الجـــــــــــــــــــمهـوريـــــة الجـــــــــــــزائـريـة الديمـــــــقـراطيـة الـشـــــــــعبيـة

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التـــــــــــــــــــــــــــعليـم العـــــــالي و البـــــــــحث العـــــلـمـــي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جـــــــــــــــامعة أبي بكـــر بـلـقـايد – تـلمســـــــــــــان –

Université Aboubakr Belkaïd – Tlemcen –
Faculté de TECHNOLOGIE

MEMOIRE

Présenté pour l'obtention du **diplôme** de **MASTER**

En : Télécommunications

Spécialité : Réseaux Mobiles et Services de Télécommunications

Par : BERRAHIL Zeyneb
BIBI TRIKI Mohammed Chakib

Sujet

Développement d'une application client de la voix sur IP (VoIP)

Soutenu publiquement, le 13 / 06 /2017, devant le jury composé de :

M. BOUABDALLAH Réda	MAA	Univ. Tlemcen	Président
M. BEMMOUSSAT Chemseddine	MAB	Centre Univ. Temouchent	Directeur de mémoire
M. FEHAM Mohammed	Prof	Univ. Tlemcen	Co-dircteur de mémoire
M. MERAD BOUDIA Omar Rafik	MAB	Univ. d'Oran	Examinateur

A cœur vaillant rien d'impossible

A conscience tranquille tout est accessible

Quand il y a la soif d'apprendre

Tout vient à point à qui sait attendre

Quand il y a le souci de réaliser un dessein

Tout devient facile pour arriver à nos fins

Malgré les obstacles qui s'opposent

En dépit des difficultés qui s'interposent

Les études sont avant tout

Notre unique et seul atout Ils représentent la lumière de notre existence

L'étoile brillante de notre réjouissance

Comme un vol de gerfauts hors du charnier natal

Nous partons ivres d'un rêve héroïque et brutal

Espérant des lendemains épiques

Un avenir glorieux et magique

Souhaitant que le fruit de nos efforts fournis Jour et nuit, nous mènera vers le bonheur fleuri

Aujourd'hui, ici rassemblés auprès des jurys,

Nous prions dieu que cette soutenance Fera signe de persévérance

Et que nous serions enchantés Par notre travail honoré

on dédie ce mémoire a....

A nos parents,

Pour l'éducation et le grand amour dont ils nous ont entouré depuis notre naissance. Et pour leurs patiences et leurs sacrifices.

A nos cher frère ;

A tous nos proches ;

A tous ceux qui nous aiment ;

A tout nos ami(e)s ;

A tous ceux que nous aime.

Remerciements

Ce mémoire est le résultat d'un travail de recherche très dur. En préambule, nous voulons adresser tous nos remerciements aux personnes avec lesquelles on a pu échanger et qui nous ont aidé pour la rédaction de ce mémoire.

En commençant par remercier tout d'abord Monsieur Bemmoussat Chemseddine, directeur de recherche de ce mémoire, pour son aide précieuse et pour le temps qu'il nous a consacré.

Merci à Monsieur Feham Mohammed , responsable de Master RMST qui a su nous guider vers les bonnes références.

Enfin, on s'adresse nos plus sincères remerciements à notre famille : nos parents, nos frères et tous nos proches et amis, qui nous ont accompagné, aidé, soutenu et encouragé tout au long de la réalisation de ce mémoire.

Liste des figures

Introduction Générale

L'acronyme VOIP signifie « Voice Over IP » (voix sur IP).

La plupart d'entre nous connaissons le système téléphonique commuté public (PSTN, Public Switched Telephone System), qui nous permet de contacter des personnes dans le monde entier en composant une série de chiffres. La VoIP constitue une alternative, qui fonctionne en acheminant les signaux vocaux numérisés sur des réseaux IP, tels que les réseaux Intranet d'entreprise ou, dans certains cas, l'Internet public.

À première vue, le système PSTN n'a pas vraiment changé en plus de 100 ans. Pourtant, de nombreuses modifications et améliorations technologiques y ont été apportées, telles que la tonalité de composition et l'identification de l'appelant. Néanmoins, pour l'utilisateur, il s'agit toujours de composer (plus récemment, par pression) une série de chiffres et d'être connecté à la personne à laquelle appartient le numéro composé. Toutefois, pour atteindre ce résultat, ce qui se passe en coulisses a considérablement changé au cours des dernières années.

La VoIP n'est pas réellement une nouvelle technologie ; des documents et brevets portant sur le sujet sont datés de plusieurs décennies et les premiers logiciels de VoIP étaient déjà disponibles en 1991. Le principe de base est assez simple : il s'agit essentiellement de la même technologie que celle utilisée pour diffuser de la musique sur Internet. Les sons de la voix sont récupérés par un microphone et numérisés par la carte son. Les sons numérisés sont alors compressés à l'aide d'un codec audio. Ce dernier supprime les données inutiles, tout en conservant la lisibilité des sons, afin de rendre le flux suffisamment compact pour qu'il soit envoyé en temps réel sur le réseau. Le terme codec est l'abréviation utilisée pour « CODeur/DÉCodeur ». Les sons sont encodés à la fin de l'envoi, envoyés sur le réseau, puis décodés à la réception, où ils sont rejoués dans les haut-parleurs ou le casque d'écoute.

Quelques années en arrière, le concept IP est devenu l'actualité des nouveaux réseaux, comme la 4G, la 5G, IOT…etc.

Notre mémoire est divisé en deux chapitres : le premier chapitre est une introduction au réseau IP, ou nous allons discuter sur le concept de base de la VoIP. Dans le deuxième chapitre nous allons se concentré sur notre développement de notre client IP sous Android.

Chapitre I

Chapitre I : Introductionà la Voix sur IP ‘’VoIP’’

I.1. Introduction :

Pendant bien des années, le transport de la voix et celui des données se faisait de manière distincte au sein des entreprises. Pour le transport de la voix il fallait recourir au réseau téléphonique (réseau fixe ou réseau GSM) et pour les données recourir au réseau informatique. Bien entendu cet état de chose n’était pas sans conséquences. En effet, les entreprises étaient confrontées à des problèmes tels : les budgets et dépenses en télécoms de plus en plus élevés, la perte de la bande passante allouée à une communication qui n’a pu être établie (gestion non rationnelle de la bande passante), affectation d’un intervalle de temps (IT) à un canal voie où données, échec de l’intégration des services de transport de données dans les PABX pour ne citer que ceux-là. Au regard de ces limites, la VoIP (Voice Over IP) qui fait l’objet de ce projet, a été pensée. Elle désigne un terme générique définissant le transport du trafic vocal au moyen de la transmission par paquets. Elle recouvre les solutions techniques et des services associés variés : communications entre PC, sur le réseau interne, services privés ou ouverts au public. En d’autres termes, il s’agit d’une technologie de transport de la voix sur un réseau IP, PABX + passerelle + réseau WAN IP. A ne pas confondre avec la ToIP (Telephony over IP) qui repose en fait sur la VoIP. Ainsi, grâce à la VoIP, les coûts des communications entre les sites distants (siège-agences) seront maintenant maîtrisés. Dans la suite de ce projet relatif à une étude de migration du PABX vers la VoIP entre le siège d’une entreprise et sa succursale, les aspects : solution VoIP retenue, matériels (hardware, software), configurations et connexions seront mis en évidence.[1]

I.2. Qu’est-ce que la VoIP ?

La VoIP est une technologie de communication dont l’une des principales applications est la téléphonie IP.Le principe fondamental de la VoIP est la transmission de données parlées via un réseau IP (Internet protocol).
Concrètement, une conversation téléphonique VoIP ne passe plus par des fils de cuivre du réseau téléphonique traditionnel analogique. Elle circule sur un réseau de données numériques en passant les câbles Ethernet à l’arrivée et au départ. Pour dire les choses simplement, une conversation téléphonique VoIP passe par Internet.[2]

I.3. Principes de la Voix sur IP :

Le but de la téléphonie sur IP est de transmettre le flux de données de la téléphonie sur un réseau IP à moindre coût et avec une qualité de transmission et une fiabilité comparables à celles du réseau téléphonique classique RTC. C'est le cas idéal d'une transmission sur un réseau IP unique ou une série de réseaux IP reliés entre eux. Cependant, dans la majorité des cas une conversation téléphonique devra passer également dans un ou plusieurs réseaux d'accès de technologies différentes. D'où la difficulté d'interconnecter des réseaux de technologies différentes tout en assurant une fiabilité et une qualité de transmission aussi bonne que dans le réseau RTC traditionnel.
La téléphonie sur IP est aujourd'hui basée sur les passerelles (gateway) qui relient le monde de la téléphonie traditionnelle et le monde IP. Une passerelle est connectée d'un côté au réseau téléphonique classique et, de l'autre, au réseau Internet. La passerelle se charge de la

conversion du signal téléphonique en paquets IP et vice-versa. Ces deux opérations sont effectuées en même temps pour permettre une connexion duplex.
Le réseau d'accès peut être du type RTC, RNIS, ADSL, WLL et GSM.

Trois architectures sont possibles pour établir une liaison téléphonique basée sur IP.

- **Téléphonie entre ordinateurs (PC to PC) :** Deux ordinateurs peuvent directement s'appeler par l'intermédiaire du réseau IP sur lequel ils sont connectés. De tels services sont aujourd'hui disponibles sur le marché soit directement dans le système d'exploitation du PC, soit comme logiciel particulier.

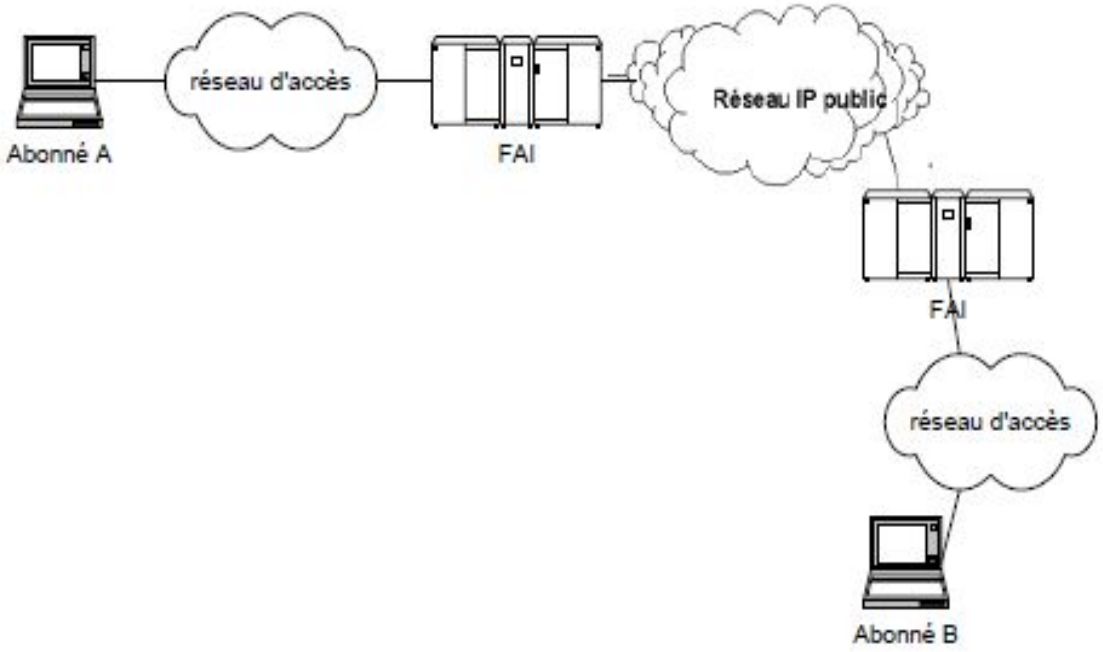


Figure 1.1 : Architecture PC to PC

Ce type d'application devrait cependant rester limité du fait de la nécessité d'avoir en permanence des ordinateurs en fonctionnement et peu mobiles.

- **Téléphonie entre ordinateurs et poste téléphonique (PC to Phone)**

  L’ordinateur, connecté à un réseau IP, n'a besoin d'aucun élément supplémentaire. Par contre, le téléphone doit être raccordé à une passerelle qui fera le lien entre le réseau téléphonique commuté (RTC) et le réseau IP concerné.

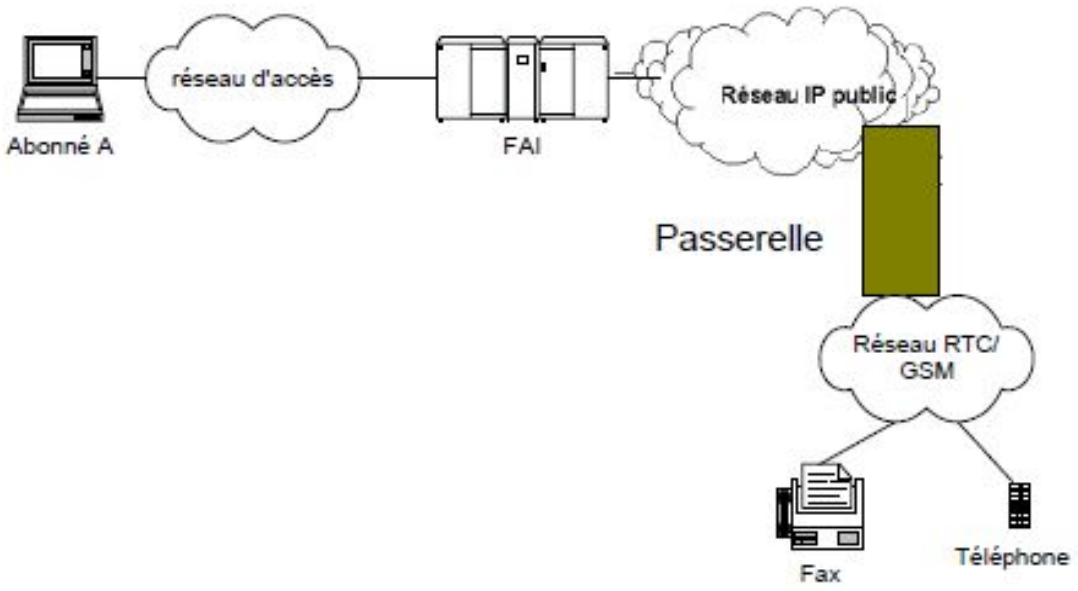


Figure 1.2 : Architecture PC to Phone

système peut Ce présenter les mêmes contraintes que dans le cas précédent.

- **Téléphonie entre postes téléphoniques (Phone to Phone) :** Chacun des deux téléphones doit être raccordé à une passerelle pour leur permettre de communiquer sur un réseau IP. [7]

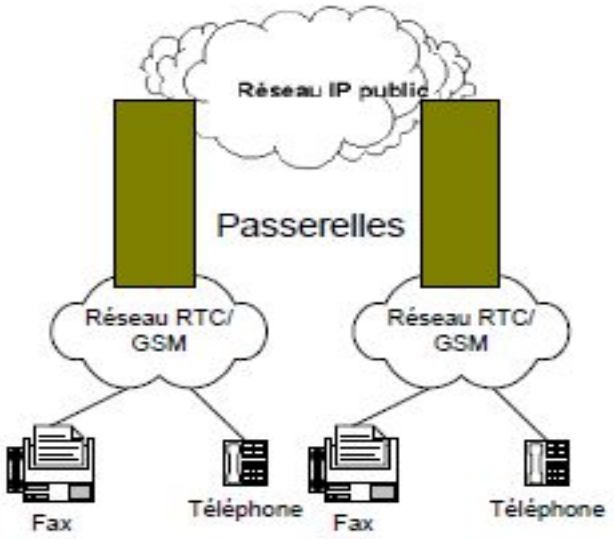


Figure 1.3 : Architecture Phone to Phone

I.4. IP version 4

Comme la VoIP est basé sur le modèle IP, dans ce paragraphe nous allons détailler les versions IP, commençons par la première version.

La version 4 du protocole a été standardisée en 1981. Elle est exploitée par la majorité des applications sur le réseau.

L’en-tête IP version 4 est représenté par la figure suivante :

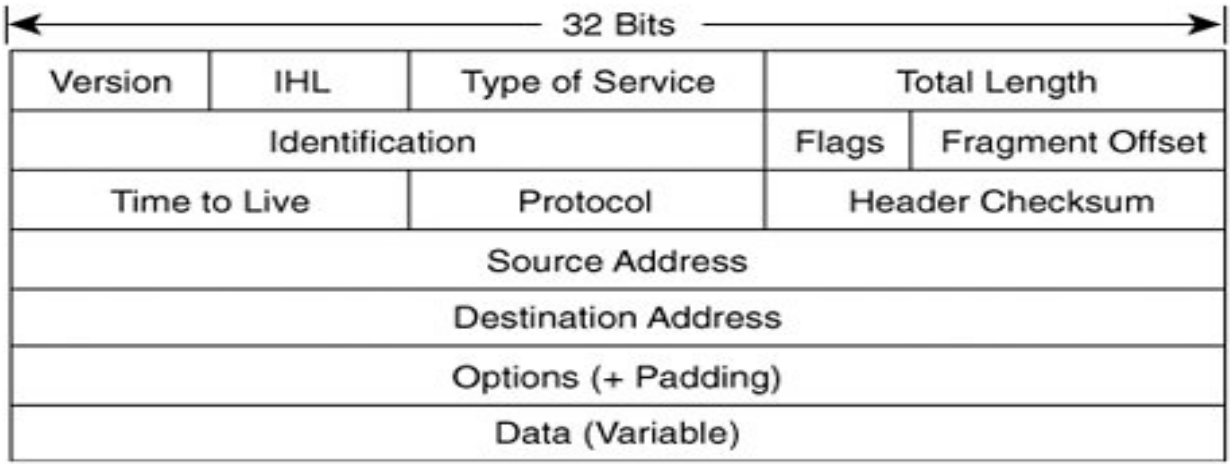


Figure 1.4 : l’entête IP version 4

- *Version (version) : sur* 4 bits. La version *actuelle est la 6 mais dans ce cas la version est noté 4 (pour IP v 4).*
- *Longueur en- tête (IHL : Internet Header Lent) :* sur 4 bit. La longueur d’un en tête IP est d’au moins 20 octets.
- *Type de service TOS (Type Of Service) :* ce champ de 8 bits a été introduit pour permettre à un émetteur de définir des classes de services, et donc des niveaux de priorité pour les paquets de trafic qu’il émet. Ces paramètres seraient très utiles au transport de contenus temps réel. Malheureusement, les algorithmes de routage n’ont jamais tenu compte du champ TOS, et sont incapables de l’exploiter. Les applications temps réel étaient inexistantes jusqu’à une période récente, et les services web n’ont pas justifié son activation. Pourtant, depuis 1997, la communauté de l’Internet commence à beaucoup s’intéresser à l’octet TOS. Il offre un champ libre pour inscrire le niveau de priorité demandé pour un paquet.
- Des routeurs intelligents sont désormais capables d’intégrer la classe de serviceainsi définie dans leur stratégie d’allocation de bande passante et de gestion desfiles d’attente. Cette nouvelle architecture, DiffServ, offre un moyen simple demaîtriser la qualité de services. Elle est détaillée aux troisièmes chapitres.

A l’origine, la structure de l’octet TOS avait été définie comme suit :

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*
Priorité			*D*	*T*	*R*	*Inutilisé*	

L’indice de priorité sur 3 bits peut servir à marquer les paquets et à leshiérarchiser, en fonction de leur classe de service. Les routeurs peuvent alorsdédier davantage de ressources à ceux qui subissent des contraintes strictes detemps. Les bits D, T et R, permettent de personnaliser le mode de routage, enl’adaptant à la nature du service.
Le bit D (*Delay*), positionné sur 1, indique un impératif de délai court, le bit T (*Throughput*) un débit élevé et le bit R (*Reliability*) une fiabilité élevé.
Le délai d’acheminement étant le facteur le plus critique dans le transport de latéléphonie, le bit D serait systématiquement positionné à 1. L’octet TOS n’ajamais été considéré par les routeurs. Les « services différenciés »commencent à l’exploiter.

- *Longueur total (total length) :* sur 16 bits. La taille maximale du paquet est définie par le MTU.
- *Identification :* sur 16 bits. Le numéro d’identification permet de reconstruire le paquet lorsque celui-ci a été fragmenté dans sa traversée du réseau. Tous les fragments associés à un même paquet peuvent ainsi être reconstitués.
- *Drapeau (Flag) :* sur 4 bits. Le drapeau est employé dans le processus de fragmentation des paquets ; il permet par exemple d’identifier le dernier fragment appartenant à un paquet, pour signaler que la reconstitution de celui-ci peut commencer.
- *Déplacement de fragment (fragment offset) :* sur 12 bits. Indique la place du fragment dans la composition du paquet. Le premier fragment est marqué « 0 », puis l’indice est incrémenté de 1 pour chaque fragment suivant.
- *TTL (Time To Live) :* temps de vie sur 8 bits, pour éviter qu’un datagramme soit pris dans une boucle (*loop*) et ne tourne indéfiniment dans le réseau, le champ TTL est décrémenté lors de chaque passage dans un routeur. Lorsque ce temps arrive à zéro, le routeur détruit le paquet et en informe la source en émettant un message ICMP.
- *Protocol :* ce champ précise le protocole de niveau supérieur utilisé. Les « enveloppes » des différentes couches de l’architecture internet sont « emboîtées » l’une dans l’autre. Le niveau 2 de l’architecture, fourni par exemple par un réseau Ethernet, établira un en tête Ethernet sur lequel sera mentionné le protocole supérieur, IP. Le protocole IP, de couche 3, construira ses paquets en établissant un en tête sur le quel figurera le nom du transport de la couche 4. Ce peut être TCP, mais les services temps réel préfèrent le protocole simple UDP. A son tour, l’enveloppe UDP contiendra un paquet RTP, décrit dans le chapitre trois. Les protocoles sont repérés par un simple numéro enregistré auprès de l’IANA, parmi les principaux, ICMP : 1, TCP : 6, UDP : 17.

- *Checksum ou « total de contrôle d'en tête » :* sur 16 bits : permet de détecter d'éventuelles erreurs de transmission des informations contenues dans l'en tête. Les erreurs de transmission sont devenues rares.
- Les adresses de l'émetteur et du destinataire sont codées sur 32 bits dans la version 4 de l'IP.
- Les options sont spécifiées par la source, et leur longueur peut atteindre 40 octets. Elles sont complétées par des bites de bourrage (*padding*). [1]

I.5. IP version 6

Connue pendant sa phase de définition et de développement sous le nom d'IPng, pour IP *nextgeneration*, le nouveau protocole internet a pris officiellement l'appellation IP version 6. IPv6 est une évolution du protocole IPv4. Il en constitue une mise à jour et assure une interopérabilité ascendante avec lui. La transition s'opère en douceur et de manière transparente. L'adoption d'IPv6 doit prendre plusieurs années.

I.5.1. Préparer l'avenir avec IPv6

La simplicité et l'universalité de l'IP furent la clé de son sucées. Le protocole apportait au parc d'ordinateur un langage commun facilement implantable, et permettant un dialogue immédiat. IPv6 conserve la même philosophie, tout en offrant des compléments indispensables au développement futur du réseau. Il permet, par exemple, de traiter de façon adaptée les flux temps réel, grâces à des indices de flux et de priorité ; il fournit par ailleurs des solutions pour assurer la sécurité de bout en bout, pour s'adapter à la mobilité des terminaux et des hôtes, et pour faciliter la configuration automatique.

I.5.2. Les objectifs principaux d'IPv6

Les spécifications du protocole IP, dans sa version 4, ont été figées et standardisées en 1981. Depuis, que le chemin parcouru pour le langage d'interconnexion des réseaux !. La nouvelle version du protocole ne répond pas seulement au besoin urgent d'augmenter le champ d'adressage, pour pouvoir intégrer les milliards d'objets de notre environnement, elle a également servi à le moderniser. Parmi les missions d'IPv6 :

- Augmenter la capacité d'adressage ;
- Simplifier la structure de l'en tête et réduire son temps de traitement par les routeurs ;
- Ajouter une étiquette de flot (*flow label),* pour faciliterle repérage et le traitement des paquets appartenant à un même flot temps réel par exemple ;
- Améliorer les mécanismes d'extension d'en-tête, et la définition d'options. [1]

I.5.3l’en-tête IPv6

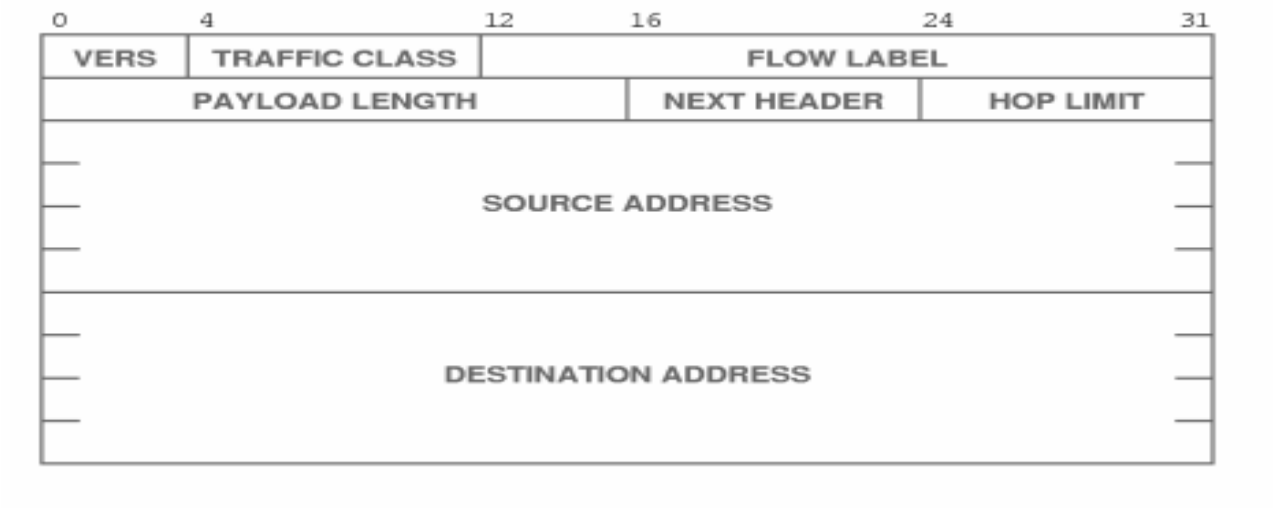


Figure 1.5 : l’entête IP version 6

- *Version :* sur 4 bits. Cette version du protocole est 6
- *Classe de trafic (trafic class) :* sur 8 bits. Indice servant à distinguer un paquet, pour, par exemple, lui affecter un traitement prioritaire. Ce champ est semblable à celui de l’octet TOS d’IPv4.
- *Etiquette de flot (flow label) :* sur 20 bits. Indique que ce paquet fait partie d’un flot représenté par un même couple d’adresse émetteur/récepteur et le numéro de port.
- *Longueur du contenu (payloadlength) :* sur 16 bits. C’est la taille exprimée en octets du contenu des données de ce paquet IPv6, à l’exclusion de l’en-tête.
- *En-tête suivant (next header) :* sur 8 bits. Identifie le type d’en-tête suivant immédiatement l’en-tête IPv6, TCP ou UDP par exemple. ce champ est identifié à celui de l’en-tête IPv6 (la version actuelle du protocole IP).
- *Limite nœuds (Hop Limit) :* sur 8 bits. Comme le champ TTL de l’en-tête d’IPv4, cette valeur indique le nombre maximal de nœuds par lesquels peut transiter le paquet. La valeur inscrite dans ce cham est décrémentée à chaque étape. Le paquet détruit lorsque cette valeur s’annule. Ce champ définit la durée de vie du paquet, le TTL ou Time To Live.
- *Adresse source (Source Adress) :* sur 128 bits. C’est l’adresse d’origine, correspondant à l’émetteur de ce paquet. Cette adresse ne comportait que 32 bits dans la version 4 d’IP.
- *Adresse de destination (Destination Adress) :* sur 128 bits. L’adresse du destinataire de ce paquet.
- *IPv6* a introduit dans son en-tête deux marqueurs servant à caractériser les paquets : « étiquette de flot » et « classe de trafic ». ces deux champs sont directement destinés au traitement des applications multimédias temps réel. [1]

I.6. TCP/IP :

TCP/IP désigne communément une architecture réseau, mais cet acronyme désigne en fait 2 protocoles étroitement liés : un protocole de transport, TCP (Transmission Control Protocol) qu'on utilise "par-dessus" un protocole réseau, IP (Internet Protocol). Ce qu'on entend par "modèle TCP/IP", c'est en fait une architecture réseau en 4 couches dans laquelle les protocoles TCP et IP jouent un rôle prédominant, car ils en constituent l'implémentation la plus courante. Par abus de langage, TCP/IP peut donc désigner deux choses : le modèle TCP/IP et la suite de deux protocoles TCP et IP.

I.6.1. TCP (Transmission Control Protocol) :

TCP est un service de livraison fiable, orienté connexion. TCP transmet les données sous la forme de segments et nécessite la création d'une session avant que les hôtes puissent échanger des données. TCP utilise des communications à base de flots d'octets (les données sont traitées sous la forme de suites d'octets).

TCP est fiable parce qu'il affecte un numéro d'ordre à chaque segment transmis. Si un segment est découpé en plusieurs morceaux, le récepteur sait s'il a bien reçu tous les morceaux. Les accusés de réception permettent à l'émetteur de savoir que le récepteur a reçu les données. Pour chaque segment émis, le récepteur doit retourner un accusé de réception (ACK), et ce dans un délai maximal spécifié. Si l'émetteur ne reçoit pas cet ACK, il retransmet les données. Si le segment reçu a été altéré, le récepteur le rejette. En pareil cas, il n'y a pas d'ACK et l'émetteur retransmet donc le segment.

I.6.2. IP (Internet Protocol) :

TCP découpe les données en paquets dont il garantit la livraison, mais c'est en fait IP qui effectue la livraison proprement dite. Au niveau de la couche IP, chaque paquet entrant ou sortant correspond à un datagramme. La couche IP ajoute les champs suivants à l'en-tête du paquet.

Les deux versions IP sont définies un peu plus haut dans notre mémoire.

Le modèle TCP/IP peut en effet être décrit comme une architecture réseau à 4 couches :

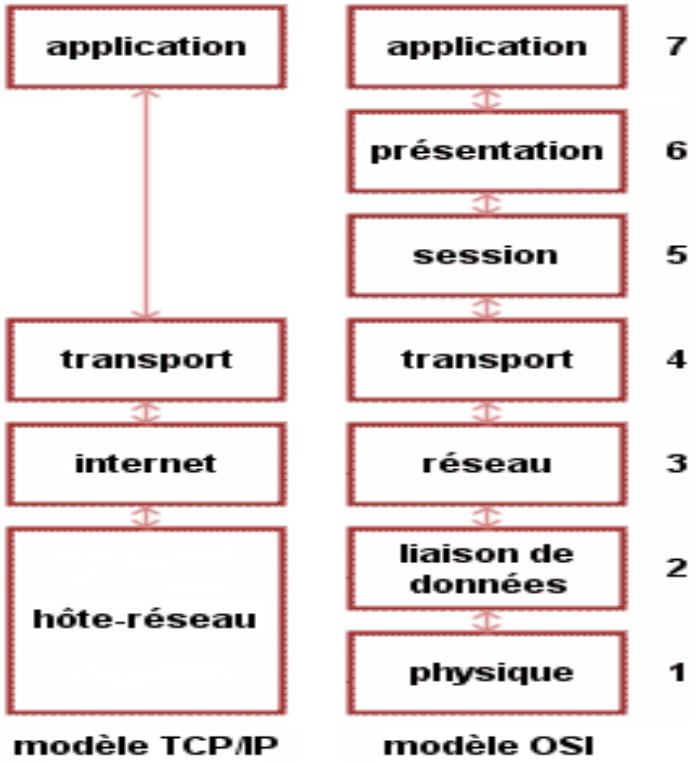


Figure 1.6 : Le modèle TCP vs OSI

I.6.3 - La couche hôte réseau :

Cette couche est assez "étrange". En effet, elle semble "regrouper" les couches physiques et liaison de données du modèle OSI. En fait, cette couche n'a pas vraiment été spécifiée ; la seule contrainte de cette couche, c'est de permettre un hôte d'envoyer des paquets IP sur le réseau. L'implémentation de cette couche est laissée libre. De manière plus concrète, cette implémentation est typique de la technologie utilisée sur le réseau local. Par exemple, beaucoup de réseaux locaux utilisent Ethernet ; Ethernet est une implémentation de la couche hôte-réseau.

I.6.4 - La couche internet

Cette couche est la clé de voûte de l'architecture. Cette couche réalise l'interconnexion des réseaux (hétérogènes) distants sans connexion. Son rôle est de permettre l'injection de paquets dans n'importe quel réseau et l'acheminement de ces paquets indépendamment les uns des autres jusqu'à destination. Comme aucune connexion n'est établie au préalable, les paquets peuvent arriver dans le désordre ; le contrôle de l'ordre de remise est éventuellement la tâche des couches supérieures.

Du fait du rôle imminent de cette couche dans l'acheminement des paquets, le point critique de cette couche est le routage. C'est en ce sens que l'on peut se permettre de comparer cette couche avec la couche réseau du modèle OSI.

La couche internet possède une implémentation officielle : le protocole IP (Internet Protocol). Remarquons que le nom de la couche ("internet") est écrit avec un i minuscule, pour la simple et bonne raison que le mot internet est pris ici au sens large (littéralement, "interconnexion de réseaux"), même si l'Internet (avec un grand I) utilise cette couche.

I.6.5- La couche transport :

Son rôle est le même que celui de la couche transport du modèle OSI : permettre à des entités paires de soutenir une conversation.

Officiellement, cette couche n'a que deux implémentations : le protocole TCP (Transmission Control Protocol) et le protocole UDP (User Datagram Protocol). TCP est un protocole fiable, orienté connexion, qui permet l'acheminement sans erreur de paquets issus d'une machine d'un internet à une autre machine du même internet. Son rôle est de fragmenter le message à transmettre de manière à pouvoir le faire passer sur la couche internet. A l'inverse, sur la machine destination, TCP replace dans l'ordre les fragments transmis sur la couche internet pour reconstruire le message initial. TCP s'occupe également du contrôle de flux de la connexion.

UDP est en revanche un protocole plus simple que TCP : il est non fiable et sans connexion. Son utilisation présuppose que l'on n'a pas besoin ni du contrôle de flux, ni de la conservation de l'ordre de remise des paquets. Par exemple, on l'utilise lorsque la couche application se charge de la remise en ordre des messages. On se souvient que dans le modèle OSI, plusieurs couches ont à charge la vérification de l'ordre de remise des messages. C'est l’avantage du modèle TCP/IP sur le modèle OSI, mais nous y reviendrons plus tard. Une autre utilisation d'UDP : la transmission de la voix. En effet, l'inversion de deux phonèmes ne gêne en rien la compréhension du message final. De manière plus générale, UDP intervient lorsque le temps de remise des paquets est prédominant.

I.6.6 - La couche application :

Contrairement au modèle OSI, c'est la couche immédiatement supérieure à la couche transport, tout simplement parce que les couches présentation et session sont apparues inutiles. On s'est en effet aperçu avec l'usage que les logiciels réseau n'utilisent que très rarement ces 2 couches, et finalement, le modèle OSI dépouillé de ces 2 couches ressemble fortement au modèle TCP/IP.

Cette couche contient tous les protocoles de haut niveau, comme par exemple Telnet, TFTP (trivial File Transfer Protocol), SMTP (Simple Mail Transfer Protocol), HTTP (HyperText Transfer Protocol). Le point important pour cette couche est le choix du protocole de transport à utiliser. Par exemple, TFTP (surtout utilisé sur réseaux locaux) utilisera UDP, car on part du principe que les liaisons physiques sont suffisamment fiables et les temps de transmission suffisamment courts pour qu'il n'y ait pas d'inversion de paquets à l'arrivée. Ce choix rend TFTP plus rapide que le protocole FTP qui utilise TCP. A l'inverse, SMTP utilise TCP, car

pour la remise du courrier électronique, on veut que tous les messages parviennent intégralement et sans erreurs.[3]

I.7. Conclusion :

Dans ce chapitre nous avons vu les bases de la VOIP et son principe de fonctionnement, pour cela nous avons fait une introduction sur le modèle en couche qui est dans notre cas le TCP/IP. Dans le prochain chapitre nous allons entamer notre application qui consiste a programmé une application Android coté client qui va être testé avec un serveur SIP pour avoir à la fin une application VoIP de bout en bout

Chapitre II

Chapitre II : Notre client Android

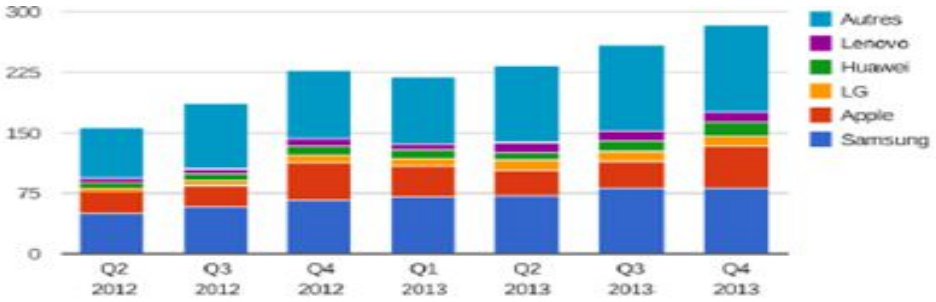
II.1. Motivation d'Android :

Dans ce paragraphe nous allons démontrer l'utilité et la motivation qui nous a laissé choisir le logiciel de notre mémoire qui est Android dans notre cas.

- Pourquoi développer des applications mobiles ?

Explosion des Smartphones :

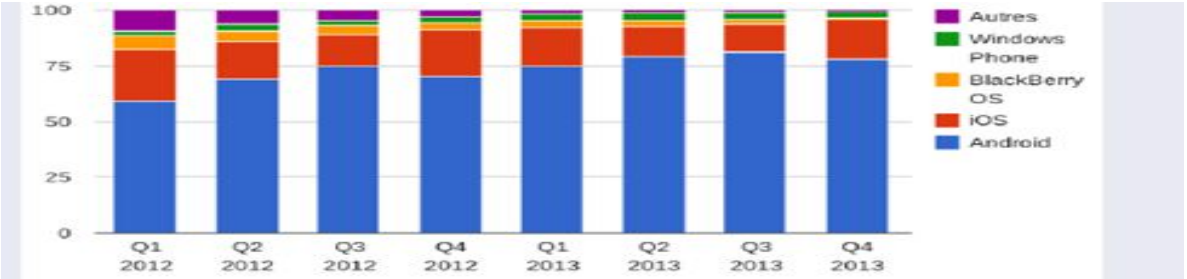
- Nombres d'abonnements mobiles :
  - ✓ En 2000 : < 1milliard.
  - ✓ En 2014 : Presque 7 milliards.
  - ✓ En 2017 : Plus que 10 milliards
- Ventes mondiales en 2012/2013 de Smartphones en millions d'unités :



Des Prévisions ont montré que d'ici 2018, les ventes seront multipliées par trois.

Répartition du marché :

- Parts de marché mondiales des OS de Smartphones en 2012 et 2013 en (%) :



Nous remarquons dans les statistiques ci-dessus que les chiffres des smartphone sont en croissance sans limites, et que Android prends une grande part de ces ventes.

- Principe :

Qu'est-ce qu'Android :

- ✓ Plates forme pour les devices mobiles.
- ✓ Gratuit.

- ✓ Open source.
- ✓ Flexible.

Android inclue :

- ✓ Un système d'exploitation basé sur Linux.
- ✓ Des applications basiques (téléphones, contacts,…).
- ✓ Un ensemble d'API avancées.
- ✓ SDK basé sous un sous-ensemble de JAVA (existe aussi dans d'autres langages.).
- Les difficultés :

Distribution des applications :

- ✓ Dépôts centralisé ou décentralisé.
- ✓ Depuis un PC.

Hardware :

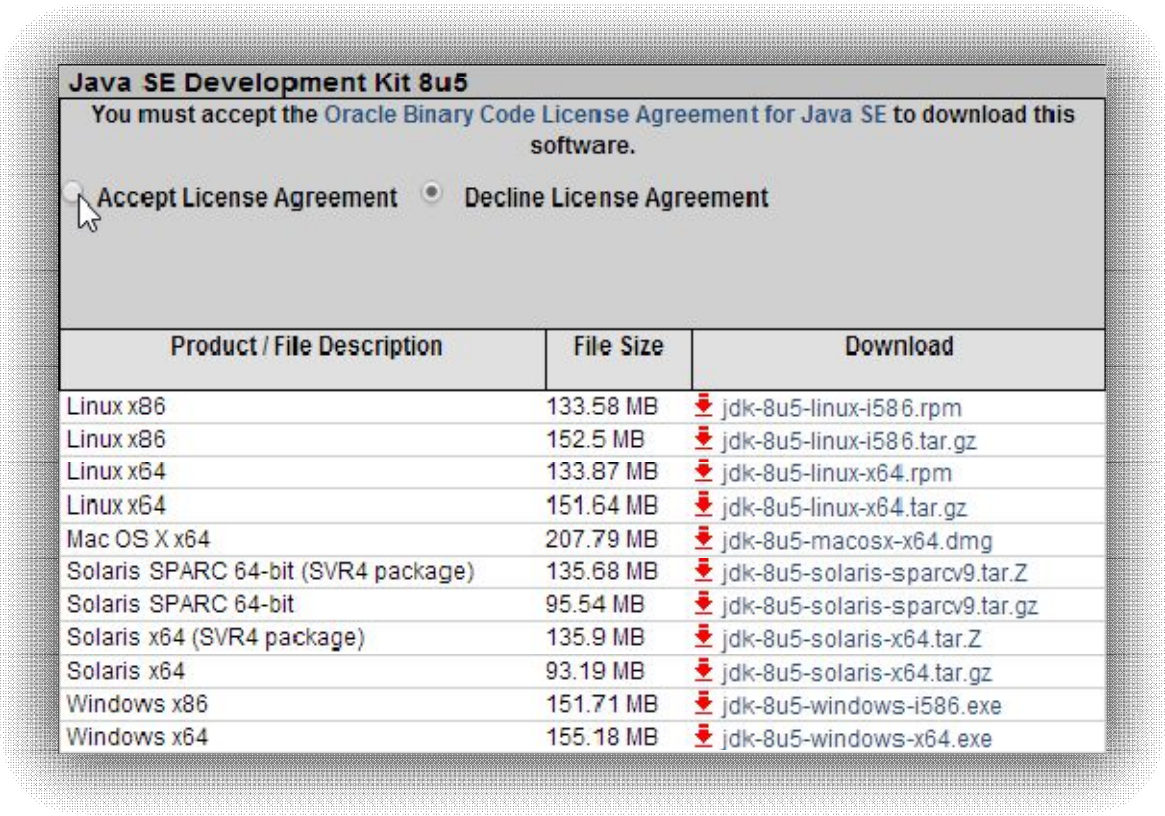
- ✓ Hétérogénéité du matériel.
- ✓ Puissance et mémoire limitées.
- ✓ Connectivité à internet (disponibilités, rapidités,…)
- ✓ Dispositifs d'affichage nombreux et variés
- ✓ Interface tactile
- ✓ Interaction avec le système sans l'interrompre. [5]

II.2. Installation et configurationdu logiciel :

II.2.1. Installation :

Tout d'abord, Avant d'installer Android, on vous propose de vérifier que votre machine de développement contient les besoins suivants :

- 2 Go de mémoire RAM, mais on ne va pas se cacher qu'en dessous de 8 Go vous risquez d'être limité.
- Plus de 1,5 Go d'espace disque pour tout installer.
- Niveau processeur, l'émulation ne peut se faire que sur 1 cœur de votre processeur, donc augmenter le nombre de cœurs ne vous servira pas à grand-chose. C'est vraiment la puissance pure qui compte. Il n'y a donc pas de minima mais le plus rapide sera le mieux.
- Pour commencer tout d'abord faut installer le JDK (**Java Development Kit)**



Java SE Development Kit 8u5

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement Decline License Agreement

Product / File Description	File Size	Download
Linux x86	133.58 MB	jdk-8u5-linux-i586.rpm
Linux x86	152.5 MB	jdk-8u5-linux-i586.tar.gz
Linux x64	133.87 MB	jdk-8u5-linux-x64.rpm
Linux x64	151.64 MB	jdk-8u5-linux-x64.tar.gz
Mac OS X x64	207.79 MB	jdk-8u5-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	135.68 MB	jdk-8u5-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	95.54 MB	jdk-8u5-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	135.9 MB	jdk-8u5-solaris-x64.tar.Z
Solaris x64	93.19 MB	jdk-8u5-solaris-x64.tar.gz
Windows x86	151.71 MB	jdk-8u5-windows-i586.exe
Windows x64	155.18 MB	jdk-8u5-windows-x64.exe

Figure 2.1 : Téléchargement du JDK

Pour continuer l'installation, il faut accepter la licence du JDK. Choisissez ensuite la version adaptée à votre configuration. Une fois le téléchargement terminé, vous pouvez installer le tout là où vous le désirez.

- **Android Studio et le SDK Android** :

Nous allons maintenant télécharger un fichier qui contient un ensemble d'outils indispensables pour développer nos applications Android. Ce paquet contient Android Studio, un environnement de développement spécialisé dans le développement d'applications Android, et un outil pour gérer l'installation du SDK Android sur notresystème.

Une fois le téléchargement terminé, lancez l'installation.

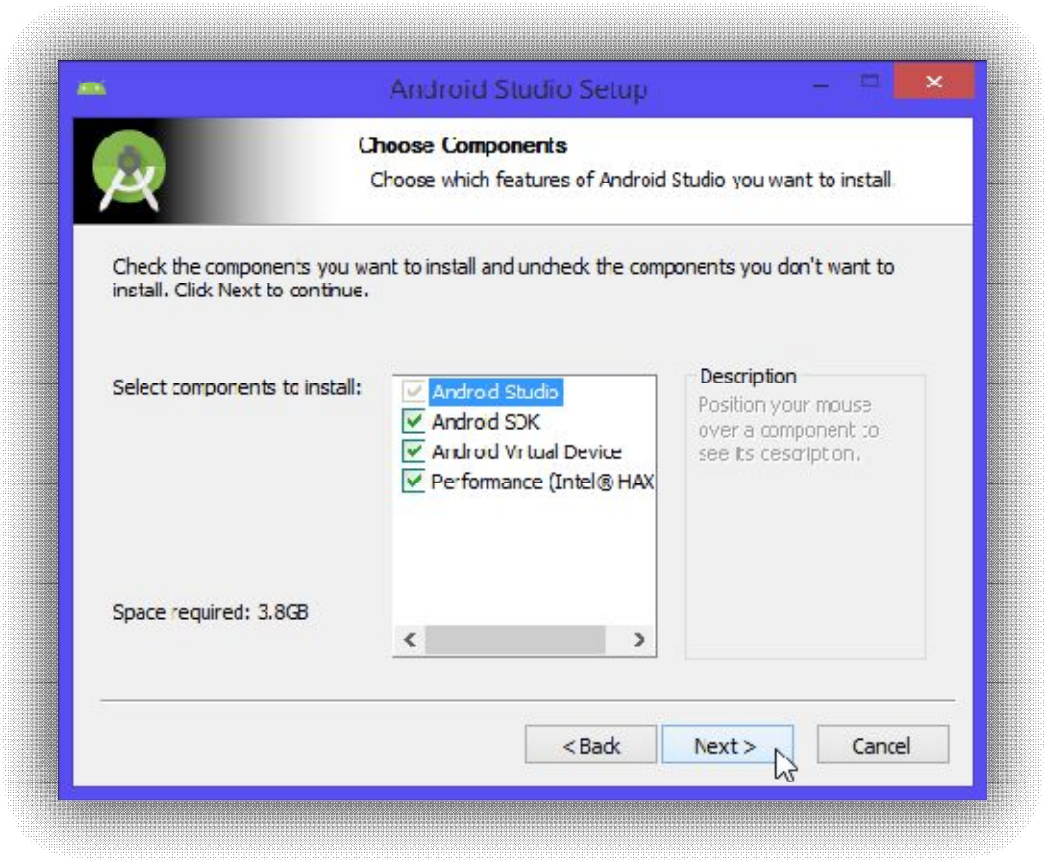


Figure 2.2: Installation du Android studio et SDK

Sélectionnez les options en fonction de votre système d'exploitation et votre configuration puis cliquez sur "Next"

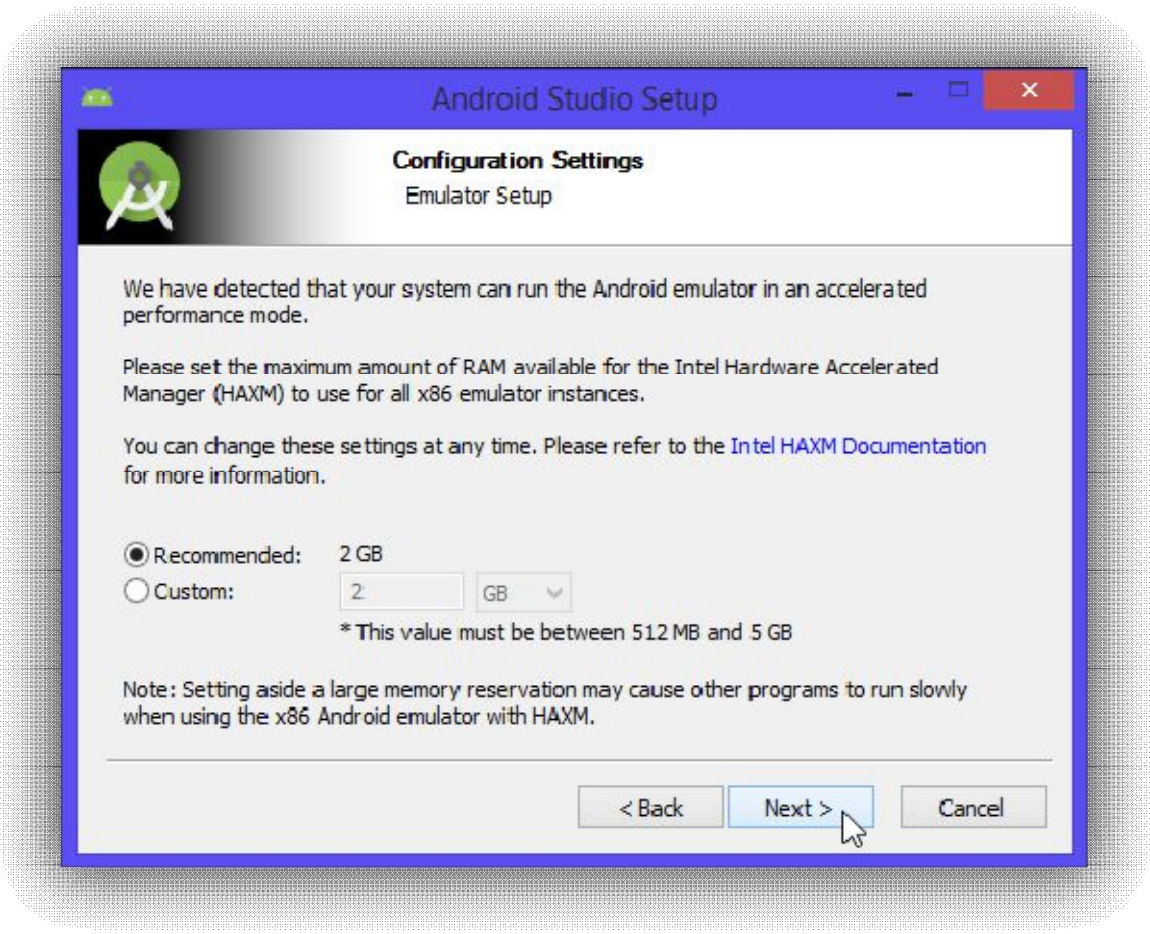


Figure 2.3 : Configuration de l'émulateur du logiciel.

Si votre ordinateur est suffisamment puissant, cet écran vous propose d'indiquer quelle quantité de mémoire vive (RAM) vous souhaitez accorder à l'émulateur Android. La valeur par défaut est 2 Go mais vous pouvez indiquer une valeur en choisissant Custom. Une fois votre sélection faite, cliquez sur le bouton « Next ».

Au premier lancement, une boîte de dialogue va s'afficher et vous demande si vous aviez déjà une version d'Android Studio installée précédemment. Si ce n'est pas le cas, sélectionnez la seconde option.

Figure 2.4 : Dernière étape pour finir l'installation

II.2.2. Configuration :

Nous avons maintenant installé Android sur notre machine, pour le bon fonctionnement de ce dernier, une configuration est nécessaire. La configuration est donnée dans les étapes ci-dessous :

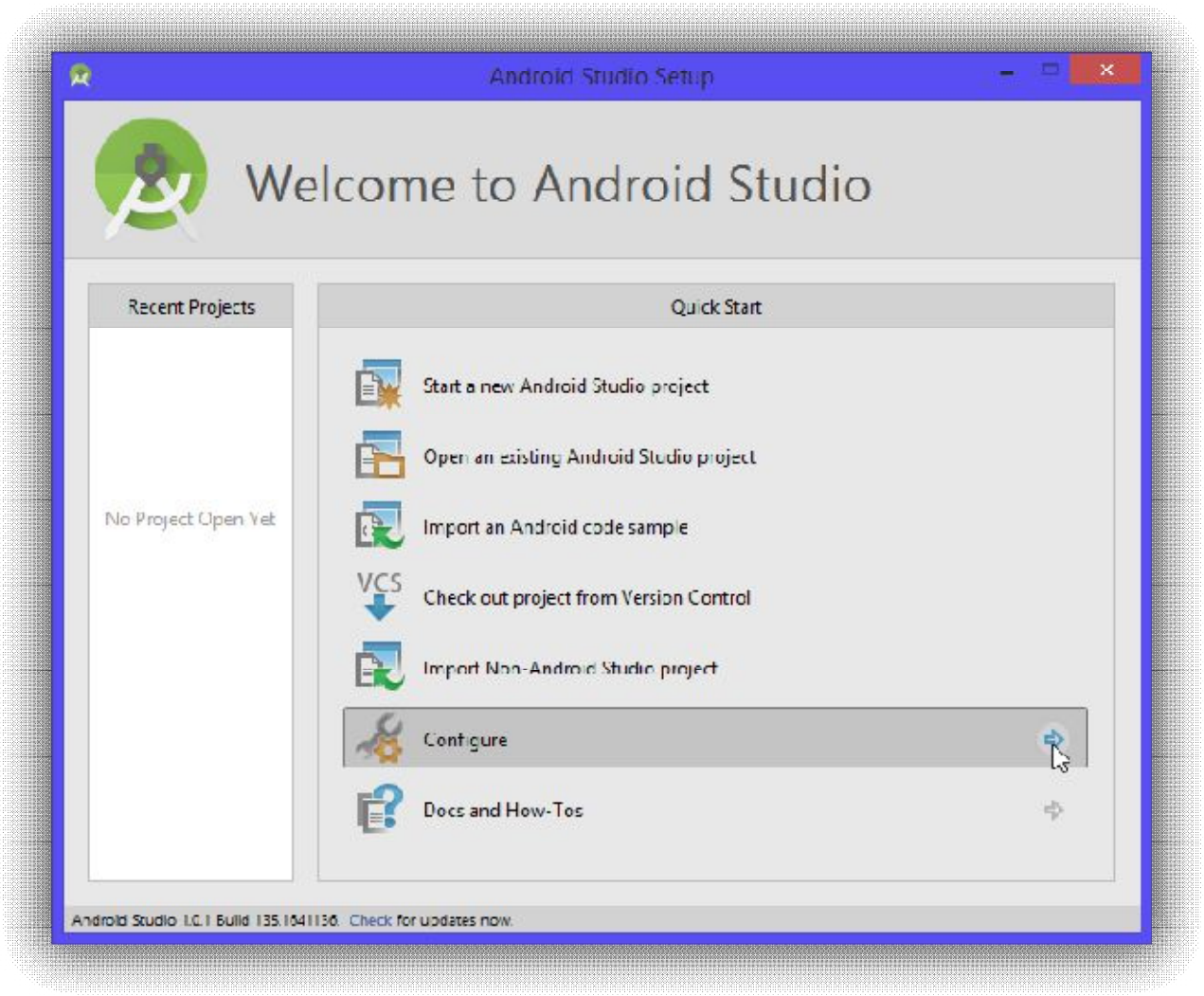


Figure 2.5 :Bienvenue a Android Studio.

- Une fenêtre s'ouvrira pour vous demander ce que vous souhaitez faire. Nous allons commencer par lui demander de télécharger le SDK d'Android.

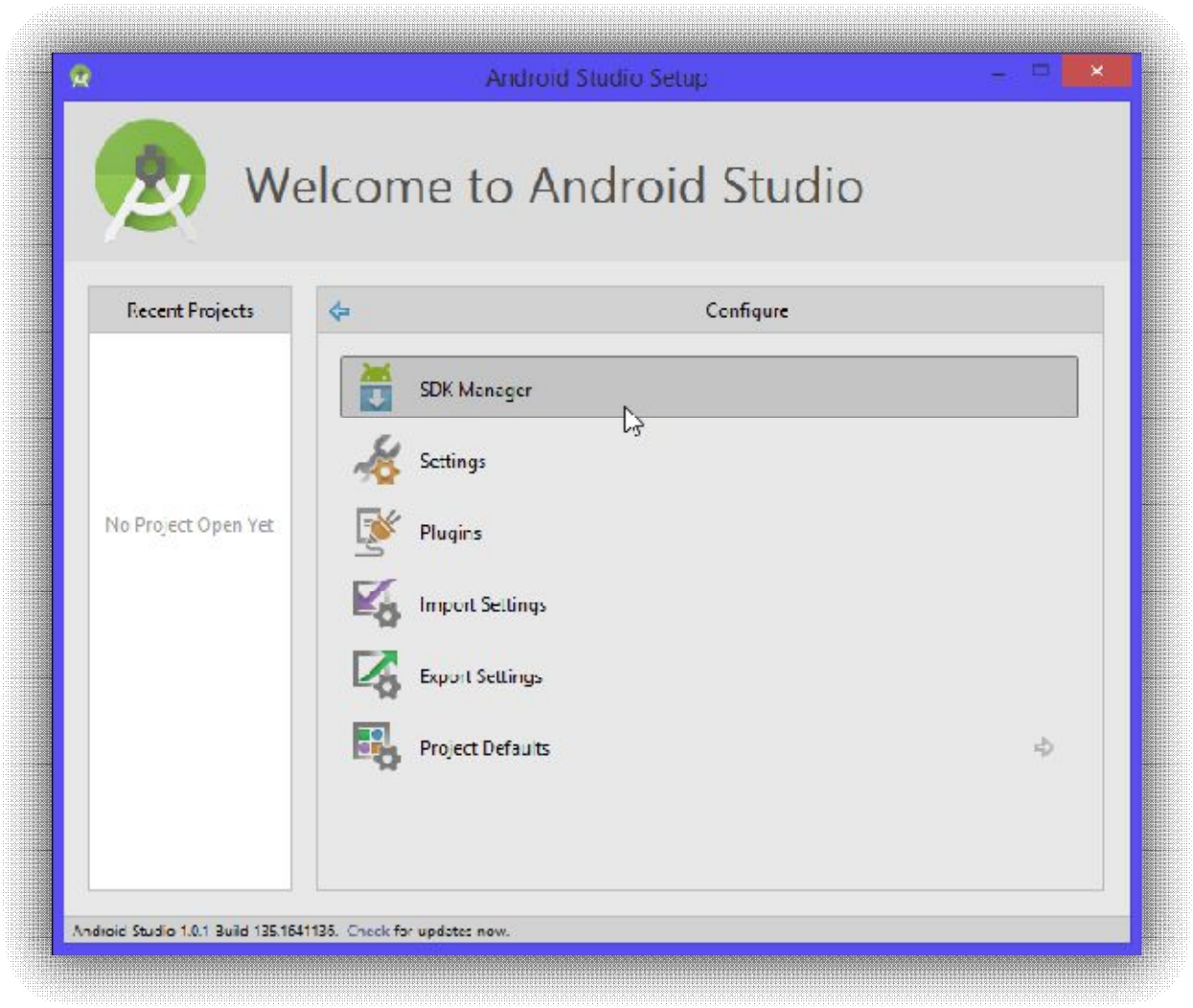


Figure 2.6 :L'ouverture du SDK.

L'Android SDK Manager s'ouvre et propose le menu affiché dans la figure ci-dessous :

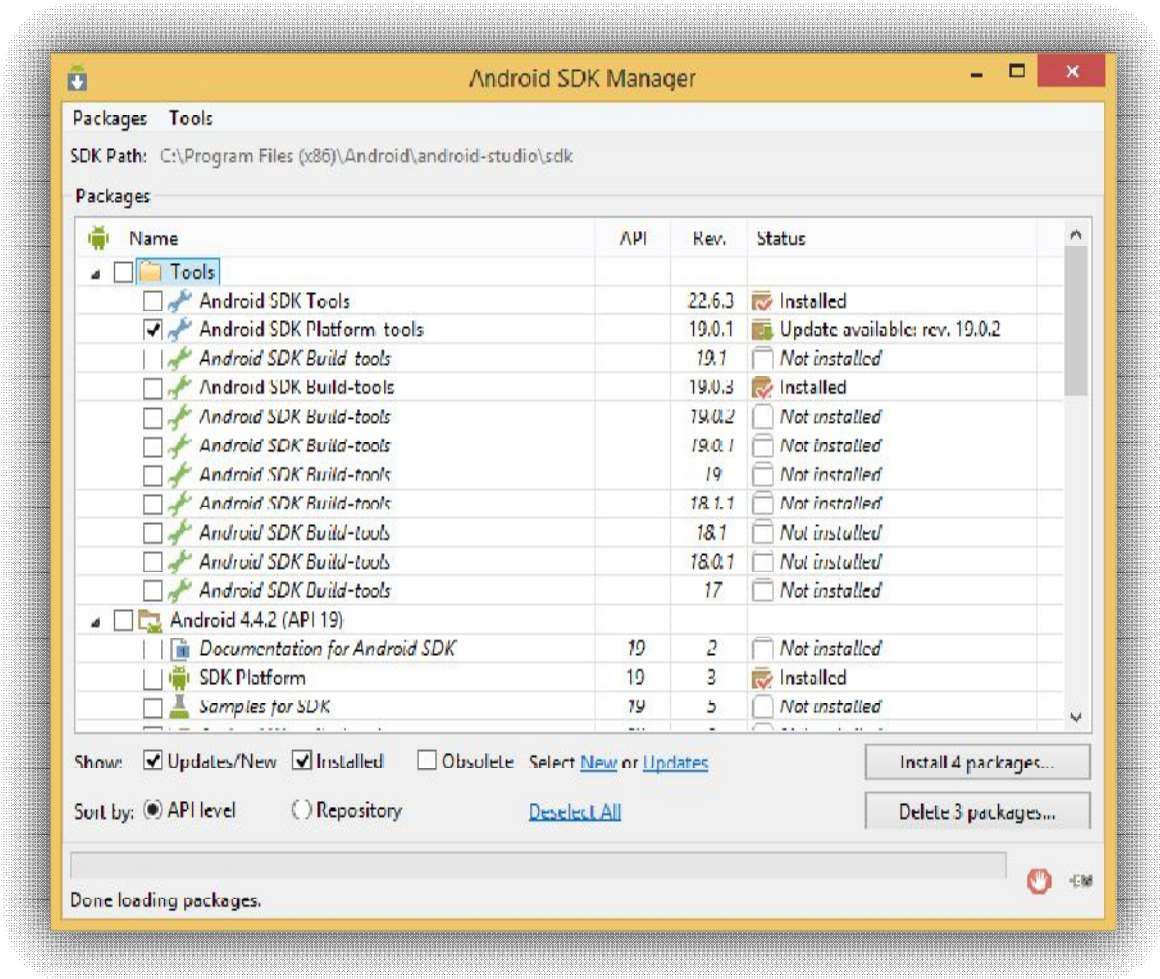


Figure 2.7 :Téléchargements des outils.

Dans ce tableau, vous trouvez deux types de lignes :

- Celles dont l'icône n'est pas un répertoire (une clé anglaise, un erlenmeyer, un petit Bugdroid, ...) correspondent à des paquets, c'est-à-dire des fichiers qui seront téléchargés pour ajouter de nouvelles fonctionnalités au SDK d'Android. A chaque ligne est associé un statut à l'aide de la colonne Status. Par exemple, vous pouvez voir que le paquet Android SDK Tools est déjà installé (Installed). En revanche, Documentation for Android SDK n'est pas installé (Not installed). Dans notre machine, Android SDK Platform-tools est installé, mais n'est pas à jour, l'Android SDK Manager nous l'indique avec le status ‘Update available’. Il se peut que nous trouvions plusieurs fois des paquets avec le même nom, dans ce cas c'est qu'il s'agit de versions différentes du même paquet, comme vous pouvez le voir dans la colonne Rev. : nous trouvons la version 19.1 d'Android SDK Platform-tools mais aussi la version 17 par exemple. Nous essayerons toujours d'avoir la dernière version de la plateforme.

- Les lignes dont l'icône est un répertoire représentent des groupes de paquets, qui appartiennent tous à une même catégorie. Nous trouvons la catégorie des outils (Tools) par exemple. Regardez le nom des autres groupes, vous remarquerez que certains suivent un certain motif. Il est écrit à chaque fois Android [un nombre] (API [un autre nombre]). La présence de ces nombres s'explique par le fait qu'il existe plusieurs versions de la plateforme Android qui ont été développées depuis ses débuts et qu'il existe donc plusieurs versions différentes en circulation. Le premier nombre correspond à la version d'Android et le second à la version de l'API Android associée.

Quand nous développons une application, il faut prendre en compte ces numéros, puisqu'une application développée pour une version précise d'Android fonctionnera sur les versions suivantes d'Android mais pas sur les versions précédentes. A des fins pédagogiques, Nous avons choisis de travailler avec une version assez récente d'Android pour vous présenter toutes les possibilités de développement. Mais dans vos développements réels, il vous faudra bien réfléchir à quelle version viser, en fonction du public visé et de ce que vous aurez besoin dans les API d'Android.Il vous faudra ensuite valider les licences pour les fichiers que vous allez télécharger :

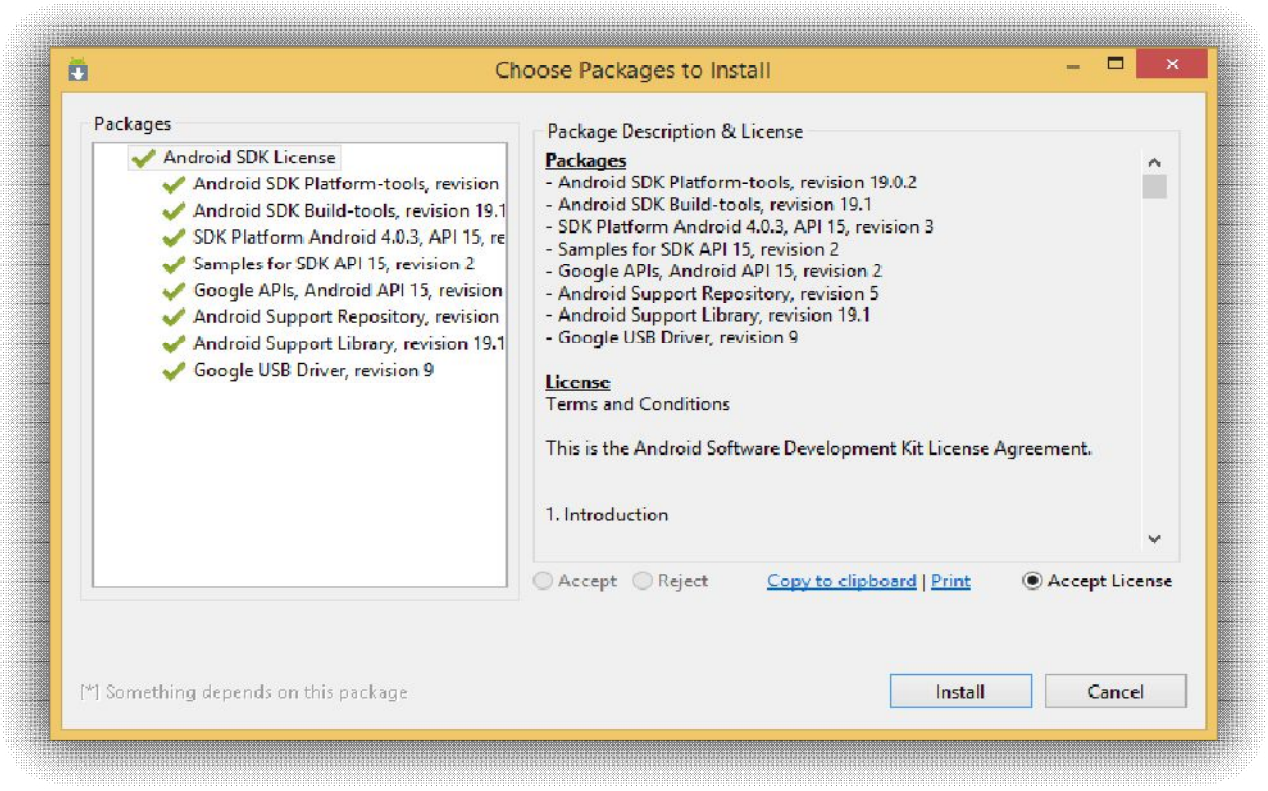


Figure 2.8 : License des outils .

Si vous installez tous ces paquets, vous aurez besoin de **1 Go** sur le disque de destination.

Maintenant nous avons installé et configurer Android sur notre machine.

II.2.3- Avantages d'Android

II.2.3.1. Les challenges :

Fonctionnalités d'Android :

- ✓ Framework d'application.
- ✓ Machine virtuelle Dalvik.
- ✓ Navigateur web intégré.
- ✓ API graphique 2D et 3D.
- ✓ SQlite.
- ✓ Codecs audio et vidéo.
- ✓ Wifi, EDGE, 3G, Bluetooth…….
- ✓ Caméra, GPS, Accéléromètre, Compass……

Contraintes :

- ✓ CPU cadencés 500-600 MHZ.
- ✓ Faible mémoire RAM disponible.
- ✓ Important temps d'accès (en écriture) sur disque flash.
- ✓ Cycle de vie spécifique des applications (autonomie énergétique, ressources limitées).
- ✓ Faible débit et intermittence des réseaux.
- ✓ Conception particulière des IHM :
- Ecran type : HVGA (320x480)
- Utilisation en mode portrait ou paysage.
- Texte de petites polices peuvent être non lisible (DPI).
- Faible résolution de touché de la dalle tactile (environ 25 pixels).

II.3. Le noyau :

Activité (Activity) :

- ✓ Brique de base d'une interface utilisateur.
- ✓ Equivalent d'une fenêtre (Windows, Linux) ou d'une boite de dialogue.
- ✓ Une activité peut éventuellement ne pas avoir d'interface utilisateur (cas des services par exemple).

Fournisseur de contenu (content provider) :

- ✓ Niveau d'abstraction pout toutes données stockées sur le terminal.
- ✓ Android encourage la mise à disposition de ses propres donnés aux autres programmes.
  - Le content provider le permet en proposant un contrôle sur la façon dont on accèdera aux données.

Intention (Intent) :

- ✓ Une intention est un message système qu'on peut qualifier d'évènement.
- ✓ Emis par le terminal pour prévenir les applications de la survenue des évènements (cas des évènements systèmes) ou par tout autre application (cas des évènements applicatifs).

- Système :
- ➢ Insertion d'une carte SD.
- ➢ Réception d'un SMS.
- ➢ …….

- Applicatif : (on peut imaginer)
- ➢ Un Intent '' le logiciel NetSpyR&T démarre ''.
- ➢ Un Intent '' L'utilisateur arrive à Paris '' en utilisant les informations de géolocalisation du terminal ''.

Service (Service) :

- ✓ Logiciel autonome prévu pour duré (contrairement aux activités, fournisseurs de contenus, récepteur d'intentions).
- ✓ Ne nécessite pas d'interface utilisateur.
- ✓ Exemples :

- Service vérifiant périodiquement des mises-à-jours de flux RSS.
- Service permettant d'écouter une playlist (indépendamment de toute activité).

Manifeste (Manifeste) :

- ✓ Point de départ de toute application Android.
- ✓ Permet de déclarer ce que l'application contient (activités, services,..).
- ✓ Précise comment ces composants sont reliés à Android (que fait-on apparaitre dans le menu ?....).
- ✓ Précise les permissions de l'application (contrôle de la Webcam, accès aux réseaux, accès aux services de localisation…).

Gadget graphique (Widget) :

- ✓ Terme résultant de la contraction des termes Window et Gadget.
- ✓ Concrètement c'est un composant d'interface graphique (libellés, champs de saisie, boutons….).

XML (XML) :

- ✓ Extensible Markup Langage (langage de balisage extensible).
- ✓ Langage de balisage extensible pour structurer des données.

Positionnement XML (XML Layout) :

- ✓ Permet de concevoir des interfaces plus simplement qu'en langage JAVA.
- ✓ Permet concrètement d'instancier les Widgets.
- ✓ Ce fichier est souvent généré par des outils qui permettent de construire graphiquement les interfaces.

Identifiant uniforme de ressource (Uniform Resource Identifier-URI) :

- ✓ Courte chaine de caractères identifiant une ressource sur un réseau réel ou abstrait.
- ✓ Respecte une norme d'Internet mise en place pour le web.
- ✓ Sont des URI :

- Les Uniform Resource Locator (URL) : identifie une ressource sur un réseau et fournit les moyens d'agir sur la ressource ou d'obtenir une présentation de la ressource en décrivant son mode d'accès primaire.
- Les Uniform Resource Name (URN) : identifie une ressource par son nom dans un espace de noms.

Conteneur (Container) :

- ✓ Permet de disposer un ensemble de Widgets pour obtenir la présentation désirée.
- ✓ La plupart des outils de construction d'interface graphique fournissent des gestionnaires de disposition (Layout manager) qui sont organisés le plus fréquemment en conteneurs.

Équipement Android Virtuel (Android Virtual Devices) :

- ✓ Les AVD permettent de simuler l'exécution d'un terminal Android sur un ordinateur.
- ✓ Ces terminaux sont personnalisables (Version d'Android, type de processeur, espace de stockage….).

- Simplifie le développement et la mise au point des applications.

Architecture :

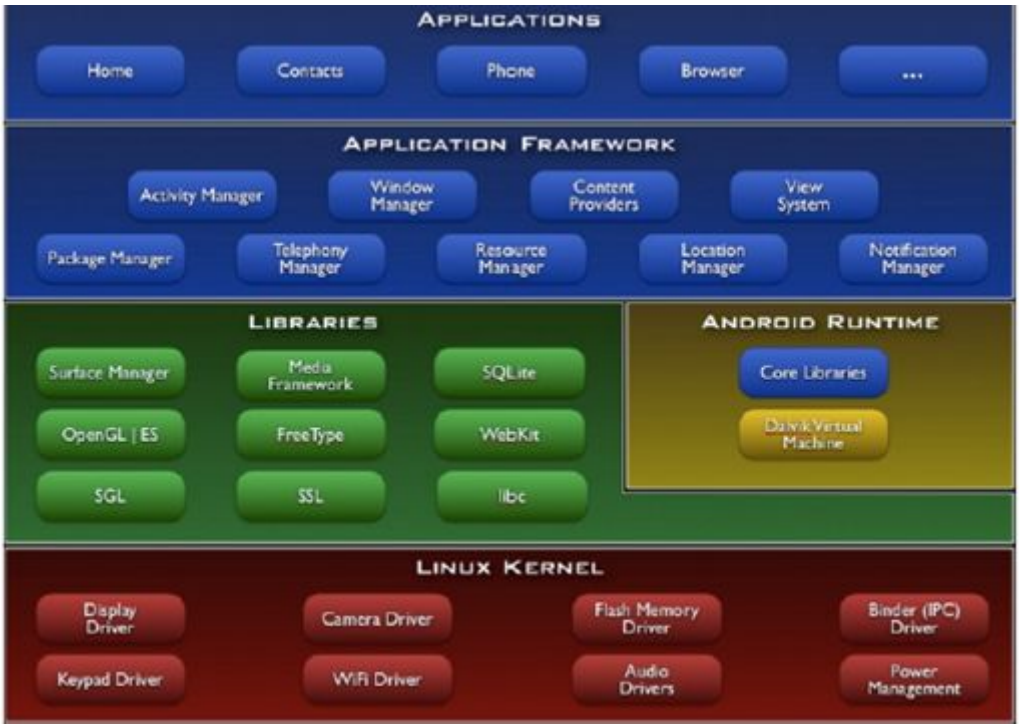


Figure 2.9 : Le noyau Android

II.4. Notre application :

II.4.1. Le principe :

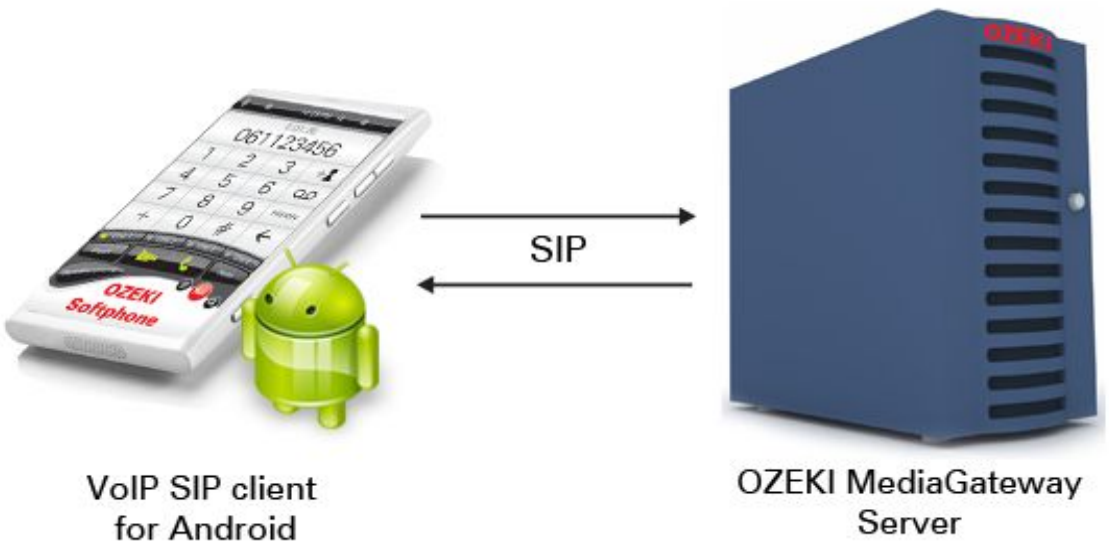


Figure 2.10 :Notre système de base

Notre système de base est très simple, nous avons développé un client IP qui peut communiquer facilement avec un serveur IP SIP via Internet.

II.4.2. notre interface Android :

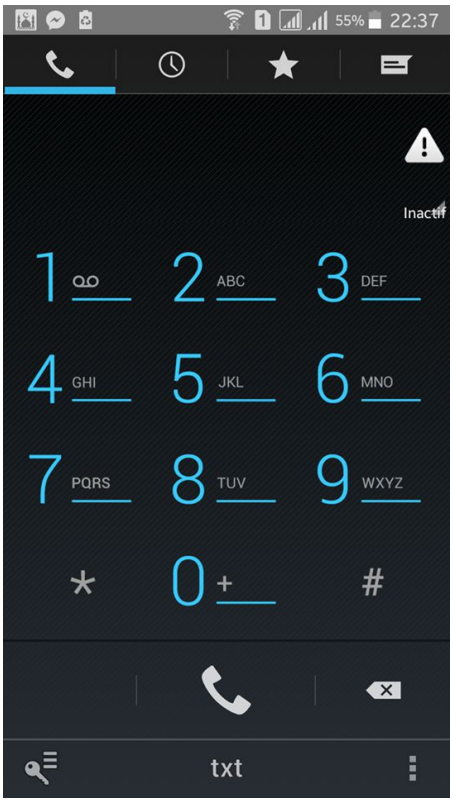


Figure 2.11 : Notre clavier de numérotation

Nous avons développé une interface très simple pour effectuer la numérotation vers le serveur SIP. Dans notre interface, nous avons le clavier numérique et un onglet pour la configuration

II.4.3. La configuration de notre client Android :

Pour le fonctionnement de notre application, un certain nombre de paramètres doivent être implémentés au niveau de notre client, pour être sur la même longueur d'onde avec le serveur IP distant. Les paramètres sont :

- Account name : Le nom de notre client
- Caller ID : Identifiant ou le numéro de notre client
- Server : Adresse IP et le port du serveur SIP
- User name : Le nom du serveur
- SIP ID : Identifiant ou le numéro du serveur
- Password : Le mot de passe pour accéder au serveur
- Use TCP : Utilisation du protocole TCP
- Proxy : Activer / Désactiver le proxy

Dans ce paragraphe, nous allons donner un exemple du code de source de comment généré notre interface de configuration.

Ce composant est utilisé pour permettre à l'utilisateur d'écrire des textes. Il s'agit en fait d'un TextView éditable

Figure 2.12 : Notre onglet de configuration

Pour des raisons de sécurité, le serveur SIP va nous générer des identifiants et des mots de passes pour rejoindre ce dernier, Après chaque connexion, le client et le serveur doivent passer par la période de vérification de l'identité. Si y'a l'identifiant ou le mot de passe ne correspondent pas, il est impossible de faire la liaison entre notre client et le serveur.

```
protected void onBindDialogView(View view) {
    super.onBindDialogView(view);
    try {
        if(showPwdCheckbox == null) {
            showPwdCheckbox = new CheckBox(getContext());
            showPwdCheckbox.setText(R.string.show_password);
            showPwdCheckbox.setOnClickListener(this);
        }

        canShowPassword = TextUtils.isEmpty(getText());
        getEditText().setInputType(
                InputType.TYPE_CLASS_TEXT | InputType.TYPE_TEXT_VARIATION_PASSWORD);
        updateCanShowPassword();
        ViewParent oldParent = showPwdCheckbox.getParent();
        if (oldParent != view) {
            if (oldParent != null) {
                ((ViewGroup) oldParent).removeView(showPwdCheckbox);
            }
        }

        ViewGroup container = (ViewGroup) view;
        if(Compatibility.isCompatible(8)) {
            container = (ViewGroup) container.getChildAt(0);
        }
```

La classe ci-dessous défini l'ajout d'un compte client / serveur pour se connecté a distance :

```
public class AccountChooserButton extends LinearLayout implements OnClickListener, View_HasStateListenerSupport {

    protected static final String THIS_FILE = "AccountChooserButton";
```

Figure 2.13 : Notre onglet de configuration

II.4.4. Configuration de notre application

Lors de notre projet, nous avons utilisé les valeurs suivant :

- Account name : Optionnel
- Caller ID : 6002
- Server : 192.168.43.6 :5060
- User name : Le nom du serveur
- SIP ID : 6001
- Password : secret
- Use TCP : Oui
- Proxy : Désactiver

II.4.5. Configuration du réseaux

Dans cette section, nous avons définit notre modèle du réseaux sur les différentes couches du modèle TCP/IP, pour plus de détails, merci de voir l'annexe.

II.5. Résultats : Pour testé notre client, nous avons fait un appel avec un serveur qui est developpé par une collegue a nous (Melle BENDELHOUM Selma).

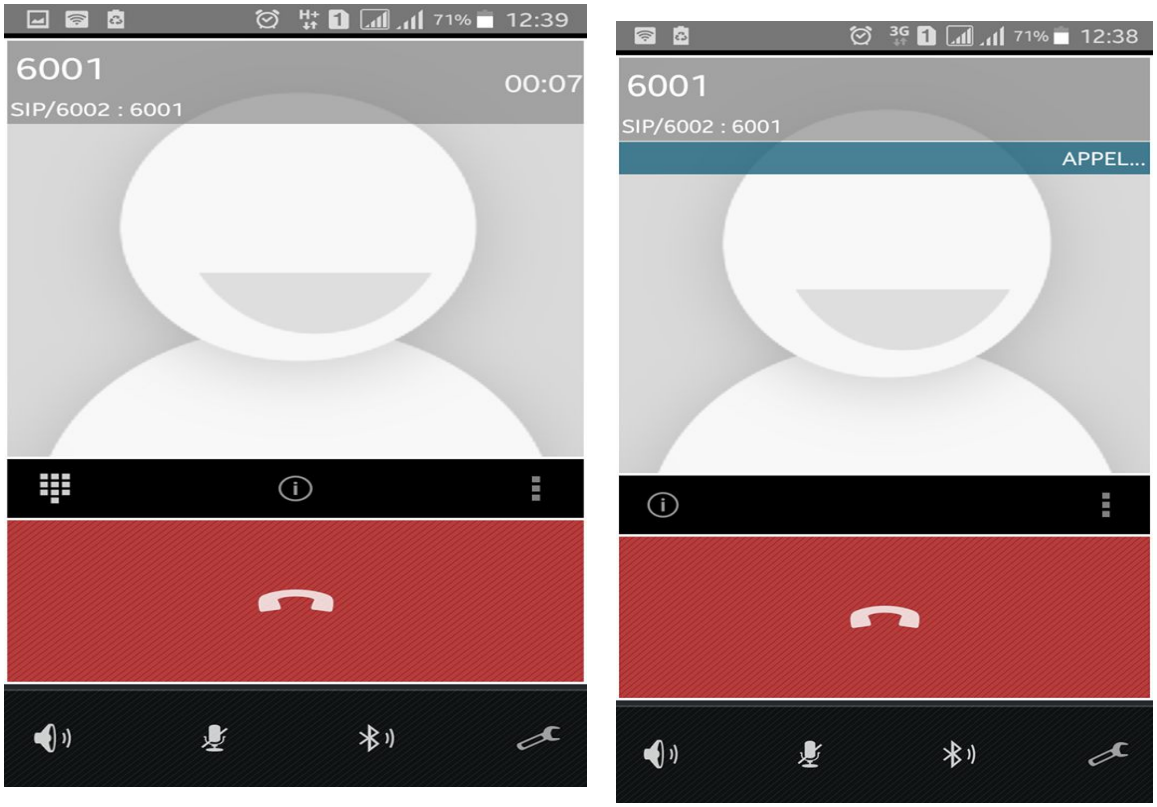


Figure 2.14 : Appel sortant du client vers le serveur

Nous remarquons que l'appel vers le serveur a bien marché, avec une durée de communication de sept secondes.

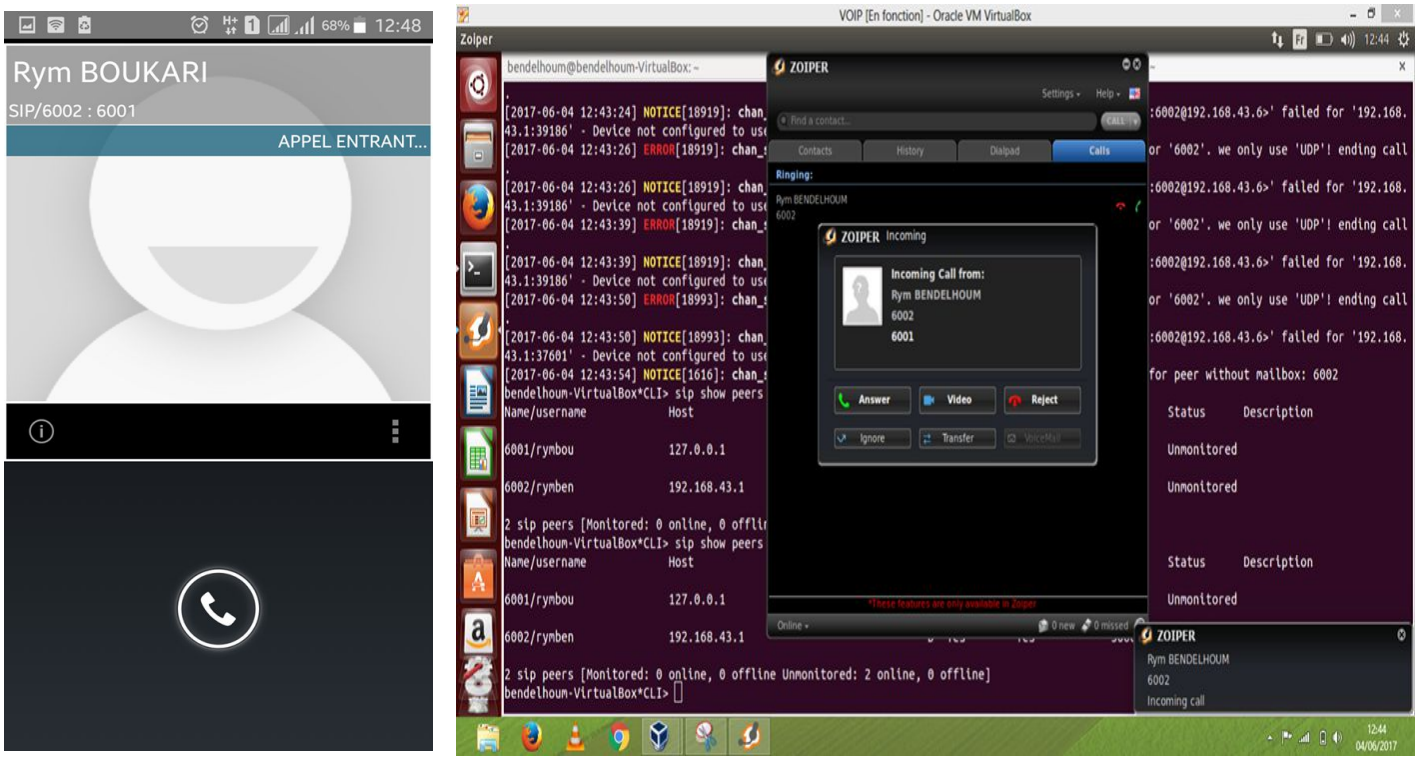


Figure 2.15 : Appel entrant du serveur vers notre client

Nous avons inversé les rôles, et testé un appel entrant. L'appel fonctionne parfaitement.

II.6. Conclusion

Dans ce chapitre nous étions intéressé a la partie client développé sous Android, après l'installation et la configuration de notre client, un teste a été nécessaire pour la vérification du bon fonctionnement de notre client.

Conclusion Générale

Conclusion générale

La téléphonie sur IP est une technologie révolutionnaire qui défie les règles édictées par la téléphonie RTC. Elle est plus souple, conviviale, ne nécessite pas un investissement lourd, coûte moins chère, propose de nouveaux services et beaucoup d'autres avantages, si bien que toute entreprise qui se veut compétitive et moderne aujourd'hui, jette son dévolu sur la téléphonie sur IP pour gérer ses communications tant internes qu'externes. Elle vise principalement à améliorer le cadre de travail des employés de l'entreprise en libérant l'utilisateur du lieu d'implantation du poste téléphonique. Actuellement, il est évident que la téléphonie sur IP va continuer à se développer dans les prochaines années. Le marché de la téléphonie sur IP bien que jeune encore se développe à une vitesse fulgurante. C'est la raison pour laquelle plusieurs entreprises dans leurs stratégies de développement investissent maintenant dans la téléphonie sur IP. Cela leur permettra à coup sûr de jouer un rôle majeur. La téléphonie sur IP ouvre aujourd'hui la voie de la convergence voix/données/image et celle de l'explosion de nouveaux services tels que les Centres d'appels actifs ou réactifs. Elle paraît comme une bonne solution en matière d'intégration, de fiabilité, d'évolutivité et de coût. Elle fait partie intégrante de l'Intranet de l'entreprise et permet même des communications à moindre coût. Nous n'avons pas la prétention d'avoir tout dit ou tout fait dans ce sujet. Nous avons posé des bases. Il appartient à la génération future de continuer la réflexion dans le domaine. Il y a beaucoup d'aspects qui restent en friche. Nous pouvons bien traiter un sujet qui consiste à savoir si : le développement de cette technologie représente un risque ou une opportunité pour les opérateurs traditionnels ? Une chose est certaine, l'utilisation massive d'Internet va augmenter le trafic et mettre à disposition de nouveaux services que pourront développer les opérateurs. Bientôt nous téléphonerons tous sur IP. Le protocole IP est devenu un standard unique permettant l'interopérabilité des réseaux mondialisés. D'aucuns pensent d'ailleurs que l'intégration de la voix sur IP n'est qu'une étape vers le tout IP.

Annexe

1.Configuration du compte

```
import org.pjsip.pjsua.pjsua_ipv6_use;
import org.pjsip.pjsua.pjsua_stun_use;
import org.pjsip.pjsua.pjsua_transport_config;
import org.pjsip.pjsua.pjsua_turn_config;
import org.pjsip.pjsua.pjsua_turn_config_use;

public class PjSipAccount {

    //private static final String THIS_FILE = "PjSipAcc";

    private String displayName;
    // For now everything is public, easiest to manage
    public String wizard;
    public boolean active;
    public pjsua_acc_config cfg;
    public csipsimple_acc_config css_cfg;
    public Long id;
    public Integer transport = 0;
    private int profile_vid_auto_show = -1;
    private int profile_vid_auto_transmit = -1;
    private int profile_enable_qos;
    private int profile_qos_dscp;
    private boolean profile_default_rtp_port = true;

    //private boolean hasZrtpValue = false;


    public PjSipAccount() {
        cfg = new pjsua_acc_config();
        pjsua.acc_config_default(cfg);

        css_cfg = new csipsimple_acc_config();
        pjsua.csipsimple_acc_config_default(css_cfg);
    }
```

2.La suite du configuration du compte

```
public PjSipAccount(SipProfile profile) {
    this();

    if(profile.id != SipProfile.INVALID_ID) {
        id = profile.id;
    }

    displayName = profile.display_name;
    wizard = profile.wizard;
    transport = profile.transport;
    active = profile.active;
    transport = profile.transport;

    cfg.setPriority(profile.priority);
    if(profile.acc_id != null) {
        cfg.setId(pjsua.pj_str_copy(profile.acc_id));
    }
    if(profile.reg_uri != null) {
        cfg.setReg_uri(pjsua.pj_str_copy(profile.reg_uri));
    }
    if(profile.publish_enabled != -1) {
        cfg.setPublish_enabled(profile.publish_enabled);
    }
    if(profile.reg_timeout != -1) {
        cfg.setReg_timeout(profile.reg_timeout);
    }
    if(profile.reg_delay_before_refresh != -1) {
        cfg.setReg_delay_before_refresh(profile.reg_delay_before_refresh);
    }
    if(profile.ka_interval != -1) {
        cfg.setKa_interval(profile.ka_interval);
    }
    if(profile.pidf_tuple_id != null) {
        cfg.setPidf_tuple_id(pjsua.pj_str_copy(profile.pidf_tuple_id));
    }
    if(profile.force_contact != null) {
```

3.code source de la couche transport (UDP et TCP)

```
public void applyExtraParams(Context ctxt) {

    // Transport
    String regUri = "";
    String argument = "";
    switch (transport) {
    case SipProfile.TRANSPORT_UDP:
        argument = ";transport=udp;lr";
        break;
    case SipProfile.TRANSPORT_TCP:
        argument = ";transport=tcp;lr";
        break;
    case SipProfile.TRANSPORT_TLS:
        //TODO : differentiate ssl/tls ?
        argument = ";transport=tls;lr";
        break;
    default:
        break;
    }

    if (!TextUtils.isEmpty(argument)) {
        regUri = PjSipService.pjStrToString(cfg.getReg_uri());
        if(!TextUtils.isEmpty(regUri)) {
            long initialProxyCnt = cfg.getProxy_cnt();
            pj_str_t[] proxies = cfg.getProxy();

            //TODO : remove lr and transport from uri
    //      cfg.setReg_uri(pjsua.pj_str_copy(proposed_server));
            String firstProxy = PjSipService.pjStrToString(proxies[0]);
            if (initialProxyCnt == 0 || TextUtils.isEmpty(firstProxy)) {
                cfg.setReg_uri(pjsua.pj_str_copy(regUri + argument));
                cfg.setProxy_cnt(0);
            } else {
                proxies[0] = pjsua.pj_str_copy(firstProxy + argument);
                cfg.setProxy(proxies);
            }
```

4.code du DNS

```
    // DNS
    if (prefsWrapper.enableDNSSRV() && !prefsWrapper.useIPv6()) {
        pj_str_t[] nameservers = getNameservers();
        if (nameservers != null) {
            cfg.setNameserver_count(nameservers.length);
            cfg.setNameserver(nameservers);
        } else {
            cfg.setNameserver_count(0);
        }
    }
```

5.Code source UDP

```
// UDP
if (prefsWrapper.isUDPEnabled()) {
    int udpPort = prefsWrapper.getUDPTransportPort();
    localUdpAccPjId = createLocalTransportAndAccount(
            pjsip_transport_type_e.PJSIP_TRANSPORT_UDP,
            udpPort);
    if (localUdpAccPjId == null) {
        cleanPjsua();
        return false;
    }
    // UDP v6
    if (prefsWrapper.useIPv6()) {
        localUdp6AccPjId = createLocalTransportAndAccount(
                pjsip_transport_type_e.PJSIP_TRANSPORT_UDP6,
                udpPort == 0 ? udpPort : udpPort + 10);
    }
}
```

6.Code source TCP

```
// TCP
if (prefsWrapper.isTCPEnabled()) {
    int tcpPort = prefsWrapper.getTCPTransportPort();
    localTcpAccPjId = createLocalTransportAndAccount(
            pjsip_transport_type_e.PJSIP_TRANSPORT_TCP,
            tcpPort);
    if (localTcpAccPjId == null) {
        cleanPjsua();
        return false;
    }

    // TCP v6
    if (prefsWrapper.useIPv6()) {
        localTcp6AccPjId = createLocalTransportAndAccount(
                pjsip_transport_type_e.PJSIP_TRANSPORT_TCP6,
                tcpPort == 0 ? tcpPort : tcpPort + 10);
    }
}
```

Référence

- [1] (Projet devoir final VOIP avec ASTERIS)
- **[2] (COMPANEO standard téléphonique)**
- [3] (https://www.frameip.com/tcpip. https://protocole-tcpip.over-blog.com/article-le-protocole-tcp-ip-definition-69189059.html).
- [4], [8] (mémoire de master ToIP et Amélioration de la QoS1 [monsieur Bemmoussat])
- [5] (Développement mobile sous Android Fabien Teytaud Université du Littoral Cote d'Opale 1er septembre 2014)
- [6] (programmation sous android, Gauthier Picard Ecole Nationale Supérieure des Mines 2012 Cette présentation a été conçue par
- **Jean-Paul Jamont** (Université Pierre Mendes France, IUT de Valence)
- [7] (PANASONIC France 2017)
- [9] (https://www.memoireonline.com /optimization de la telephonie par la VOIP)

Résumé :

La voix sur IP (Voice over IP) est une technologie de communication vocale en pleine émergence. Elle fait partie d'un tournant dans le monde de la communication. En effet, la convergence du triple play (voix, données et vidéo) fait partie des enjeux principaux des acteurs de la télécommunication aujourd'hui. Dans ce mémoire, nous nous sommes basé spécialement sur le développement d'une application qui utilise la technologie IP sous Android Studio tout en suivant plusieurs étapes afin d'assurer la configuration la notification de coté client avec un serveur SIP.

Mot clés : TCP/IP, VoIP, Android studio,SIP.

Abstract :

The voice over IP (Voice over IP) is a technology of vocal communication in full emergence. Sheisamember of a turningbend in the world of the communication. Indeed, the convergence of the triple play (voices, data and video) isa part of main stakes in the actors of the telecommunicationtoday. In this report we are speciallybased on the development of an application which uses the VoIPtechnologyunder Android Studio whilefollowingseveral stages to assure the configuration the notification of quotedcustomerwith a server SIP.

Keywords :TCP/IP,VoIP,Android,SIP