sendmessage(message) ;
 }
static void Sendmessage(String messageToSend )
{
   try
     {
       byte  Data[] = messageToSend.getBytes();
       sendPacket = new  DatagramPacket(Data,Data.length,
       InetAddress.getByName(AddressProxy), numPort);
       socketClient.send(sendPacket);
       }
```

```
        byte  Data[] = messageToSend.getBytes();
            sendPacket = new  DatagramPacket(Data,  Data.length,
                                        receivedPacket.getAddress(),
                                        receivedPacket.getPort() );
        socketClient.send(sendPacket);
        }
        catch (IOException io) {}
}
```

5.3.1.2 Classe Proxy

La classe proxy est la partie serveur de l'application SIP. Elle prend en charge les requêtes SIP envoyées par tous les clients et les traite selon les méthodes qui sont incluses dans ces requêtes. Cette classe reste en permanence à l'écoute de tous les messages qui parviennent à son port. Les requêtes sont soit acheminées vers leurs destinations appropriées ou bien une réponse est générée et renvoyée au client source. Elle dresse aussi une liste de tous les clients qui sont enregistrés et les garde en permanence en mémoire. Cette liste est rafraîchie à chaque nouveau enregistrement ou désenrégistrement d'un client donnée.

Tout client qui s'enregistre pour la première fois se voit attribuer un numéro de port qui lui permet de communiquer avec le serveur et que cette classe garde en mémoire. Ainsi à chaque requête qui parvient au proxy et qui est destiné à un autres destinataire, la classe proxy fait la correspondance avec le numéro de port qui lui a été attribué et l'achemine à sa destination. Dans cette classe on cite deux principales méthodes :

sendMessage(String IPDest, int PortDest, String message)

Cette méthode envoie le message vers la destination ayant l'adresse IP IPDest et le port PortDest.

echo(String message)

Cette méthode retourne un message vers l'adresse IP et le port reçus dans le dernier paquets reçu.

5.3.1.3 Classe MessageParser

La classe *MessageParser* est sollicitée par les classes Client et Proxy à chaque fois qu'une requête SIP est parvenue pour être traiter. Cette classe extrait les URL, noms et adresses SIP des messages reçus. Dans ce qui suit, nous allons énumérer ces quelques méthodes :

extractUrlFrom()

Cette méthode extrait l'url du client source et le retourne à la classe. Le proxy se sert de l'url source pour lui faire correspondre le port pour lui renvoyer les requêtes ou les réponses SIP.

extractUrlTo()

Cette méthode extrait l'url du destinataire. L'url To contient le nom et l'adresse du client de destination que le proxy utilise dans le traitement de la requête. Cette url sert au serveur pour faire la correspondance avec le numéro de port auquel est destiné le message

extractNameFrom()

Cette méthode extrait le nom du destinataire et le retourne à la classe. Ce nom sert à identifier le client source et à l'afficher à l'écran du destinataire pour l'informer de l'identité du client qui l'invite.

extractNameTo()

Cette méthode extrait le nom du destinataire et le retourne à la classe.

extractAddressFrom()

Cette méthode extrait l'adresse IP de l'appelant et la retourne à la classe. L'adresse source est utilisée avec le port approprié pour acheminer les messages à la bonne destination.

extractAdressTo()

Cette méthode extrait l'adresse IP du destinataire et la retourne à la classe.

Les principales méthodes de la classe *MessageParser* sont illustrés dans la partie du programme suivant :

```
static public String extractUrlFrom(String message)
{
    String messageFrom_ = message.substring(message.indexOf("From:
                                        ")+6,message.indexOf("To"));
    String messageFrom = messageFrom_.substring(0,messageFrom_.indexOf("\n"));
    String UrlFrom = messageFrom;
     return UrlFrom;
}
static public String extractNameFrom(String message)
{
     String messageFrom =
          message.substring(message.indexOf("From:")+6,message.indexOf("To"));
      String nameFrom =
          messageFrom.substring(messageFrom.indexOf("sip")+4,messageFrom.index
          Of("@"));
      return nameFrom;
}
static public String extractAddressFrom(String message)
{
     String messageFrom = message.substring(message.indexOf("From: )
                                +6,message.indexOf("To"));
```

```
        String AddressFrom =
        messageFrom.substring(messageFrom.indexOf("@")+1,messageFrom.indexOf("\n'
        ));
        return AddressFrom;
    }
    static public String extractUrlTo(String message)
    {
        String messageTo_ = message.substring(message.indexOf("To: ")
                                +4,message.indexOf("Call-ID: "));
        String messageTo = messageTo_.substring(0,messageTo_.indexOf("\n"));
        String UrlTo = messageTo;
        return UrlTo;
    }
     static public String extractNameTo(String message)
     {
         String messageTo_ = message.substring(message.indexOf("To: ")
                                +4,message.indexOf("Call-ID: "));
         String messageTo = messageTo_.substring(0,messageTo_.indexOf("\n"));
        String UrlTo = messageTo_;
        String nameTo = UrlTo.substring(UrlTo.indexOf("sip:")+4,UrlTo.indexOf("@"));
        return nameTo;
    }
    static public String extractAddressTo(String message)
    {
        String messageTo_ = message.substring(message.indexOf("To: ")
                                +4,message.indexOf("Call-ID: "));
        String messageTo = messageTo_.substring(0,messageTo_.indexOf("\n"));
         String UrlTo = messageTo_;
         String AddressTo = UrlTo.substring(UrlTo.indexOf("@")+1);
```

```
        // String messageTo =
        message.substring(message.indexOf("To:")+3,message.indexOf("Call-ID: "));
        // String messageTo =
        message.substring(message.indexOf("To:")+3,message.indexOf("\n"));
        // String AddressTo =
        messageTo.substring(messageTo.indexOf("@")+1,messageTo.indexOf("\n"));
        return AddressTo;
    }
```

5.3.1.3 Classe AudioTransmit et VideoTransmit

Les classes *AudioTransmit* et *VideoTransmit* sont des classes de transmission de la voix et de la vidéo et utilise l'API JMF, développé par Sun. Nous allons citer quelques méthodes de ces deux classes

createProcessor()
Cette méthode créé un processor qui présente les données à transmettre.

createTransmitter()
Cette méthode crée une session RTP pour transmettre la sortie du processor vers les adresses IP et ports spécifiques.

createDataSource(MediaLocator locator)
Cette méthode crée un DataSource qui va encapsuler, gérer la capture, présenter et traiter les média des données à transmettre.

setFormat(Format format)
Cette méthode est appelée pour fixer le format du média à transmettre.

La partie du programme suivante illustre l'utilisation de ces méthodes de JMF pour la transmission de la vidéo (audio) :

```
public synchronized String start()
{
    String result;
    result = createProcessor();
    if (result != null)
    return result;
    result = createTransmitter();
    if (result != null)
    {
            processor.close();
            processor = null;
            return result;
        }
processor.start();
return null;
}
/*Stops the transmission if already started */
public void stop()
{
synchronized (this)
     {
      if (processor != null)
         {
            processor.stop();
            processor.close();
            processor = null;
```

```
            for (int i = 0; i < rtpMgrs.length; i++)
                {
                 rtpMgrs[i].removeTargets( "Session ended.");
                 rtpMgrs[i].dispose();
                 }
            }
     }
    }
    private String createProcessor()
    {
        if (locator == null)
         return "Locator is null";
        DataSource ds;
        DataSource clone;
    try
    {
      ds = javax.media.Manager.createDataSource(locator);
    } catch (Exception e)
         {
        return "Couldn't create DataSource";
        }
     try
        {
          processor = javax.media.Manager.createProcessor(ds);
        } catch (NoProcessorException npe)
              {
                return "Couldn't create processor";
              } catch (IOException ioe)
             {
                return "IOException creating processor";
        }
```

```
                    // Wait for it to configure
                boolean result = waitForState(processor, Processor.Configured);
                if (result == false)
                return "Couldn't configure processor";
// Get the tracks from the processor
TrackControl [] tracks = processor.getTrackControls();
if (tracks == null || tracks.length < 1)
return "Couldn't find tracks in processor";
ContentDescriptor cd = new ContentDescriptor(ContentDescriptor.RAW_RTP);
    processor.setContentDescriptor(cd);
    Format supported[];
    Format chosen;
    boolean atLeastOneTrack = false;
    for (int i = 0; i < tracks.length; i++)
{
    Format format = tracks[i].getFormat();
    if (tracks[i].isEnabled())
     {
    supported = tracks[i].getSupportedFormats();
    if (supported.length > 0)
    {
    if (supported[0] instanceof VideoFormat)
    {
    chosen = supported[0];
    tracks[i].setFormat(chosen);
    System.err.println("Track " + i + " is set to transmit as:");
    System.err.println("  " + chosen);
    atLeastOneTrack = true;
    } else
            tracks[i].setEnabled(false);
```

```
            } else
                tracks[i].setEnabled(false);
        }
        if (!atLeastOneTrack)
            return "Couldn't set any of the tracks to a valid RTP format";
        result = waitForState(processor, Controller.Realized);
        if (result == false)
            return "Couldn't realize processor";
    dataOutput = processor.getDataOutput();
    return null;
    }
```

5.3.1.3 Classe AudioReceive et VideoReceive

L'API JMF permet la transmission et la réception de flux RTP grâce aux bibliothèques javax.media.rtp, javax.media.rtp.event et java.media.rtp.rtcp. Les principales méthodes de ces deux classes sont citées ci-dessous :

initialize()

Cette méthode crée une sessionManager et ouvre des sessions RTP.

addReceiveStreamListener()

Cette méthode est appelée pour permettre au gestionnaire de session de s'identifier comme *listener*.

getReceiveStream()

Cette méthode cherche les nouveaux flux de données qui arrivent.

getDataSource()

Cette méthode appelée sur la variable stream de ReceiveStream affecte le résultat au *dataSource* qui servira pour créer un *player*.

createPlayer(ds)

Cette méthode crée un objet *player* avec comme paramètre *datasource ds*. Ce *player* est responsable de la présentation des données.

setTrackControls()

Cette méthode garde trace du *processor*.

createSendStream()

Cette méthode crée un objet *sendStream* pour indiquer au gestionnaire de session où chercher la donnée à transmettre.

La partie du programme suivante illustre l'utilisation de ces méthodes de JMF pour la transmission de la vidéo (audio) :

```
protected boolean initialize()
{
try {
    InetAddress ipAddr;
    SessionAddress localAddr = new SessionAddress();
    SessionAddress destAddr;
    mgrs = new RTPManager[sessions.length];
    playerWindows = new Vector();
    SessionLabel session;
    // Open the RTP sessions.
    for (int i = 0; i < sessions.length; i++) {
    // Parse the session addresses.
```

```
try {
    session = new SessionLabel(sessions[i]);
    } catch (IllegalArgumentException e)
      {
        System.err.println("Failed to parse the session address given: " + sessions[i]);
        return false;
      }

    System.err.println("  - Open RTP session for: addr: " + session.addr + " port: " +
    session.port + " ttl: " + session.ttl);
    mgrs[i] = (RTPManager) RTPManager.newInstance();
    mgrs[i].addSessionListener(this);
    mgrs[i].addReceiveStreamListener(this);
    ipAddr = InetAddress.getByName(session.addr);
    if( ipAddr.isMulticastAddress())
     {
      localAddr= new SessionAddress( ipAddr, session.port, session.ttl);
      destAddr = new SessionAddress( ipAddr,session.port,session.ttl);
      } else {
            localAddr= new SessionAddress( InetAddress.getLocalHost(), session.port);

            destAddr = new SessionAddress( ipAddr, session.port);
            }
      mgrs[i].initialize( localAddr);
      BufferControl bc =
      BufferControl)mgrs[i].getControl("javax.media.control.BufferControl");
      if (bc != null)
      bc.setBufferLength(350);
      mgrs[i].addTarget(destAddr);
 }
 }
```

```
catch (Exception e){
                    System.err.println("Cannot create the RTP Session: " +
                    e.getMessage());
                    return false;
                  }
// Wait for data to arrive before moving on.
long then = System.currentTimeMillis();
// wait for a maximum of 30 secs.
long waitingPeriod = 30000;
try{
    synchronized (dataSync)
     {
        while (!dataReceived &&System.currentTimeMillis() - then < waitingPeriod)
        {
        if (!dataReceived)
        System.err.println(" - Waiting for RTP data to arrive...");
        dataSync.wait(1000);
        }
      }
  } catch (Exception e) {}
if (!dataReceived) {
System.err.println("No RTP data was received.");
close();
return false;
}
return true;
}
public boolean isDone() {
return playerWindows.size() == 0;
}
```

```
            try {
                ((PlayerWindow)playerWindows.elementAt(i)).close();
                } catch (Exception e) {}
        }
    playerWindows.removeAllElements();
    // close the RTP session.
    for (int i = 0; i < mgrs.length; i++)
     {
      if (mgrs[i] != null)
      {
        mgrs[i].removeTargets( "Closing session ");
        mgrs[i].dispose();
        mgrs[i] = null;
      }
     }
  }
```

5.3.1.4 Classe AudioTransmissionRTCP et VidéoTransmissionRTCP

Nous utilisons les statistiques RTCP fournies par l'interface JMF pour évaluer la performance de la communication. En effet, La *sessionManager* maintient une trace de toutes les statistiques RTP et RTCP des paquets envoyés et reçues dans une session. Les deux classes *AudioTransmissionRTCP* et *VideoTransmissionRTCP* permettent d'avoir des rapports de statistiques des paquets RTCP transmis à chaque instant, et utilisent des méthodes définies dans la classe *GlobalTransmissionStats* de l'API JMF. Ces deux classes sont instantiées à partir des classes *AudioTransmit* et *VideoTransmit* [20]. Les méthodes qui sont définis sont :

getRTPSent()

Cette méthode retourne le nombre de paquets RTP envoyés.

getBytesSent()

Cette méthode retourne le nombre d'octets RTP envoyés par le transmetteur.

getRTCPSent()

Cette méthode retourne le nombre de paquets RTCP envoyés.

getLocalColls()

Cette méthode retourne le nombre de collisions de paquets RTP correspondant au cas où deux éléments différents de la session génèrent un même SSRC ou un même CNAME.

getTransmitFailed()

Cette méthode retourne le nombre d'échec de transmissions.

La partie du programme suivante illustre l'utilisation de ces méthodes pour afficher le rapport de statistiques de transmission de la vidéo (audio) :

```
private void updateFields ()
 {
    int  i;
    int  nCount;
    Object  objMngr;
    SessionManager  mngrSession;
    GlobalTransmissionStats  stats;
    nCount = sessionManager.length;
    for ( i = 0;  i < nCount;  i++ )
     {
       objMngr = sessionManager[i] ;
       if ( !(objMngr instanceof SessionManager) )
          continue;
```

```
            mngrSession = (SessionManager)objMngr;
            stats = mngrSession.getGlobalTransmissionStats ();
            fieldTotalRtpPacketsSent.setText ( "" + stats.getRTPSent() );
            fieldTotalBytesSent.setText ( "" + stats.getBytesSent() );
            fieldRtcpPacketsSent.setText ( "" + stats.getRTCPSent() );
            fieldLocalCollisions.setText ( "" + stats.getLocalColls() );
            fieldRemoteCollisions.setText ( "" + stats.getRemoteColls() );
            fieldFailedTransmissions.setText ( "" + stats.getTransmitFailed() );
        }
    }
```

5.3.1.5 Classe AudioReceptionRTCP et VideoReceptionRTCP

De la même manière que les classes de statistiques de transmission, les classes *AudioReceptionRTCP* et *VideoReceptionRTCP* permettent d'avoir les statistiques de réception de paquets RTP et utilisent la classe *GlobalReceptionStats* .
Ces méthodes sont :

getPacketsRecd()
Cette méthode retourne le nombre de paquets reçus.

getBadRTPkts()
Cette méthode retourne le nombre de mauvais paquets RTP reçus.

getLocalColls()
Cette méthode détecte le nombre de collisions de paquets RTP (paquets différents ayant un même SSRC ou un même CNAME).

getRTCPRecd()
Cette méthode retourne le nombre de paquets RTCP reçus.

getSRRecd()

Cette méthode retourne le nombre de paquets SR reçus.

getBadRTCPPkts()

Cette méthode retourne le nombre de mauvais paquets RTCP reçus.

La partie suivante de notre programme montre comment ces méthodes sont utilisés :

```
private void updateFields ()
{
    GlobalReceptionStats    stats;
    stats = sessionManager.getGlobalReceptionStats ();
    fieldTotalRtpPackets.setText ( "" + stats.getPacketsRecd() );
    fieldTotalBytes.setText ( "" + stats.getBytesRecd() );
    fieldBadRtpPackets.setText ( "" + stats.getBadRTPkts() );
    fieldLocalCollisions.setText ( "" + stats.getLocalColls() );
    fieldRemoteCollisions.setText ( "" + stats.getRemoteColls() );
    fieldPacketsLooped.setText ( "" + stats.getPacketsLooped() );
    fieldFailedTransmissions.setText ( "" + stats.getTransmitFailed() );
    fieldRtcpPackets.setText ( "" + stats.getRTCPRecd() );
    fieldSrPackets.setText ( "" + stats.getSRRecd() );

    fieldBadRtcpPackets.setText ( "" + stats.getBadRTCPPkts() );
    fieldUnknownRtcpTypes.setText ( "" + stats.getUnknownTypes() );
    fieldMalformedRr.setText ( "" + stats.getMalformedRR() );
    fieldMalformedSdes.setText ( "" + stats.getMalformedSDES() );
    fieldMalformedBye.setText ( "" + stats.getMalformedBye() );
    fieldMalformedSr.setText ( "" + stats.getMalformedSR() );
}
```

5.4 Tests et résultats

Dans cette section, nous présentons les différents résultats générés durant la phase des tests de notre plate-forme. Dans l'implémentation de la partie signalisation, nous avons choisi d'utiliser le protocole de transport UDP et le port 5060. Ce dernier est le port par défaut utilisé dans le protocole SIP, bien qu'il nous était possible de choisir tout autre port public.

5.4.1 Initiation d'appels

Lorsque nous lançons l'application client, une boite de dialogue (Figure 18) apparaît à l'écran. Cette fenêtre permet à l'utilisateur de saisir son adresse SIP (pour s'identifier dans une session), de saisir aussi l'adresse SIP du destinataire et de connaître l'état de connexion SIP en tout instant (ringing, trying. Not found, connection established ...) . A gauche de l'écran apparaît un menu défilant qui permet à l'utilisateur de choisir l'un des boutons. Une action sur l'un de ces boutons, enclenche un processus de création de la requête en lisant les url source et destination, saisis dans la boite de dialogue, constituant le message de la requête SIP approprié et l'envoyant au serveur proxy qui s'en chargera de son traitement. Nous avons aussi rajouté un bouton « Get List » qui ne fait pas partie du protocole SIP, et qui permet à l'utilisateur de demander au serveur de lui envoyer la dernière mise à jour de tous les clients qui font partie de la session à ce moment précis.

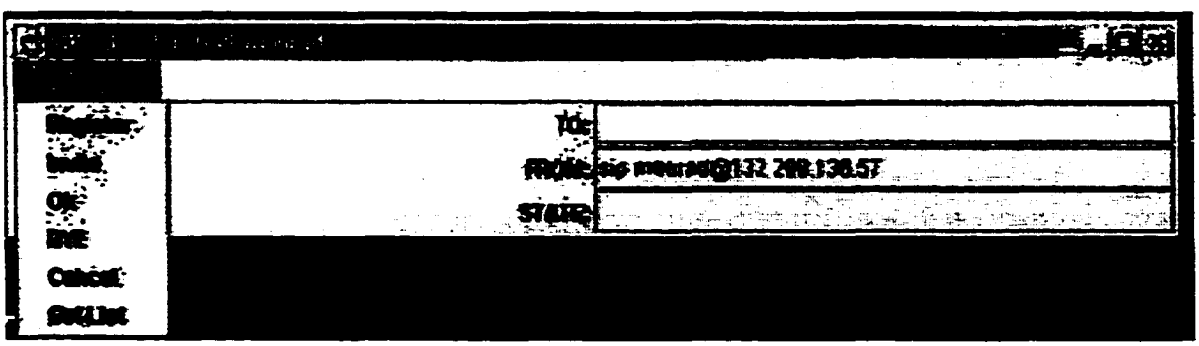


Figure 18 Boîte de dialogue du client

5.4.2 Enregistrement des clients

La première étape dans une session SIP est l'enregistrement de tout les clients auprès du serveur en lui envoyant un requête REGISTER. En cliquant sur le bouton Register du menu défilant de la boite de dialogue, celui-ci permet une telle opération et le client est enregistré. Le champs source (From) est nécessaire pour accomplir cet opération. Le serveur de son côté dresse une table de tous les clients présent dans la session avec leurs noms, adresses IP, url et ports. La figure 19 montre deux exemples d'enregistrements des clients identifiés par les adresses IP : 132.208.136.57 et 132.208.136.110 envoyés au serveur dont l'adresse et le port sont 132.208.136.110 et 5060 respectivement.

Figure 19 Message Register

En cliquant sur le bouton d'enregistrement une requête SIP Register est créé et envoyée. Dans cette requête les url source (From) et destination (To) sont identiques. Si le client existe déjà dans la liste du serveur, celui-ci retourne une réponse d'échec « REGISTRATION FAILED » lui indiquant qu'il s'est déjà enregistré.

5.4.3 Invitation

Un client qui désire initier un appel et inviter un autre client, envoie une requête INVITE en spécifiant dans le champ To l'url du destinataire. Le proxy extrait l'adresse IP et le nom du destinataire du message, les compare à sa table et achemine le message vers sa destination. La figure 20 montre un exemple de requête INVITE d'au client dont l'adresse IP est 132.208.136.110 via l'adresse IP du serveur 132.208.0136.110.

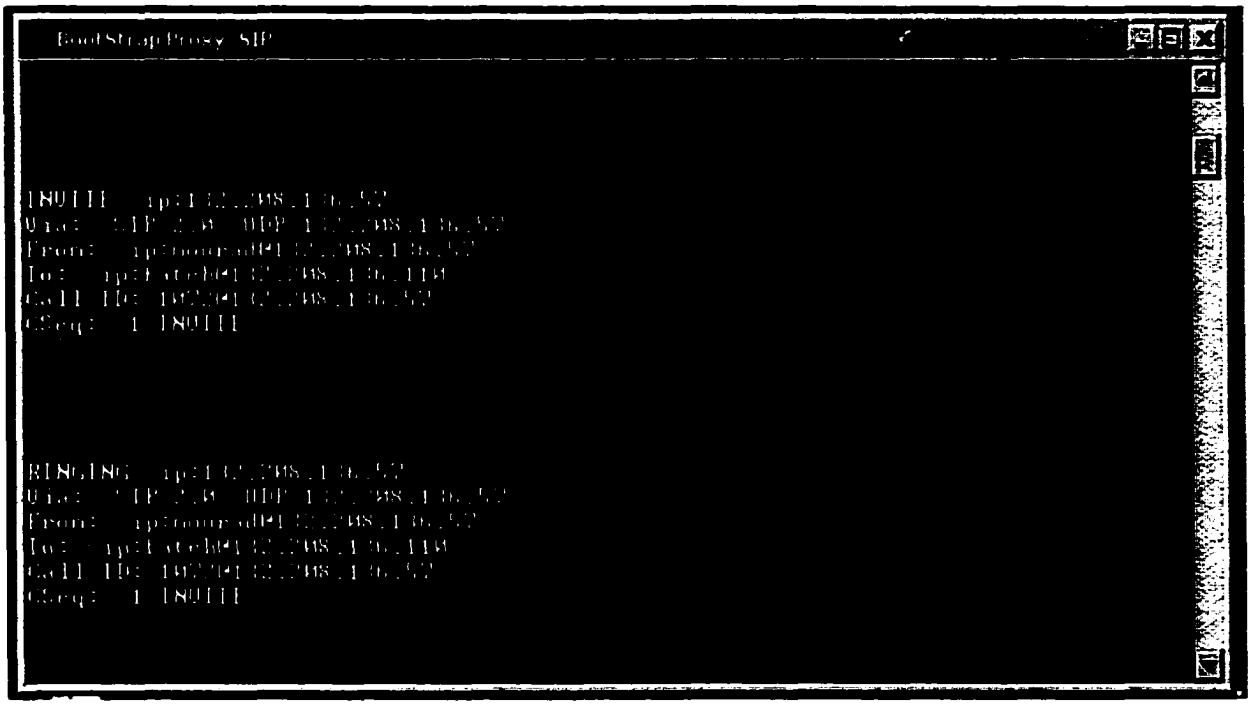


Figure 20 Message INVITE

Le proxy transmet la requête au destinataire et ce dernier voit s'afficher à son écran un message d'information « vous avez un appel de : » avec le nom du client source, comme illustré dans la figure 21.

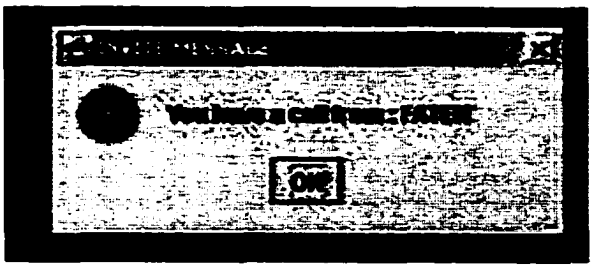


Figure 21 Message INVITE

Le serveur, d'après le modèle établit dans le chapitre 5, retourne en même temps un message «TRYING» vers le client source lui indiquant qu'il essaye de faire parvenir son message au client.

Dès réception de la requête INVITE par le destinataire, une réponse RINGING est retournée automatiquement au client source via le proxy ayant les mêmes url From et To que la requête INVITE. La figure 22 montre une réponse SIP : RINGING.

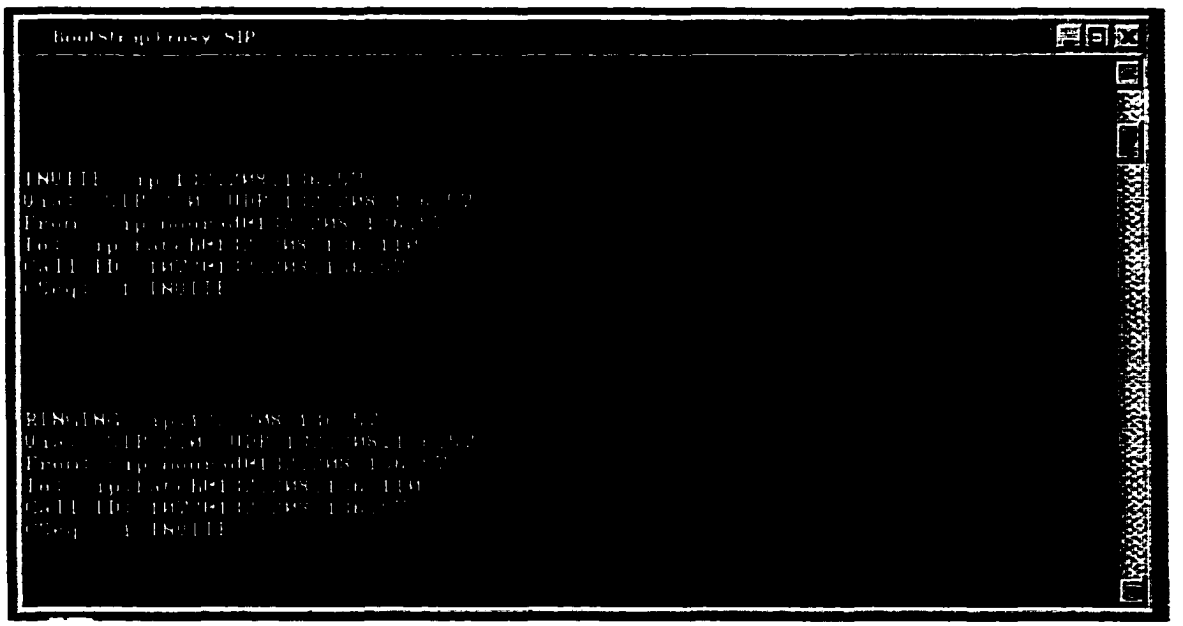


Figure 22 Message RINGING

Un client qui reçoit un message d'invitation, peut accepter l'invitation en cliquant sur le bouton Ok du menu défilant de la boite de dialogue, ce qui a pour effet de créer un réponse SIP OK et l'envoyer au serveur qui la retransmet à son tour au client source. De manière identique au message RINGING, la destination du message Ok est contenue dans l'url From. La figure 23 présente un message Ok :

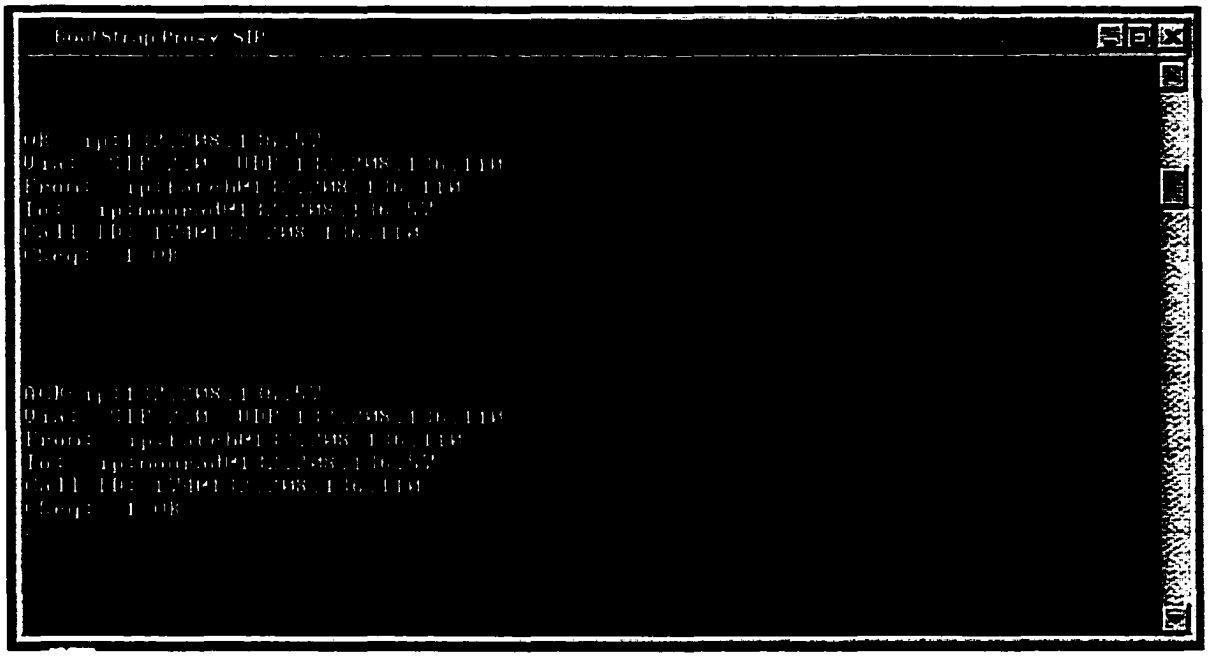


Figure 23 Message OK

La dernière étape de l'échange de requêtes SIP est l'acquittement par le client source. Lorsque celui-ci reçoit le message Ok, l'informant qu'il a accepté la communication, un message d'acquittement est envoyé automatiquement au destinataire via le proxy, comme il est montré dans la figure 24.

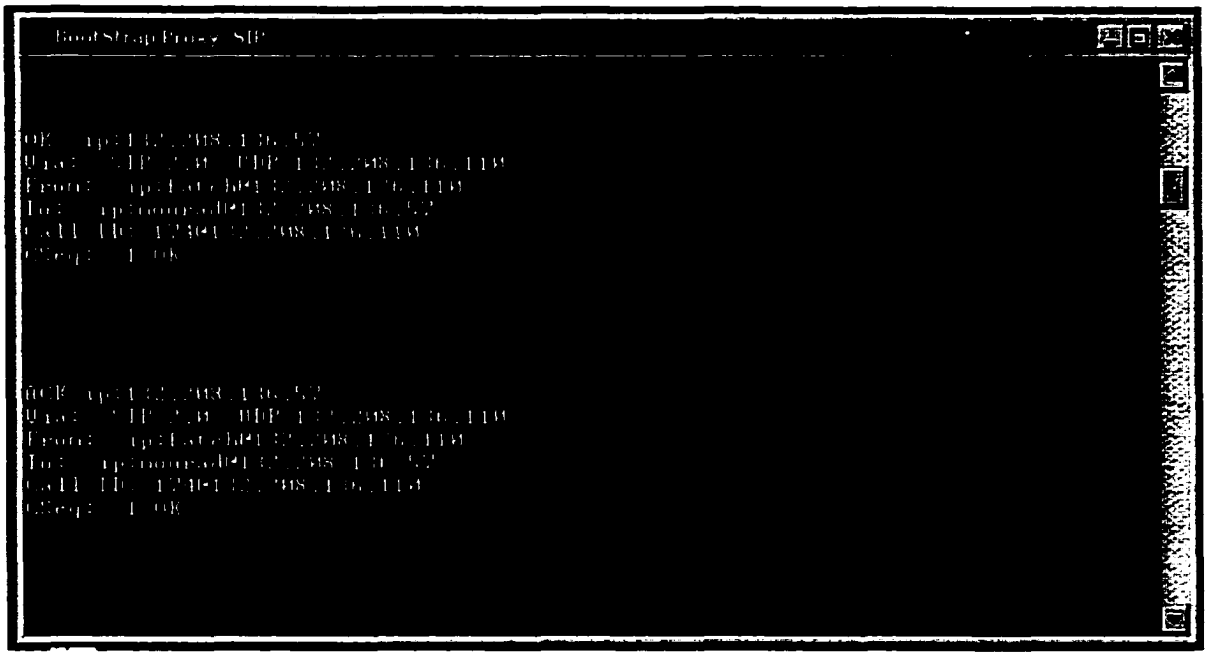


Figure 24 Message ACK

Ce dernier message est une indication que l'échange des paquets RTP et RTCP (audio et vidéo) peut commencer comme indiqué dans tous les champs d'états des boites de dialogues des clients concernés par le message « CONNEXION ÉTABLIE » durant toute la période de la conférence comme on peut le voir sur la figure 25.

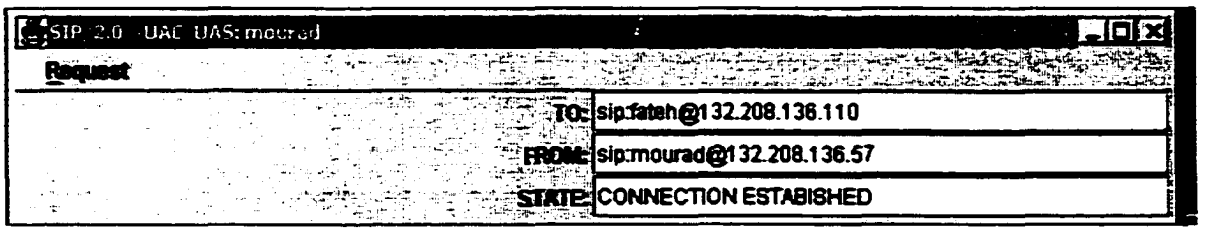
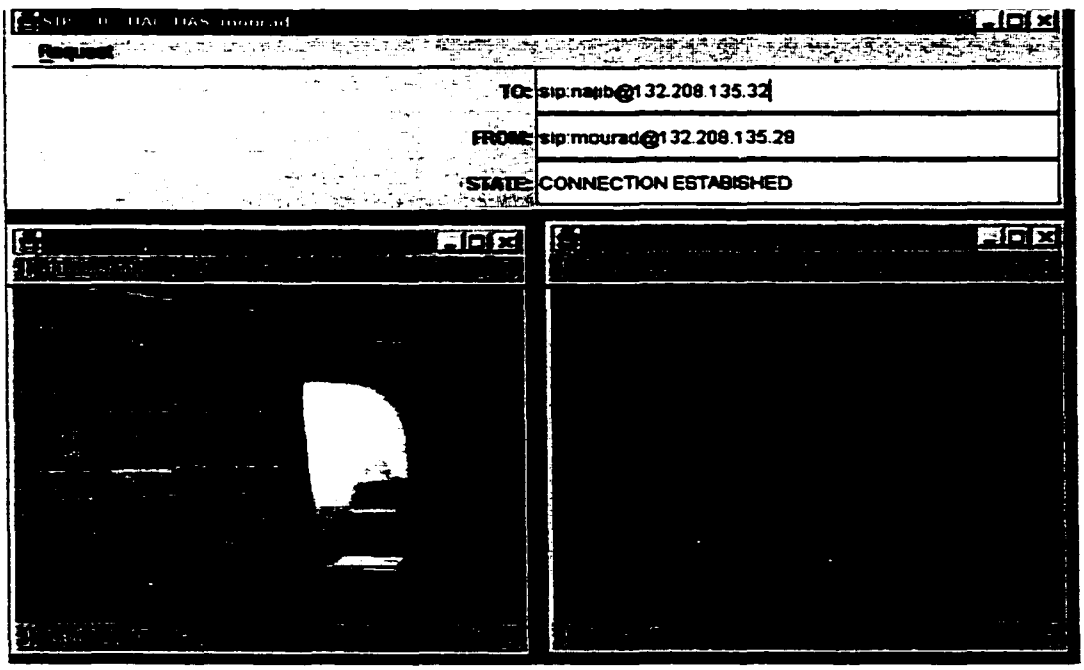


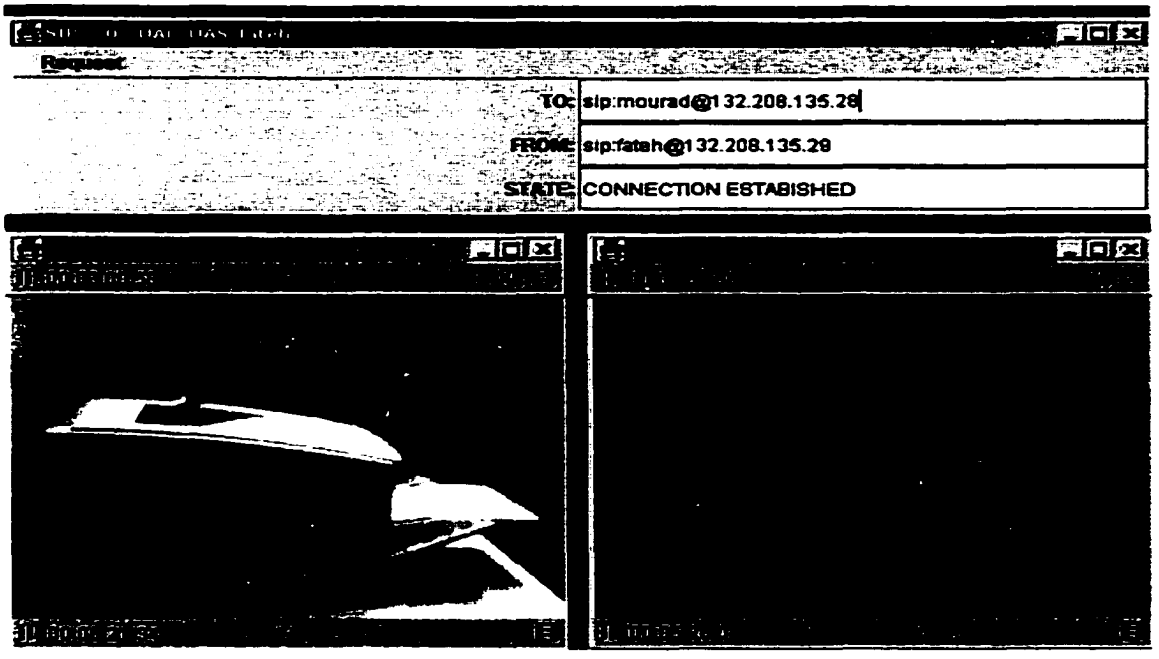
Figure 25 Message «connection établie»

5.4.4 Communication audio et vidéo

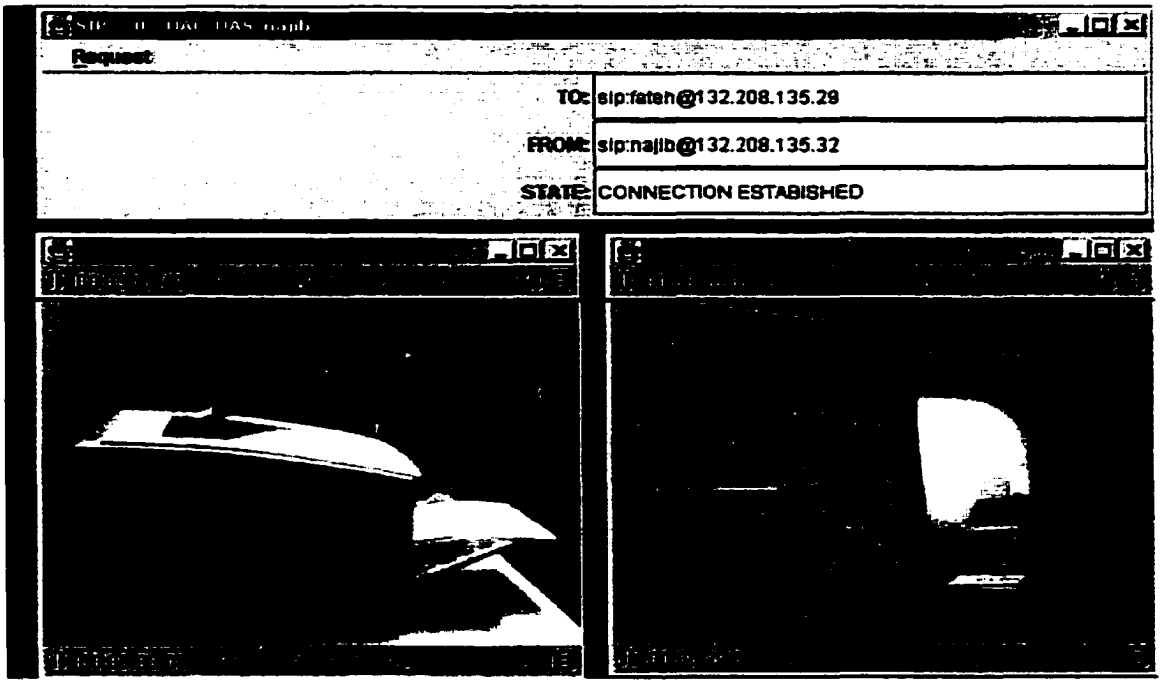
Lors de la réception des acquittements ACK par le serveur proxy, celui ci attribue les port de communications audio et vidéo par lesquelles les participants peuvent échanger les données multimédia, et les envoient à tous les clients en même temps. Du côté clients, chacun lance l'application relative au transport de la voix et vidéo par RTP et RTCP qui utilise l'API JMF. Au niveau de chaque client s'ouvre des écrans vidéo qui montrent les images reçues des autres clients participants à cette session ainsi qu'une interface audio permettant le contrôle du son. Et chaque client transmet aussi le son et la vidéo captés par le microphone et la caméra connectés à son poste.
Les figures 26-a, 26-b, 26-c montrent trois captures d'écrans d'une conférence audio et vidéo à trois participants. Sur chaque poste sont affichés deux écrans des deux autres participants et leurs interfaces audio. Il est à noter que dans le cadre de la protection de l'individu, nous avons jugé utile d'afficher des images sans participants.



(a)



(b)



(c)

Figure 26 Conférence à trois

5.4.5 Statistiques de transmission et de réception

Des méthodes de l'API JMF permettent de recueillir des statistiques sur la transmission et la réception audio et vidéo. Les deux figures 27 et 28 suivantes nous ces rapports RTCP audio et vidéo recueillis.

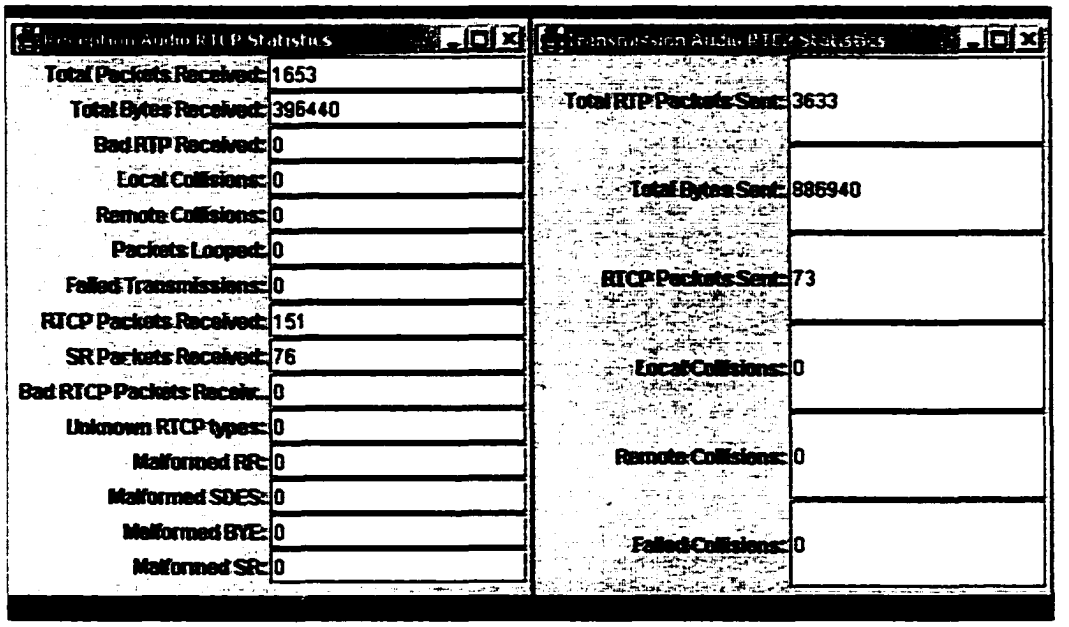


Figure 27 Statistiques RTCP vidéo

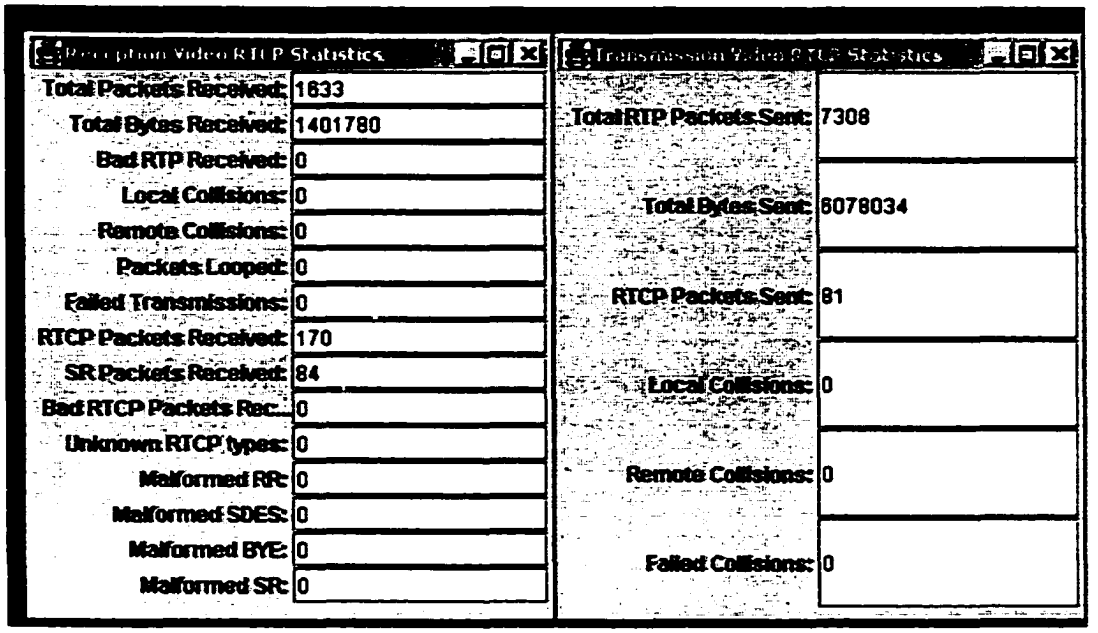


Figure 28 Statistiques RTCP audio

La qualité de service regroupe de nombreux paramètres qui donnent une estimation de la qualité de la voix et de la vidéo transmise et reçue, comme le nombre de paquets perdus, le nombre de collisions. Les mesures comprises dans le rapport RTCP n'indiquent aucune perte de données ni de collisions lors de la transmission sur le réseau. Ceci peut être dû au fait que le système a été testé sur un réseau local (LAN), et non sur le réseau Internet où le trafic est important et des problèmes de congestion, de pertes de données et de collisions peuvent surgir. Il est aussi à noter que l'application prend un temps assez long pour initialiser les sessions RTP après que la session soit établie et lancée. Ceci peut être dû aux importantes ressources utilisés par l'API JMF et à ses composantes graphiques. D'autant plus que le kit de java est assez lourd et consomme beaucoup de la capacité mémoire et influe la vitesse du CPU ce qui rend les temps d'exécution très lents.

5.4.6 Terminaison d'appel SIP

Un client qui désire mettre fin à sa connexion envoie une requête BYE vers le serveur proxy. Le proxy achemine cette requête vers tous les participants, la transmission et la réception des paquets audio et vidéo sont arrêtés et le proxy désenregistre le client, remet à jour sa liste de clients et libère les tous les ports de transmission et réception multimédia.

5.5 Difficultés rencontrées

Certaines difficultés rencontrées durant la période de réalisation de notre projet ont rendu notre travail assez difficile. En voici une énumération :
L'étude et l'apprentissage des différents protocoles mis en jeu pour aboutir à terme d'une application de voix sur IP et sa signalisation est assez difficile et la difficulté ici concerne surtout la mauvaise estimation de notre part du temps nécessaire pour cerner le travail.
Notre inexpérience de la programmation orienté objet (java) a été un problème majeur pour implémenter notre application.
Enfin, le manque total de documentation sur le l'API JMF a aussi été un énorme obstacle qui n'a pas facilité la progression du travail ni la bonne interprétation des résultats obtenus dans les rapports RTCP. La seule documentation qu'on a trouvé est le guide utilisateur de Sun téléchargé du site, mais qui reste insuffisante à notre avis, surtout pour quelqu'un qui n'a pas d'expérience dans ce domaine.

5.6 Conclusion

Dans ce chapitre, nous avons implanté une application de type client/serveur permettant de tester et de mettre en œuvre les différentes composantes présentées dans le chapitre précédent. Ces composants sont des classes et des méthodes qui peuvent être réutiliser dans le domaine de la téléphonie sur Internet. Ces méthodes permettent de réaliser la signalisation des appels selon le protocole SIP. D'autres

méthodes de l'API JMF permettent aussi de réaliser une communication multimédia et d'avoir des statistiques pour évaluer les performances de cette communication.

Dans la phase de test, nous avons présenté les différents message SIP échangés entre les clients pour établir un appel, ainsi que les images captées par les caméras et reçus au niveau de chaque client. Nous avons aussi présenté les statistiques RTCP transmis et reçues pour la voix et la vidéo.

CONCLUSION

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.

Pour certains, actuellement le seul frein à l'essor de la téléphonie sur IP serai la qualité. Or, comme celle-ci s'améliore de plus en plus grâce à l'augmentation conjointe de la bande passante d'Internet, de la vitesse de commutation, de la performance des CPU et enfin des algorithmes de compression, la téléphonie sur IP ne peut que se développer.

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 H323, 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 H323 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 défini les protocoles et concepts de mise en œuvre et de signalisation. Pour notre application nous nous sommes penchés sur deux volets, la conférence audio- vidéo entre usagers , et la signalisation des appels. Dans le premier volet, le protocole qui permet un transfert de données fiable en temps réel est le protocole RTP, couplé au protocole RTCP. Ce dernier permet le contrôle du flux de données et la gestion de la bande passante. La compagnie Sun Microsystems, œuvre depuis quelques années dans ce domaine et a mis à disposition des programmeurs Java, l'API JMF qui offre un très bon outil de programmation de ce type d'applications.

Il y a lieu de remarquer que malheureusement, mis à part le guide utilisateur de JMF que Sun a élaboré, la documentation sur cette API est extrêmement rare, et sa bonne compréhension nécessite des connaissances assez solide en programmation orienté objet (Java) et des concepts audio.

Dans le deuxième volet, nous avons opté pour le protocole SIP. Notre application est une application client-serveur. Le serveur est un proxy, il traite toute les requêtes provenant des clients et les achemine selon le contexte. Il inclut aussi le registrar , qui est un serveur qui s'occupe de l'enregistrement des clients et fournit à chaque fois au proxy le non, l'adresse IP et le numéro de port du destinataire. Notre cas d'application est un cas simple d'utilisation du protocole SIP vu que la présence des autres serveurs de localisation et de redirection ne sont pas nécessaires. On espère que ce modeste travail pourra servir à des travaux futurs pour élargir le domaine d'application.

Limitations et travaux futures

Des limitations sont observées dans notre travail dues surtout à notre inexpérience, mais qui peuvent être compléter dans des travaux futures. Parmi ces limitations, notre application SIP n'a pas été conçue pour être supporté par le protocole de transport fiable TCP. Son implémentation est basée sur le même principe client-serveur et utilise plutôt les sockets de flux, par contre il faudra utiliser les threads de java pour pouvoir travailler en multitâche. De plus, dans notre application basée sur UDP nous n'avons pas prévu de mécanismes de fiabilité basés sur les retransmission. En effet UDP n'est pas un protocole fiable, et il revient donc au protocole SIP d'assurer sa fiabilité en ajoutant d'autres services. De plus nous n'avons implémenté qu'une partie du protocole SIP qui répond à notre cahier de charge. Une autre limitation est que l'application n'a pas été testée sur le grand réseau, où d'autres problèmes peuvent surgir tels les délais, les collisions et pertes de paquets.

Les améliorations que nous proposons sont les suivantes :

- Concevoir un serveur de redirection SIP pour permettre la mobilité des stations et les autres services SIP.

- Implanter une passerelle pour permettre des communication téléphoniques en utilisant le réseau téléphonique public commuté (PSTN).
- Implanter un serveur de localisation dont le rôle est d'aider à router les messages SIP jusqu'à leurs destinations finales, dans le cas par exemple où des adresses privées ou des pare feu (firewalls) sont utilisés.
- Développer une interface entre les deux protocoles concurrents SIP et H323 pour rendre l'interopérabilité possible.
- Dans le but d'avoir une bonne performance et une meilleure qualité de service de la transmission et de la réception voix et vidéo sur des réseaux larges ou des réseaux métropolitains, il est important de penser à intégrer le protocole de réservation de ressource RSVP avec les protocoles SIP, RTP et RTCP.
- Un autre point concerne aussi la sécurité et l'authentification des messages. SIP permet d'échanger des messages confidentiels. Pour pallier à ce problème SIP possède des mécanismes de cryptages. Le cryptage de bout en bout du message SIP et de certains champs d'en-tête sensibles aux attaques, le cryptage au saut par saut pour empêcher les pirates de connaître les participants à la session et le cryptage de l'en-tête *Via* pour dissimuler le chemin emprunter par les messages. SIP possède aussi des mécanismes d'authentifications des messages pour empêcher tout intrus indésirable de modifier et de retransmettre les message SIP. Les mécanismes d'authentifications utilisés par le protocole http peuvent aussi être utilisé par SIP.

RÉFÉRENCES BIBLIOGRAPHIQUES

[1] Bill Douskalis (1999). *IP Telephony : The integration of robust VoIP Services.* Prentice Hall.

[2] Pierre Miller (1988). *Intégration voix et video, principes et concepts.* Paris : Masson .

[3] International Engineering Consortium. *The future of Voice/Data Consolidation : Markets, Technologies and Strategies*, [En ligne] http://www.iec.org/pubs/future_voice.html (Consulté le 15 Juin 2002)

[4] International Engineering Consortium, *Accelerating the deployment of Voice Over IP (VoIP) and Voice Over ATM (VoATM)* , [En ligne] http://www.iec.org/online/tutorials/voip_voatm/index.html (Consulté le 15 Juin 2002)

[5] D.Black Uyless (1999). *Voice Over IP*, Prentice Hall,.

[6] Jean-François Susbielle (1996). *Téléphonie sur Internet.* Paris : Eyrolles.

[7] Matthew Kolon, Walter J.Goralski (1999). *IP Telephony* - McGraw Hill.

[8] D.Minoli and E.Minoli (1998). *Delivering Voice Over IP Networks*, John Wilew,.

[9] Draft Recommendation H.323, version 4, (2000). Packet-based multimedia communication systems. *International Telecommunication Union (ITU-T)*

[10] Recommendation H.245, (1999) Control Protocol for Multimedia Communication. *International Telecommunication Union (ITU-T)*

[11] Draft Recommendation H.225.0, (2000). Call signalling protocols and media stream packetization for packet-based multimedia communication systems,. *International Telecommunication Union (ITU-T)*

[12] H Schulzrinne, S. Casner, R.Frederick, V.Jacobson, (1996). RTP: A Transport Protocol for Realtime Applications, Request for Comments: 1889, *Internet Engineering Task Force.*

[13] Sun Microsystems, *The Java Telephony API*, [En ligne]. http://www.java.sun.com (Consulté le 20 Février 2002)

[14] R. Stevens, G.R. Wright, TCP/IP Illustrated, Volume 2, Addison Wesley, 1995.

[15] Journal No 2/2001. *IP telephony*, [En ligne]. http://www.itu.int/itunews/issue/2001/02/index.html (Consulté le 05 Juillet 2002)

[16] William Stallings (1993). *Networking standards, a guide to OSI, ISDN, LAN and MAN Standards*, Addition Wesley.

[17] E.R.Harold.(1997). *Programmation réseau en Java*. Paris : O'reilly

[18] H.Schulzrine, *RTP: overview, May 1997*, [En ligne]. http://www.cs.columbia.edu/~hgs/rtp/ (Consulté le 02 Mars 2002)

[19] Sudheer Dhulipalla, Greg Meyer, Christopher Ross, Draft (2000) *SIP Interoperability Scenarios Test Plan, International Multimedia Teleconferencing Consortium*, [En ligne]. http://www.cs.columbia.edu/sip/drafts/sipsig_interop.doc (Consulté le 26 Juillet 2002)

[20] Sun Microsystems, *Java Media Framework API Guide, JMF 2.1.1*, [En ligne]. http://java.sun.com/products/java-media/jmf/2.1.1/guide/ (Consulté le 13 Janvier 2002)

[21] Shweizer Laurent, (2001). *Tutorial sur le protocole VoIP, SIP* , [En ligne].. http://www.tcom.ch/Etudiants/2001/lschweizer/sip.pdf (Consulté le 28 Juin 2002)

[22] Rapport de Jean-Claude Merlin, La téléphonie sur Internet, Avril 1999, *Conseil général des technologies de l'information* [En ligne]. http://www.telecom.gouv.fr/documents/merlin/rap_merlin0499_2.htm#d (Consulté le 03 Juillet 2002)

[23] Henning Schulzrinne, (1999) .*Session Initiation Protocol (SIP)*,. [En ligne]. http://www.cs.columbia.edu/~hgs/sip/ (Consulté le 07 Mars 2002)

[24] Nichlas Beijar (1998). *Signaling protocols for Internet Telephony, Architectures based on H.323 and SIP*, Helsinki University of Technology, Laboratory of Telecommunication Technology.

[25] Jonathan Davidson, Jim Peters(1999). *Voice Over IP Fundamentals, Macmillan*

[26] M.Handley, H.Schulzrinne, E.Schooler et J.Rosenberg, (1999). SIP: Session Initiation Protocol, *RFC 2543. IETF*, [En ligne]. www.cis.ohio-state.edu/cgi-bin/rfc/rfc2543.html (Consulté le 07 Mars 2002)

[27] H.Schulzrine, (2002). Universal Emergency Address for SIP-based Internet Telephony, *Internet Draft, IETF*, [En ligne]. http://www.ietf.org/internet-drafts/draft-schulzrinne-sipping-sos-02.txt (Consulté le 20 Mars 2002

[28] H.Schulzrinne, RTP News. July 2002. http://www.cs.columbia.edu/~hgs/rtp/

[29] Philipe Dax, *RTP/RTCP*, [En ligne]. http://www.infres.enst.fr/~dax/polys/multicast_api/rtp.html (Consulté le 22 Fevrier 2002)

[30] H.Schulzrinne, (2002). *SIP: Comparison of SIP and H.323*, [En ligne]. http://www.cs.columbia.edu/~hgs/sip/h323-comparison.html (Consulté le 02 Août 2002)

[31] H.Schulzrinne. D.Oran, G.Gamarillo, (2002). The reason Header Field for the Session Initiation Protocol, Internet Draft, *IETF*, [En ligne].. http://quimby.gnus.org/internet-drafts/draft-schulzrinne-sip-reason-01.txt (Consulté le 20 Mars 2002)

[32] H.Schulzrinne, J.Rosenberg, (1998). A Comapaison of SIP and H.323 for Internet Telephony, Network and Operating Support for Digital Audio and Video, Cambridge, England.

[33] Andrew S. Tanenbaum, *Computer Networks, 3rd Edition*, New York Prentice Hall

[34] G.Pujolle (1998). *Les réseaux, 2ème édition*, Eyrolles

[35] Farid Ait-ouyahia (2000) *Implémentation d'une passerelle en Java pour la voix entre le réseau Internet et un réseau CDMA*, Thèse de maîtrise, UQAM.

[36] P. Bernier, (1998). The Standards Struggle Will SIP be a Drain on H.323's momentum. *Sounding Board*, May. [En ligne]. http://www.soundingboardmag.com/articles/851feat2.html (Consulté le 02 Mai 2002)

[37] D.Crocker, Standard for the format of ARPA internet text messages, RFC 822, Internet Engineering Task Force, Aug 1982.

[38] D.Sisalem, H.Schulzrinne, The multimedia internet terminal. In accepted for pubication in the special issue on Multimedia of the Journal of Telecommunication Systems, June 1997.

[39] K.Bakhti (2000). La voix sur Internet dans le context du Standard H.323, Thèse de maîtrise, UQAM.

[40] H.Schulzrine. J.Rosenberg, (1999). The Session Initiation Protocol : Providing Advanced Telephony Servicesw Accross the Internet, *Bell Labs Technical Journal*

[41] H.Schulzrine, J.Rosenberg, Signaling for Internet Telephony, Technical Report, Columbia University, New York, February 1998.

[42] H.Schulzrinne, J.Rosenberg, SIP call control services, Internet Draft, Internet Engineering Task Force, work in progress, February 1998.

[43] Robert Caputo (1999). *Cisco Packetized Voice and Data Integration*, McGraw-Hill,

[44] M.Michael, (2000) SIP Rules, Computer Telephony,. [En ligne]. http://www.computertelephony.com/article/CTM20000515S0003 (Consulté le 17 Fevrier 2002)

[45] The OpenH323 Project [En ligne]. http://www.openH323.org (Consulté le 10 Août 2002)

[46] The SipCenter. [En ligne]. http://www.sipcenter.com (Consulté le 10 Août 2002)

[47] Ubiquity Software Corporation, [En ligne]. http://www.ubiquity.net/solutions/wpaper.htm (Consulté le 10 Août 2002)

[48] White Paper, Application Powered Networks with SIP, Release 2.0, Ubiquity Software Corporation.

[49] N.J.Muller (1999). *Desktop Encyclopedia of Voice and Data Networking*, McGraw-Hill, August.

[50] Benoit Boute (1997). Multicast over RTP, *ENST*. [En ligne] http://www.infres.enst.fr/~dax/guides/multicast/menst2.html (Consulté le 14 Mars 2002)

[51] C.Kirk (1998). *The Internet Phone Connection*, Osborne McGraw-Hill.

[52] www.jpeg.org

[53] www.mpeg.org

[54] http://perso.club-internet.fr/f_bailly/VoIP/rapport_FINAL.htm (Consulté le 13 Août 2002)

[55] http://www.commentcamarche.net/internet/rtcp.php3 (Consulté le 15 Août 2002)