Alma Mater Studiorum · Università di Bologna

facoltà di scienze matematiche, fisiche e naturali
Corso di Laurea Magistrale in Informatica

Applicazioni Voip per Android: Analisi

Tesi di Laurea in Architettura degli Elaboratori

Relatore:
Chiar.mo Prof.
Ghini Vittorio

Presentata da:
Di Lullo Giuseppe

Sessione I
Anno Accademico 2011/2012

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Lo sviluppo delle connessioni Internet a banda larga e l'utilizzo di piattaforme tecnologiche innovative come i sistemi wireless hanno comportato uno spostamento del traffico telefonico dalle reti tradizionali a commutazione di circuito a reti basate sul protocollo Internet. In questo scenario, la telefonia via Internet (VoIP) costituisce uno dei servizi a più alto potenziale. Si tratta sostanzialmente di una tecnologia di trasmissione a commutazione di pacchetto: i segnali di voce vengono convertiti in pacchetti di dati digitali, spediti nella rete Internet e assemblati a destinazione.

Nel settore delle comunicazioni sono state sviluppate diverse applicazioni che consentono agli utenti di effettuare chiamate telefoniche attraverso reti mobili o wireless LAN utilizzando Internet.

Le possibili configurazioni dei servizi VoIP si distinguono, in funzione dei terminali di fruizione, in:

- **PC-to-PC:** Questa configurazione rappresenta la forma più semplice per effettuare le chiamate su IP. Il terminale è emulato tramite un softclient che viene installato su PC, laptop, PDA (Personal Digital Assistent) e qualsiasi altro dispositivo dotato di microfono e altoparlanti e che sia collegato alla rete IP. Il cliente utilizza il softclient per selezionare la persona con cui intende parlare da una lista di contatti e il client attiva la sessione utilizzando direttamente l'indirizzo IP del client che riceve la chiamata. Queste chiamate, in genere, viaggiano sulla Internet pubblica con qualità best effort e possono essere del tut-

to indipendenti dai SP (Service Provider) che offrono la connettività IP.

- **PC-to-Phone:** in questa modalità un end-point è costituito dal PC dotato di softclient e l'altro è rappresentato da un telefono.
- **Phone-to-Phone:** in questa configurazione i terminali ai capi della connessione sono entrambi dei telefoni (IP-Phone).

L'analisi che verrà fatta in questo lavoro di tesi prenderà in considerazioni lo scenario Phone-to-Phone e in base a dei parametri verrà valutata l'efficenza di alcune applicazioni open-source disponibili in rete. In particolare verrà vedremo come la presenza di firewall rappresenti ancora un limite nelle attuali applicazioni.

Il documento di tesi sarà così strutturato. Nel primo capito si descriverà una panoramica della tecnologia Voip e del protocollo SIP utilizzato per stabilire questo tipo di connessioni multimediali. Successivamente sarà descritta la struttura della piattaforma Android.

Nel secondo capitolo andremo ad esaminare le applicazioni che sono state prese in considerazione, descrivendone i principi di funzionamento e le caratteristiche inplementative.

Nel terzo capitolo verrà messe in evidenza i parametri sui quali sarà effettuata l'analisi coi relativi test, che daranno un'idea dettagliata delle problematiche che attualmente vengono riscontrate in questo particolare contesto di comunicazione.

Il quarto capito rappresenterà la parte inplementativa del progetto, come è stata strutturata e l'ambiente di sviluppo utilizzato. Infine verrà fatto un sunto del lavoro svolto e i possibili sviluppi futuri.

1.1 Voip

Il Voice over Internet Protocol (VoIP)[1], è una tecnologia tramite la quale possibile effettuare una conversazione telefonica sfruttando una connessione ad Internet, o anche una rete dedicata che utilizzi il protocollo IP, senza passare attraverso la rete telefonica tradizionale (PSTN). Quindi si intende l'insieme dei protocolli di comunicazione di strato applicativo che rendono possibile tale tipo di comunicazione.

Il vantaggio principale che si trae dall'utilizzo del VoIP riguarda il minor costo delle chiamate rispetto ai servizi di telefonia tradizionale. Telefonare utilizzando il VoIP significa: spesa nulla per le chiamate verso utenti facenti capo allo stesso VoIP provider, e nel peggiore dei casi una spesa molto ridotta per le chiamate verso altre destinazioni geografiche (specialmente per le lunghe distanze). Grazie al VoIP viene inoltre garantita la portabilità del numero a prefisso geografico, infatti il numero non è più legato fisicamente ad una linea telefonica e dal punto di vista dell'utente non cambia nulla, in quanto si utilizzerà sempre un normale telefono.

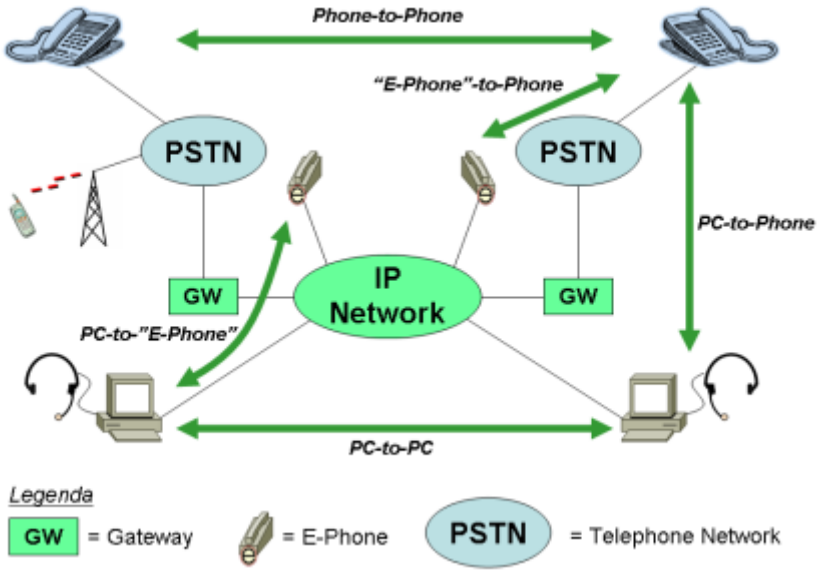

Figura 1.1: Elementi terminali coinvolti nelle comunicazioni VoIP

In questo tipo di comunicazioni basate su Internet, e di conseguenza sullo stack TCP/IP, i problemi principali che possono manifestarsi riguardano la qualità della trasmissione dei dati e la gestione dei pacchetti trasmessi. Infatti i dati in trasmissione vengono suddivisi in pacchetti, e spediti attraverso Internet, per essere poi ricostruiti in ricezione. Le difficoltà che possono sorgere nel caso di utilizzo della tecnologia VoIP, per la comunicazione vocale, sono legate alla latenza e al jitter (è necessario ridurre e mantenere costante il tempo di transito e di elaborazione dei dati durante le conversazioni), e all'integrità dei dati (è necessario evitare le perdite d'informazione dovute alla perdita dei pacchetti). Diminuire il tempo di latenza significa aumentare la velocità di transito delle informazioni, ed evitare possibili pause nella conversazione; mentre migliorare l'integrità dei dati significa evitare la perdita dei pacchetti, da cui conseguirebbe una perdita di parole e/o di frasi durante la conversazione.

La tecnologia VoIP utilizza due protocolli di comunicazione che funzionano in parallelo, uno utilizzato per il trasporto dei dati (pacchetti voce su IP) e nella grande maggioranza delle implementazioni di VoIP viene impiegato il protocollo RTP (Real-Time Transport Protocol). L'altra tipologia di protocollo è necessaria per la segnalazione che assiste la conversazione. I protocolli di segnalazione più utilizzati sono H.323 e SIP (Session Initiation Protocol) formulati rispettivamente dall'ITU (International Telecommunications Union) e dall"IETF (Internet Engineering Task Force).

H.323 [2] fu progettato con l'obiettivo principale di promuovere uno standard per gli utenti delle sessioni di comunicazione audio, video e dati attraverso reti non orientate alla connessione dove non è garantita la qualità di servizio (QoS), come sono le reti IP. Implementando l'H.323, i prodotti e le applicazioni multimediali di differenti produttori possono operare congiuntamente senza problemi di compatibilità.

1.2 Il Protocollo SIP

Session Initiation Protocol (SIP) [3] è un protocollo di segnalazione definito dalla IETF (Internet Engineering Task Force) nel Marzo 1999, poi aggiornato dalla RFC 3261 del Giugno 2002, fa parte dell'Internet Multimedia Conferencing Suite. SIP è basato su IP, ed è impiegato principalmente per applicazioni di telefonia VoIP; ma esso trova applicazione anche nei servizi telefonici supplementari, nella video-comunicazione, nei videogiochi interattivi, e nella messaggistica istantanea.

Ci sono molte applicazioni in internet che richiedono l'instaurazione di una sessione e la sua gestione dove sessione è considerata uno scambio di dati tra più partecipanti. Le implementazioni di queste applicazioni sono complicate dalle abitudini che hanno i partecipanti: gli user possono spostarsi tra più end point, possono essere contattati attraverso più nomi, e possono comunicare attraverso differenti media, qualche volta simultaneamente. Sono stati creati numerosi protocolli per trasportare le varie forme di trasmissioni multimediali real-time come video, voce, o messaggi di testo. Il SIP lavora con questi protocolli per permettere agli endpoints(chiamati user agent) di trovarsi l'uno con l'altro e accordarsi sul tipo di sessione che vogliono instaurare. Per localizzare i potenziali partecipanti alle sessioni, e per altre funzioni, il SIP permette la creazione di un infrastruttura di server (chiamata proxy servers) a cui l'utente può inviare richieste di registrazione, instaurazione di sessione ed altro. Il SIP è un potente mezzo per creare, modificare e terminare sessioni indipendentemente dal trasport protocol e senza dipendere dal tipo di sessione che sarà stabilita. Inoltre, SIP favorisce un'architettura modulare e scalabile, ovvero capace di crescere con il numero degli utilizzatori del servizio. Queste potenzialità hanno fatto sì che questo protocollo di segnalazione sia, ad oggi, il protocollo più diffuso, nel mercato residenziale e business, per il supporto al VoIP.

Intorno a SIP sono sorti diversi tipi di utilizzatori, pensati per facilitare la fruizione della telefonia VoIP. Alcuni esempi sono: gli ATA (Analog Telephone Adapter), capaci di convertire la segnalazione elettrica di un normale telefono analogico in un flusso di dati IP; e gli IP Phone, dall'unione tra telefono tradizionale e ATA, i quali sono telefoni dalle funzionalità elevate, e ai

quali non arriva il doppino telefonico ma i cavi di rete. Evoluzione ulteriore sono i softphone, applicazioni software per personal computer che emulano le funzioni di un telefono VoIP.

1.2.1 Descrizione

Il Sip supporta cinque caratteristiche di creazione e terminata delle chiamate:

- *User location:* determina se l'end user è disponibile per la comunicazione;
- *User availability:* determina la disponibilità degli utenti a instaurare la comunicazione;
- *User capabilities:* determina i parametri dei media da usare;
- *Session setup:* ringing, stabilisce i parametri della sessione che devono usare entrambi i chiamanti;
- *Session management:* include il trasferimento e il termine della sessione, modifica dei parametri di sessione e invocazione dei servizi.

Il SIP non è un sistema integrato di comunicazione verticale bensì è un componente che può essere utilizzato con altri protocolli per costruire un'architettura multimediale completa. Tipicamente, quest'architettura include protocolli come il Real-time Trasport Protocol (RTP) per trasportare i dati in tempo reale e fornire Quality of Service, il Real-time streaming protocol(RTSP) [4] per controllare la consegna dello streaming, il Media Gateway Control Protocol (MEGACO) per controllare i gateways alla rete telefonica pubblica (PSNT), e il Session Description Protocol (SDP) per descrivere le sessioni multimediali, quindi gestire le caratteristiche di ogni tipologia di sessione. Per cui, il SIP se è usato con altri protocolli è in grado di fornire un servizio completo agli utenti. Comunque, le funzionalità di base e operative del SIP non dipendono da nessuno di questi protocolli.

Il protocollo più usato nel trasporto dei messaggi di segnalazione si chiama UDP (User Datagram Protocol) , con porta di default 5060. Le revisioni di questo standard però ne permettono anche l'utilizzo attraverso TCP e TLS.

La decisione di utilizzare TCP al posto di UDP viene presa dallo *user agent client* (il terminale mittente), quando ad esempio la dimensione del messaggio è tale da non rendere possibile l'utilizzo di un singolo pacchetto UDP, oppure nell'eventualità che la comunicazione sia ritenuta a priori inaffidabile. Il TLS è invece utilizzato per rendere sicuro il trasporto; in questo caso la SIP URI di destinazione userà lo schema *sips* anziché *sip*. Nel caso del trasporto UDP, SIP fa uso di una macchina a stati che definisce i parametri della modalità di ritrasmissione seguente. Se lo *user agent client* non riceve risposta ad un suo messaggio entro un tempo T_1 (posto a 500 msec), lo re-invia di nuovo, e raddoppia il valore del timer T_1. Al nuovo scadere di T_1 ripete ancora l'invio del messaggio, e raddoppia nuovamente T_1, e così via (ritrasmette e raddoppia il *timer*) finché:

- non riceve una risposta valida, oppure
- è trascorso più di 1 minuto dal primo invio, oppure
- viene ricevuto un errore di tipo ICMP.

In questo modo, mentre si sfruttano i vantaggi di velocità legati all'utilizzo di UDP, si aggiunge una funzione di affidabilità al trasporto, che permette di gestire il caso dei messaggi perduti.

Le funzioni fondamentali del protocollo SIP sono:

- la localizzazione degli utenti e l'acquisizione delle preferenze di questi ultimi;
- l'invito degli utenti a partecipare ad una sessione, effettuando una negoziazione delle *capability*, e trasportando una descrizione della sessione;
- l'instaurazione delle connessioni di sessione;
- la gestione di eventuali modifiche dei parametri di sessione;
- il rilascio delle parti;
- la cancellazione della sessione in qualunque momento. Inoltre, grazie ad alcune estensioni, il protocollo può:

- pubblicare ed aggiornare le informazioni di presenza;
- richiedere il trasporto di informazioni di presenza;
- notificare l'evento di presenza;
- trasportare messaggi istantanei.

Come già detto, SIP non si occupa del trasporto dei media (audio/video), ma soltanto della segnalazione. La comunicazione vera e propria avviene tramite altri protocolli, tra i quali il più utilizzato è RTP.

Alcune caratteristiche importanti del protocollo Session Initiation Protocol sono:

- la possibilità di essere impiegato sia in contesti *client-server* che in contesti *peer-to-peer*;
- la facilità di estensibilità e di programmazione;
- la possibilità di avere server sia *stateless* che *stateful*;
- l'indipendenza dal protocollo di trasporto.

Per instaurare una sessione avviene un *three-way handshake* (concettualmente simile a quello che avviene con il protocollo TCP).

Gli utenti SIP sono risorse identificabili o localizzabili mediante URI o URL che contengono informazioni sul dominio, sul nome d'utente, sull'host, o sul numero col quale l'utente partecipa alla sessione. Gli indirizzi sono stile email. Ecco alcuni esempi:

- sip: bob@212.123.1.213
- sip: support@dominio.com
- sip: 22444032@dominio.com

1.2.2 I messaggi

Un messaggio SIP può essere o una richiesta (*Request*) o una risposta (*Response*). Una sequenza composta da una richiesta e da una o più risposte è detta *transazione*: una *transazione* è identificabile da un *transaction-ID*, che ne specifica la sorgente, la destinazione ed il numero di sequenza.

Entrambe le tipologie di messaggio (richiesta e risposta) sono costituite da una start-line, uno o più campi di intestazione (*header*) una linea vuota usata come separatore tra le intestazioni ed il corpo vero e proprio del messaggio (*body*), che è da considerarsi opzionale.

1. ***Request:*** I messaggi Request hanno come start-line una request-line costruita come segue:

   Request-Line =≪*Method*≫ ≪*Request-URI*≫ ≪*SIP-Version*≫

   Contenente il metodo di richiesta, cioè la semantica del messaggio, l'URI a cui è indirizzata la richiesta e la versione del protocollo utilizzata. Secondo le specifiche di SIP sono previsti i seguenti metodi di richiesta:

   - ***REGISTER:*** inviato da uno User Agent per registrare presso un Registrar Server il proprio punto di ancoraggio alla rete;
   - ***INVITE:*** serve ad invitare un utente a partecipare ad una sessione;
   - ***ACK:*** è un messaggio di riscontro; è inviato dallo User Agent chiamante, verso lo User Agent chiamato, per confermare la ricezione di una risposta definitiva ad un INVITE.
   - ***CANCEL:*** serve a terminare un dialogo quanto la sessione non ha avuto inizio;
   - ***BYE:*** utilizzato per terminare un dialogo SIP;
   - ***OPTIONS:*** è utilizzato per interrogare uno User Agent riguardo alle sue funzionalità, in questa maniera lo User Agent chiamante può decidere il tipo di comunicazione da instaurare;

- ***PRACK:***consente ad un client di riscontrare la ricezione delle risposte provvisorie, che possono essere così trasmesse in maniera affidabile, al pari di quelle definitive.
- ***REFER:*** lo User Agent che lo riceve trova nell'intestazione Referto una nuova SIP URI da contattare; dopo aver chiesto conferma alla persona che gestisce lo User Agent, la nuova SIP URI viene contattata (può essere una pagina web, od un altro User Agent), ed il mittente del REFER viene notificato dell'esito (con un messaggio NOTIFY);
- ***SUBSCRIBE:*** consente a chi lo invia di manifestare l'interesse a ricevere delle notifiche riguardanti l'evoluzione di alcune variabili di stato (tramite messaggi NOTIFY), indicate mediante l'intestazione Event, in cui si fa riferimento ad un event package; è utilizzato per E-Presence;
- ***NOTIFY:*** tiene uno User Agent al corrente dell'evoluzione di alcune variabili di stato; può essere inviato anche senza aver prima ricevuto un messaggio SUBSCRIBE;
- ***MESSAGE:*** permette l'invio di messaggi istantanei, ospitati nel body, e descritti da un'intestazione Content-Type.
- ***UPDATE:***consente ad un client di aggiornare i parametri di una sessione (quali tipologie di flussi multimediali e relative codifiche) senza modificare lo stato della sessione. Il metodo è usato dopo un messaggio di INVITE, ma prima che la sessione sia stata instaurata.
- ***INFO:***utilizzato per inviare ad uno User Agent, con cui si è già instaurata una sessione, delle informazioni relative ad eventi che avvengono dall'altro lato, come ad esempio, la pressione dei tasti del telefono;

2. ***Response:*** I messaggi Response hanno come start-line una status-line costruita come segue:

Status-Line = «SIP-Version» «Status-Code» «Reason-Phrase»

Costituita dalla versione del protocollo usata, un numero intero di tre cifre (status-code) ed una frase opzionale a commento della risposta. Lo status-code indica il tipo di risposta contenuta nel messaggio. In base al codice le tipologie di risposta si possono distinguere le seguenti sei classi:

- ***1xx - Provisional:*** risposte provvisorie necessarie ad interrompere i timer di ritrasmissione;
  - *100 Trying:* Questa risposta indica che la richiesta è stata ricevuta dal server il quale sta cercando il destinatario. Questa risposta ferma la ri-trasmissione di un INVITE.
  - *180 Ringing:* Lo User Agent che riceve l'INVITE cerca di avvertire lo user.
  - *181 Call Is Being Forwarded:* Un server può usare questo status code per indicare che la chiamata è stata inoltrata a diversi destinatari.
  - *182 Queued:* La chiamata nn è al momento disponibile, ma il server ha deciso di accodarla piuttosto che rifiutarla.
  - *183 Session Progress:* Può essere usato per comunicare al chiamante informazioni relative allo stato di una chiamata.
- ***2xx - Successfull:*** indica che l'operazione è avvenuta con successo;
  - *200 OK:* La richiesta è stata eseguita con successo. L'informazione di ritorno con la risposta dipende dal metodo usato nella richiesta.
- ***3xx - Redirection:*** richieste di redirezione della richiesta. In questo caso è il client (lo User Agent o il proxy che ha inoltrato la chiamata) che si deve occupare di richiamare l'indirizzo specificato;

- *300 Multiple Choises:* l'indirizzo fornito nella richiesta ha con- dotto all'individuazione di più terminali. L'utente dovrà quindi effettuare una scelta;
- *301 Moved Permanently:* l'utente cercato non è più raggiungibile all'indirizzo specificato nella richiesta;
- *302 Moved Temporarily:* l'utente cercato non è temporaneamente raggiungibile all'indirizzo specificato nella richiesta.
- *305 Use Proxy:* la risorsa a cui è destinata la richiesta deve essere acceduta attraverso un Proxy Server, il cui indirizzo è fornito nel campo Contact;
- *380 Alternative Service:* la chiamata non ha avuto succes- so, ma sono disponibili servizi alternativi descritti nel corpo del messaggio.

- ***4xx - Request Failure:*** la richiesta non può essere soddisfatta perché contiene qualche errore sintattico;
  - *400 Bad Request:* la richiesta non è stata compresa a causa di una sintassi errata;
  - *401 Unauthorized:* la richiesta necessita di un'autorizzazione da parte di un UAS o di un Registrar Server;
  - *402 Payment Required:* riservato per usi futuri;
  - *403 Forbidden:* la richiesta è stata rifiutata;
  - *404 Not Found:* l'utente cercato non può essere trovato;
  - *405 Method Not Allowed:* il metodo tentato non è consentito sull'indirizzo specificato;
  - *406 Not Acceptable:* La risorsa identificata dalla richiesta è in grado di generare solo risposte che hanno le caratteristiche descritte nel campo header dell' Accept inviato nella richiesta;
  - *407 Proxy Authentication Required:* la richiesta necessita di un'autenticazione presso un Proxy Server;
  - *408 Request Timeout:* Il server non ha potuto produrre una risposta entro una certo tempo, per esempio, se non è riuscito a determinare la posizione dell'utente entro il timeout.

- *410 Gone:* La risorsa richiesta non è più disponibile sul server e l'indirizzo di inoltro non è più noto;
- *413 Request Entity Too Large:* Il server si rifiuta di processare una richiesta perché il body della richiesta è più grande di quello che il server è in grado di elaborare;
- *414 Request-URI Too Long:* Il server si rifiuta di servire la richiesta perché la Request-URI è più lungo di quello che il server riesce ad interpretare.
- *415 Unsupported Media Type:* il corpo del messaggio si presenta in un formato non supportato;
- *416 Unsupported URI Scheme:* Il server non può elaborare la richiesta, perché lo schema della URI nella Request-URI è sconosciuto al server.
- *421 Bad Extension:* il server non supporta o non comprende l'estensione del protocollo richiesta dal metodo e specificata nel campo Require o Proxy-Require dell'intestazione;
- *420 Extension Required:* L' UAS ha bisogno di una particolare estensione per processare la richiesta, ma questa estensione non è elencato nel campo header Supported della richiesta;
- *423 Interval Too Brief:* Il server rifiuta la richiesta perché il tempo di aggiornamento della risorsa è troppo breve;
- *480 Temporarily Unvailable:* il chiamato è attualmente non disponibile;
- *481 Call/Transaction Does Not Exist:* la L'UAS ha ricevuto una richiesta che non corrisponde ad alcun dialogo esistente o transazione;
- *482 Loop Detected:* il server ha rilevato un loop;
- *483 Too Many Hops:* Il server ha ricevuto una richiesta che contiene il campo Max-Forwards dell'header con il valore zero;
- *484 Address Incomplete:* Il server ha ricevuto una richiesta con una Request-URI che era incompleta;

- *485 Ambiguous:* l'indirizzo indicato nella richiesta è ambiguo e non può quindi essere associato ad un utente in maniera univoca;
- *486 Busy Here:* il destinatario è stato contattato con successo, ma al momento è occupato;
- *487 Request Terminated:* La richiesta è stata terminata da una richiesta BYE o CANCEL;
- *488 Not Acceptable Here:* La risposta ha lo stesso significato di 606 (Not Acceptable), ma si applica solo alla specifica risorsa indirizzate dalla Request-URI
- *491Request Pending:* La richiesta è stata ricevuta da un UAS che aveva una richiesta pendente all'interno dello stesso dialogo;
- *493 Undecipherable:* La richiesta è stata ricevuta da un UAS che conteneva un corpo MIME crittografato per il quale il destinatario non possiede un'appropriata chiave per decriptare.

- ***5xx - Server Failure:***la richiesta appare valida, ma non può essere soddisfatta per un problema interno del server;
  - *500 Server Internal Error:* il server ha incontrato degli ostacoli imprevisti nel tentativo di soddisfare la richiesta. Se tale condizione è momentanea, il server può usare il campo di intestazione Retry-After per indicare al client quando ripetere la richiesta;
  - *501 Not Implemented:* il server non supporta la funzionalità necessaria per soddisfare la richiesta. Questa risposta viene generata quando il metodo usato nella richiesta non è stato riconosciuto, distinguendosi pertanto dal codice 405 (Methot Not Allowed);
  - *502 Bad Gateway:* Il server, mentre agiva come gateway o proxy, ha ricevuto una risposta non valida dal server downstream nel tentativo di soddisfare la richiesta;
  - *503 Service Unavailable:* il server al momento non è in grado di elaborare la richiesta a causa di un sovraccarico o di

operazioni di manutenzione in corso. Il client può rivolgere la richiesta ad un altro server e non dovrebbe tentare di ricontattare il primo server prima del termine stabilito dal campo Retry-After della risposta;

- *504 Server Time-out:* il server non ha ricevuto una risposta in tempo utile da un server esterno al quale aveva avuto accesso nel tentativo di ottenere quanto richiesto;
- *505 Version Not Supported:*Il server non supporta quella versione di protocollo SIP che è stato utilizzato nella richiesta;
- *513 Message Too Large:* Il server non è stato in grado di elaborare la richiesta in quanto la lunghezza del messaggio superato le sue capacità.

- ***6xx - Global Failure:*** la richiesta non può essere accettata da parte di nessun server.
  - *600 Busy Everywhere:* il destinatario è stato contattato su ogni indirizzo disponibile, ma non può rispondere perché occupato;
  - *603 Decline:* questa risposta viene inviata se il destinatario, con- tattato con successo, non vuole o non può accettare la richiesta di comunicazione;
  - *604 Does Not Exist Anywhere:* il server sa con certezza che l'utente indicato nel Request-URI non esiste all'interno della rete SIP;
  - *606 Not Acceptable:* lo User-Agent cercato è stato contattato con successo, ma alcuni aspetti della descrizione della sessione presenti nella richiesta, per esempio l'ampiezza di banda, non sono stati accettati.

Per quanto riguarda le intestazioni, sono rappresentate nel seguente formato:

«header-name»: «header-value» (, «header-value»)

dove «header-name» è il nome dell'intestazione, ed «header-value» il valore; (,«header-value») significa che ogni intestazione può avere più di un valore. Inoltre, per ogni valore possono essere specificati dei parametri aggiuntivi, ognuno con il proprio valore, e separati da punto e virgola, in modo da arricchire il loro potere espressivo, e permettere lo sviluppo di nuove estensioni; dunque il formato di «header-value» è il seguente:

«header-name»: «value (; «parameter-name»=«parameter-value»)

«value» è il valore dell'intestazione, «parameter-name» è il nome del parametro, mentre «parameter-value» ne è il valore. Le intestazioni più importanti sono:

- ***To:*** indica la URI del destinatario della richiesta;
- ***From:*** rappresenta la URI di chi invia la richiesta;
- ***Call-ID:*** è un identificatore semi-casuale, che resta uguale per tutti i messaggi di uno stesso dialogo, ovvero è univocamente associato ad un INVITE iniziale;
- ***CSeq:*** è costituita da un numero, seguito dal nome del metodo che ha dato inizio alla transazione;
- ***Via:*** è inserita da ogni elemento che invia una richiesta SIP, in cui indica il proprio indirizzo, porta, trasporto. Ogni elemento di transito che deve inoltrare la risposta rimuove l'intestazione da lui inserita, ed usa quella in cima per determinare a chi inviarla. In questo modo non occorre consultare il DNS, ed è sufficiente un proxy stateless;
- ***Max-Forwards:*** utile per limitare il numero di volte che un messaggio è inoltrato;
- ***Contact:*** contiene uno o più URI presso le quali il mittente di una richiesta desidera essere ricontattato;
- ***Allow:*** annuncia i metodi supportati da un'entità;

- ***Supported:*** elenca le estensioni supportate tra quelle elencate presso IANA;

- ***Record-Route:*** specifica la volontà di un proxy di essere mantenuto nel path dei futuri messaggi del dialogo.

1.2.3 Panoramica dell'operatività

Introdurremo ora le operazioni base del SIP utilizzando dei semplici esempi. Il primo esempio mostra le funzioni base del SIP: localizzazione di un end point, segnalazione di una richiesta di comunicazione, negoziazione dei parametri di sessione per stabilire la sessione, e la distruzione della sessione una volta stabilita.

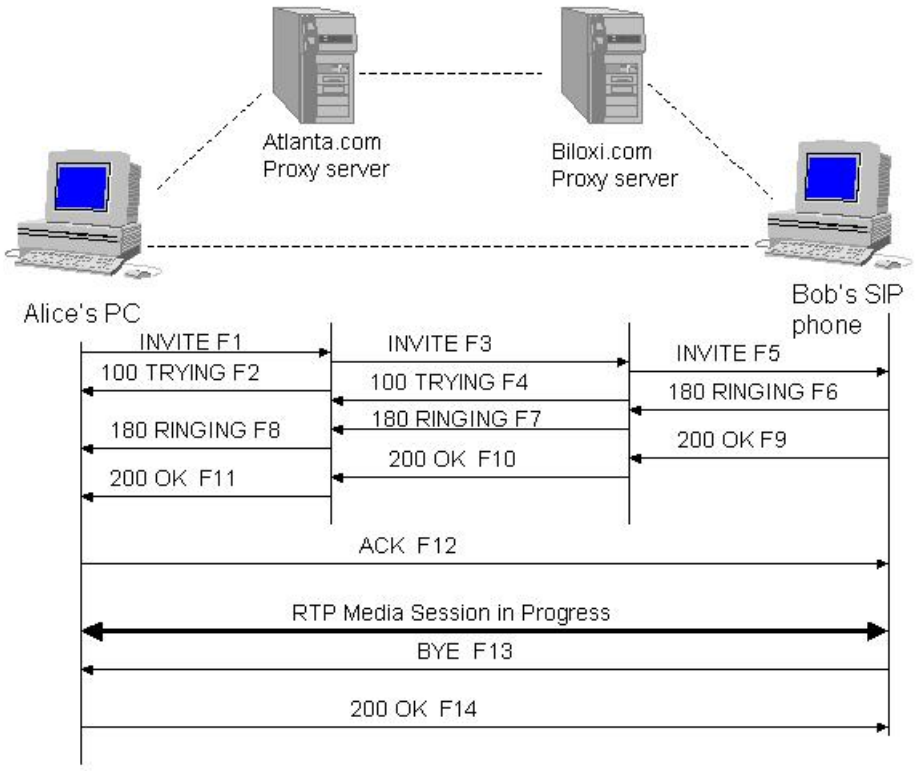

Figura 1.2: Esempio di impostazione di una sessione SIP con trapezio SIP

La figura 1.2 mostra un tipico esempio di un messaggio SIP scambiato tra due user, Alice e Bob. (Ciascun messaggio è etichettato con la lettera F e un numero di riferimento). In questo esempio Alice usa un'applicazione SIP sul suo PC (un softphone) per chiamare Bob sul suo SIP phone, via Internet. Mostra anche due SIP proxy server che facilitano l'instaurazione della sessione tra Alice e Bob.

Alice chiama Bob usando la sua SIP identity(URI) appunto chiamato SIP URI. Questa ha una forma simile all'indirizzo email, tipicamente contiene un username e un hostname. In questo caso è ***sip: bob@biloxi.com***, dove *biloxi.com* è il dominio del ISP di Bob. Alice ha come SIP URI ***sip: alice@atlanta.com***. Alice ha digitato l'URI di Bob o lo ha selezionato dalla sua rubrica. Il SIP fornisce anche un secure URI chiamato SIPS URI. Un esempio può essere sips: bob@biloxi.com Una chiamata fatta a un SIPS URI garantisce la sua sicurezza, è usato un trasporto criptato(chiamato TLS) per trasportare tutti i messaggi SIP da chiamante al chiamato. Perciò la richiesta è mandata in modo sicuro al chiamato, ma con meccanismi di sicurezza che dipendono dalle policy implementate sul dominio del chiamato.

Il SIP è basato su un modello di transazione richiesta/risposta HTTP-like. Ciascuna transazione consiste in una richiesta che invoca un metodo particolare, o funzione, su un server e riceve almeno una risposta. In questo esempio, la transazione comincia con il softphone di Alice che invia una richiesta INVITE al SIP URI di Bob. INVITE è un esempio di un metodo SIP che specifica l'azione che il richiedente (Alice) vuole che il server (Bob) compia. La richiesta INVITE contiene un numero di campi header. I campi Header sono chiamati attributi e forniscono informazioni addizionali riguardo il messaggio. In questo esempio l'INVITE contiene un identificativo univoco per la chiamata, l'indirizzo di destinazione, l'indirizzo di Alice, e le informazioni riguardanti il tipo di sessione che Alice vuole stabilire con Bob. L'INVITE message è rappresentato dalla Figura 1.3 (il messaggio F1 della Figura 1.2):

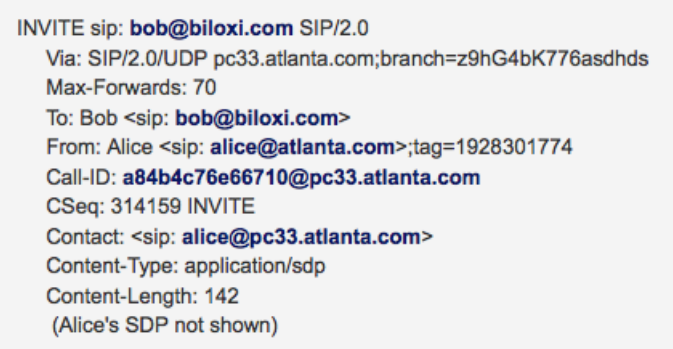

Figura 1.3: Esempio di messaggio SIP (INVITE)

La prima riga del messaggio contiene il metodo chiamato INVITE. Le righe che lo seguono sono una lista dei campi dell'header. Questo esempio ne contiene il minimo indispensabile.

1.2.4 Architettura di Rete

Le entità principale di una rete SIP sono:

- ***SIP User Agent:*** Rappresenta il punto di accesso ad un sistema SIP. E' un end-point e può fungere da client o da server; i due ruoli sono dinamici, nel senso che, nel corso di una sessione, un client può fungere da server e viceversa. Quando funge da client dà inizio alla transazione originando richieste. Quando funge da server accoglie le richieste e le soddisfa se possibile. Uno user agent registra le informazioni rilevanti del dialogo. Il dialogo ha inizio quando si risponde positivamente al messaggio di *INVITE* e termina con un messaggio *BYE*. Il compito di uno UA è quello di fornire un'interfaccia per la gestione dei servizi offerti dal sistema che nel caso di VoIP sono tipicamente conversazioni audio o video, messaggistica istantanea o conferenze. Di conseguenza, l'interfaccia deve contenere gli strumenti utili a gestire tali servizi, il cui numero e tipologia varia in base al terminale utilizzato.

- ***SIP Registrar Server:*** è un server dedicato o collocato in un proxy. Quando un utente è iscritto ad un dominio, invia un messaggio di regi-

strazione del suo attuale punto di collegamento alla rete ad un registrar server. Il suo compito è quindi quello di accettare o rifiutare l'ingresso di uno UA all'interno del network SIP durante la fase di autenticazione.

- ***SIP Proxy Server:*** è un server intermedio; può rispondere direttamente alle richieste, oppure «reinstradarle» ad un client, ad un server, o ad un ulteriore proxy. Richieste e risposte possono attraversare più Proxy Server prima di raggiungere la destinazione finale. Un proxy server analizza i parametri d'instradamento dei messaggi, e «nasconde» la reale posizione del destinatario del messaggio, essendo quest'ultimo indirizzabile con un nome convenzionale del dominio di appartenenza. I proxy possono essere di tipo *stateless* o *stateful*. Il primo tipo processa ogni richiesta o risposta SIP, ma non viene immagazzinata nessuna informazione riguardante il messaggio. Per ogni richiesta o risposta pervenuta si limita a inoltrare il messaggio all'entità SIP successiva, sfruttando unicamente informazioni di instradamento fornite nel messaggio stesso.; il secondo tipo tiene traccia delle richieste e delle risposte ricevute, ed utilizza queste informazioni per processare i messaggi futuri. Ciascuna richiesta SIP viene eseguita considerando le informazioni ottenute dalle richieste precedenti. Quando uno user agent invia sistematicamente le proprie richieste ad un proxy vicino (di default), allora tale proxy viene detto Outbound-Proxy. Viceversa, un Inbound-Proxy è un proxy che instrada le chiamate entranti in un dominio. Infine un Forking-Proxy può instradare una stessa richiesta in parallelo o in sequenza a più destinazioni.

- ***SIP Redirect Server:*** server che reinstrada le richieste SIP, consentendo al chiamante di contattare un insieme alternativo di URI. Il compito principale di un Redirect Server è quello di semplificare la localizzazione di un utente, fornendo informazioni opportune su dove si possa trovare il destinatario ricercato. Quando il Redirect Server riceve una richiesta, per determinare dove essa vada inviata consulta il Location Service, il quale fornisce il legame utente-indirizzo IP o il riferimento ad altri URI, in modo analogo al funzionamento del sistema DNS di Internet.

- ***Location Server:*** è un database contenente le informazioni riguardanti l'utente, come il profilo, l'indirizzo IP, e l'URL. Tipicamente risiede sulla stessa macchina dove è presente il Registrar Server.

1.3 Gli altri protocolli interessati

Vengono mostrati alcuni protocolli di rete comunemente usati nel contesto VoIP:

- **UDP:** [5] E' un protocollo di livello trasporto, che si affida al sottolivello IP per la trasmissione e la ricezione di pacchetti, detti datagram. E' un altro protocollo di di tipo connectionless, ovvero orientato alla trasmissione, e non garantisce una trasmissione affidabile dei pacchetti e nemmeno il loro ordinamento. A fronte di questi svantaggi, UDP è compatibile e comunemente usato anche per applicazioni che effettuano trasmissioni broadcast e multicast. La sua caratteristica più importante è che l'overhead dei pacchetti risulta minimo, rendendolo adatto per trasmissioni dati come quelle VoIP, in cui si inviano grandi quantità di pacchetti audio. Inoltre la perdita di alcuni pacchetti è tollerabile, seppur entro certi limiti al fine di garantire QoS.

- **SDP:** [6] E' protocollo utilizzato per descrivere i parametri di inizializzazione di streaming media (sessioni multimediali). SDP gestisce l'annuncio, l'invito e altri metodi di inizializzazione di una sessione multimediale. Non si occupa del trasporto dei dati, ma permette agli end-point di negoziare parametri di sessione, come il tipo di trasmissione, formati, codec e tutta una serie di proprieta? a cui spesso ci si riferisce col termine profilo di sessione.

- **RTP:** [7] E' un protocollo di livello applicativo utilizzato per trasmissioni real-time di dati come audio, video o dati di simulazione, attraverso servizi di rete unicast o multicast. Si noti che RTP non fornisce meccanismi per garantire la QoS (Quality of Service), ma fa affidamento ai protocolli di livello inferiore, per la gestione di tale problematica. Non vengono garantite n?e la consegna affidabile dei datagram n?e l'ordine di arrivo sequenziale, tuttavia il numero di sequenza presente nell'header rtp permette al ricevente di ricostruirne l'ordine corretto. Insieme a RTP viene usato il protocollo RTCP, per monitorare la QoS e trasmettere informazioni che riguardano i partecipanti a una sessione in corso.

- **SRTP:** [8] SRTP è un profilo per RTP, che garantisce le proprietà di confidenzialità, autenticazione dei messaggi e protezione da replay attack, ai datagram RTP e RTCP. Il protocollo mette a disposizione un framework per la cifratura e l'autenticazione dei messaggi, definendo un set di trasformazioni crittografiche. Se supportato da un appropriato sistema di distribuzione delle chiavi, rende sicure applicazioni RTP sia unicast che multicast. SRTP può? garantire un livello elevato di throughput e una ridotta espansione dei pacchetti, fornendo una protezione adeguata attraverso tipi di rete eterogenei.

- **ZRTP:** [9] E' un nuovo protocollo per telefonate sicure che consente di effettuare chiamate criptate sulla rete Internet. Utilizza un algoritmo pubblico ed evita la complessità dell'Infrastruttura a Chiave Pubblica. In effetti non impiega alcun sistema pubblico di distribuzione delle chiavi crittografiche. Utilizza l'algoritmo Diffie-Hellman di generazione e scambio delle chiavi e consente il rilevamento di attacchi di tipo Man in The Middle (MiTM), visualizzando una breve stringa di autenticazione che gli utenti leggono e confrontano al telefono. Lo scambio di chiavi Diffie-Hellman (Diffie-Hellman key exchange) è un protocollo crittografico che consente a due entità di stabilire una chiave condivisa e segreta utilizzando un canale di comunicazione insicuro (pubblico) senza la necessità che le due parti si siano scambiate informazioni o si siano incontrate in precedenza.

Capitolo 2

Applicazioni di riferimento

In questo capitolo verranno descritte le applicazioni che sono state considerate per effettuare l'analisi prestazionale. In particolare sono stati considerati dei progetti open-source. Andremo a vedere l'architettura e i dettagli implementati, così come il layout attraverso il quale l'utente interagisce con l'applicazione stessa.

2.1 Sipdroid

Sipdroid[10] è un client SIP open source fornito sotto licenza GPL. Disponibile solo per la piattaforma Android permette di effettuare chiamate su rete WIFI, 3G ed EDGE con un'ottima qualità audio e una bassa latenza. Allo stato attuale è giunto alla versione 2.4 ed è pienamente compatibile con la totalità dei dispositivi Android. SipDroid supporta inoltre la videochiamata permettendo all'utente in possesso di un dispositivo in grado di effettuare un'acquisizione video di inoltrare la stessa ad un utente remoto. Con l'uso della libreria jSTUN [11] viene garantito il suo funzionamento attraverso le varie strutture NAT preesistenti. Dal punto di vista multimediale il client Sipdroid mette a disposizione diversi tipi di codec audio garantendo all'utente un buon rapporto di qualità/compressione ed una corretta connessione con la maggior parte dei server SIP.

I codec audio supportati sono:

- *U-LAW e A-LAW:* sono derivati rispettivamente dai codec T1 (utilizzato in Nord America e in Giappone) e E1 (utilizzato nel resto del mondo) di tipo G.711 PCM appartenenti allo standard ITU. T1 ed E1 vengono utilizzati nella telefonia tradizionale e permettono di codificare segnali audio con teorica banda massima pari a 4000 Hz. Utilizzati con la tecnologia voip permettono di codificare segnali vocali con un'ottima qualità poichè non viene utilizzato alcun tipo di compressione rendendo il suono prodotto molto simile a quello di un regolare telefono. La mancanza di compressione (quindi l'assenza di un audio processing iniziale in fase di trasmissione) permette ai pacchetti di essere consegnati con una bassa latenza. D'altro canto, richiede un bit rate mediamente superiore agli 84 Kbps.

- *BV16:* codec ottimizzati per la trasmissione di segnali vocali sulla rete IP. La progettazione di questo codec ha avuto come obiettivo quello di rendere il ritardo dovuto alla codifica della trasmissione del segnale vocale il più basso possibile e di mantenere una buona qualità audio. Inoltre le richieste computazionali sono ridotte dalla bassa complessità dell'algoritmo di codifica e decodifica.

- *G722:* supportano un bit rate di 64, 56 e 48 kbps, possono essere integrati su un chip permettendo un ritardo complessivo dovuto all'elaborazione di all'incirca 3 ms, ritardo abbastanza piccolo da non causare problemi di echo. Questi codec offrono prestazioni accettabili per tassi di errore sui bit trasmessi fino a 10^3. Questo requisito assicura un lieve degrado delle prestazioni anche nelle condizioni peggiori di trasmissione che possono verificarsi all'interno di una rete.

- *GSM:* i codec di tipo GSM full rate operano ad un rate di 13 kbits/s e fanno uso di codec Regular Pulse Excited (RPE). Codificano un segnale vocale con una buona qualità sebbene sia inferiore a quella prodotta da codec che utilizzano un rate più alto come G728. Il principale vantaggio dell'utilizzo di codec con un basso rate è la relativa semplicità di implementazione che permette loro di lavorare in real time su sistemi con basse prestazioni.

- *SILK:* codec proprietari della società Skype, distribuiti con una licenza royalty free permettono di codificare una sorgente audio con un'ottima qualità, si adattano alle condizioni di traffico della rete in tempo reale scalando la larghezza di banda del segnale generato.

- *SPEEX:* codec che si adattano in modo ottimale per le applicazioni internet, possiedono caratteristiche non presenti in molti altri codec come la codifica stereo, l'integrazione di più frequenze di campionamento all'interno dello stesso stream di byte e dispone della modalità VBR (variable bit rate). D'altro canto richiedono risorse hardware superiori rispetto ad altri codec.

L'applicazione si presenta con la seguente interfaccia utente:

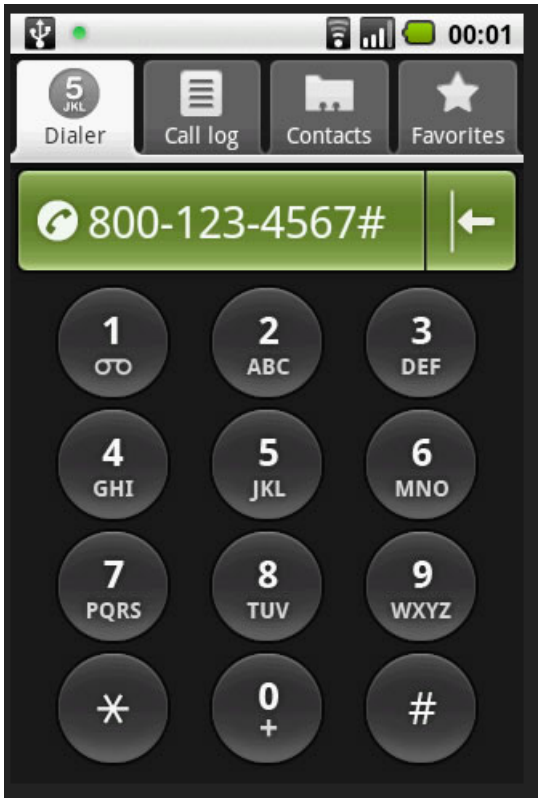

Figura 2.1: Sipdroid Main

Adesso illustreremo una "classica" procedura di *registration/logging* per ottenere un client Voip funzionante, considerando come esempio di riferimento l'applicazione *Sipdroid*. In linea di massima le altre applicazioni seguono lo stesso sequenza di step. Come vedremo in seguito, *CSipSimple* mette a disposizione un wizard che permette di configurare 10 account distinti, con la possibilità di sceglierne ognuno nel momento della chiamata Voip.

1. Registrarsi su un provider VOIP (Es. Ekiga.net) fino all'acquisizione di un numero telefonico geografico (è possibile sceglierne uno italiano), dopo aver ricevuto la notifica, le credenziali di accesso saranno:
   - **Login**
   - **Password**
   - **Sip Proxy Server**.
2. Registrarsi su *www.pbxes*. Di questa registrazione servirà Nome Utente e Password.
3. Loggarsi su *www.pbxes*, dopo **Setup**, successivamente:
4. Create un interno di tipo SIP **add Interno** e impostare come parametri:
   (a) **Numero interno:** 2000 (è un numero dell'interno che servirà nel caso si desideri condividere con più persone lo stesso account di *pbxes.com*.
   (b) **Display Name:** Giuseppe
   (c) **Password:** per comodità si può impostare la password precedente
   (d) I seguenti paramentri sono opzionali:
      - *Casella vocale e Directory:*
      - *Password:* Per la casella vocale;
      - *Email:* Indirizzo email sulla quale riceverla
      - *Allegati email:* SI
      - *RIproduci CID:* NO

- *Riproduci Data:* NO
- *Riprocudi Seguente:* NO
- *Elimina Messaggi Vocali* SI

(e) **Submit**

5. **Tronco**: Aggiungere un tronco SIP

   - **Nome Tronco:** Ekiga nel seguente esempio
   - **Lingua:**
   - **dtmfmode:** rfc2833
   - **audio bypass:** SI
   - **username:** Ekiga login del passo 1
   - **password:** Ekiga password del passo 1
   - **SIP Server:** Sip Proxy Server (il server voip associato al numero, voip.ekiga.it)
   - **Register:** SI
   - **Submit**

6. **Rotte in entrata:** Nel seguente esempio *2000*

7. **Rotte in uscita:** Nel seguente esempio *SIP/ekiga*, dunque **ADD**

8. Impostiamo adesso la nostra applicazione **SIPDROID**

   - **Menu**
   - **Username:** *pbxes.org -2000*
   - **Password:** del passo 2;
   - *Selezionare la connettività che si desidera utilizzare: 3G o WiFi*

Una volta terminata tale procedura l'applicazione è impostata correttamente per il funzionamento.

2.1.1 MjSip

SipDroid utilizza lo stack SIP creato da MjSip[12]. MjSip include tutte le classi e i metodi per la creazione di applicazioni SIP-based.

E' formata da diversi packeges che includono:

- Oggetti standard SIP come messaggi SIP, transazioni, etc.
- Varie estensioni SIP definite all'interno del IETF.
- Call API di controllo.
- Un'implementazione di riferimento di alcuni sistemi SIP come server ed user agents.

Le principali caratteristiche di MjSip sono:

- È basato su Java quindi è cross-platform.
- Leggerezza del codice, e possibilità di utilizzo su piattaforme come la nostra, che devono fronteggiare una capacita' di calcolo ed autonomia energetica limitata;
- Non è solo un API, ma include l'implementazione completa dello stack SIP.
- È molto leggero e può essere utilizzato contemporaneamente sia per l'implementazione del server sia per quella del client.
- Semplicità d'uso e di estensione o modifica delle funzionalità;
- Inclusione di API interfacciate con l'utente in maniera identica a JAIN(Java APIs for Integrated Networks) API;

Per quanto stabilito nell'architettura SIP definita nell' RFC 3261, l'architettura di MjSip è strutturata in 3 livelli base: *Transport, Transaction, Dialog.* Alla cime di questi 3 livelli MjSip fornisce un *Call Controll* e il livello applicativo con le APIs corrispondenti:

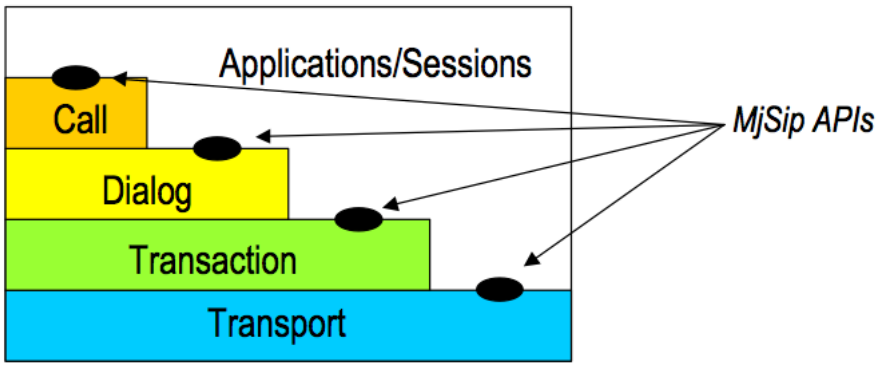

Figura 2.2: Architettura MjSip

- Il layer più basso è il livello *Transport* che fornisce il trasporto dei messaggi SIP. Il **SipProvider** è l'oggetto MjSip che fornisce il livello di trasporto a tutti i livelli superiori. Esso è responsabile di spedire e ricevere messaggi SIP attraverso differenti protocolli di trasporto di basso livello e di demodulare i messaggi in ingresso attraverso appropriate entità del livello superiore. Ogni elemento SIP deve utilizzare le API del SipProvider se vuole avere accesso al servizio di trasporto MjSip.

- Il secondo layer è il *Transaction*; esso è una componente essenziale poiché nel SIP una transazione è una richiesta spedita da un client (transaction client) verso un transaction server e le risposte che quest'ultimo invia al transaction client.. Il livello di transazione, quindi, gestisce la ritrasmissione dei livelli superiori, il matching delle richieste e risposte e il timeout. Ovviamente il livello di transazione deve inviare e ricevere messaggi per mezzo del livello di trasporto. E' possibile identificare due tipi di transazioni:

  1. *two-way transaction:* questo tipo di transazione non richiede da parte del client di inviare il riscontro del messaggio di fine transazione inviato da server.

2. *three-way transaction:* questo tipo di transazione prevede invece il riscontro della ricezione del messaggio inviato dal server transaction al client transaction.

- Il terzo layer è il *Dialog* il quale lega differenti transazioni della stessa "sessione". Il dialogo è una relazione SIP peer-to-peer tra due user agent che persiste per un certo periodo di tempo. Il livello di dialogo facilita la successione dei messaggi e le routin di richiesta tra i vari user agent.

- Il layer più alto è costituito dal *Call Control* che implementa una suite completa di chiamate SIP. E' implementato dall'API ***Call*** che offre una semplice interfaccia che si occupata delle chiamate in entrata e in uscita. Una Call può essere formata da più di un dialogo e mette a disposizione dello sviluppatore una serie di API per il controllo di una chiamata SIP.

Sopra questi 4 livelli ci sono le sessioni SIP che uniscono due o più entità applicative (client) su sistemi diversi.

MjSip offre APIs per l'accesso a tutti i livelli SIP precedenti (dal SipProvider fino al Call) e una implementazione di riferimento per vari sistemi a livello applicativo.

2.2 Linphone

Linphone [13] è un'applicazione specifica per la telefonia via internet, Voip, basata sul protocollo SIP. L'applicazione permette di effettuare oltre alle chiamate anche videochiamate includendo al suo interno diversi codec video, oltre che audio, supporta i protocolli IPv4 e IPv6 per il trasferimento dei pacchetti dati, offre inoltre uno strumento per la gestione della rubrica dei contatti. Con Linphone è possibile comunicare liberamente con le persone attraverso internet, tramite voce, ma anche tramite video e messaggi (chat). Linphone usa il protocollo SIP, il protocollo standard per la telefonia tramite internet. E' possibile quindi usare Linphone con qualsiasi operatore che fornisca un provider VOIP SIP. Linphone è free ed inoltre è open-sorce, e quindi è possibile scaricare liberamente il software ma anche 'e possibile adattarlo alle nostre esigenze, dato che viene rilasciato come open-sorce. Linphone è disponibile per i PC, sia Linux che Windows, per MacOSX e per i telefoni mobili, come Android e iPhone.

Le caratteristiche del core di Linphone sono:

- General:
  - Portabilità: E' eseguibile su varie piattaforme. Linux/x86 e Linux/x86-64; Windows XP, Vista e 7; Linux/ARM - Blakfin; MacOSX x86; Google Android; Iphone OS; Blackberry OS; WebOS
  - Mobile phones interfaces: Applicazioni Android(con supporto video) - iPhone/iPad(con supporto video) - Blackberry (senza supporto video)
  - Fornisce una Rubrica
  - Riconosce SIP ENUMS (Telephone number mapping)
  - Fornisce i messaggi istantanei e il rilevamento della presenza
  - Supporta chiamate multiple simultanee con la gestione delle chiamate
  - Fornisce DTMF (toni di telefonia) usando SIP INFO o RFC2833
- Signaling:

  - Supporta il NAT. Riconosce gli indirizzi NAT per i messaggi SIP, usa STUN per gli indirizzi RTP
  - E‘ un User Agent SIP compliant dell’RFC3261
  - SIP/UDP, SIP/TCP, SIP/TLS
  - Supporta IPv6
  - Supporta Digest authentication
  - Supporta Multiple SIP proxy: registrar, proxies, outbound proxies

- Media:

  - L’audio può essere impostato con i seguenti codec: speex, G711 (ulaw, alaw), GSM. Grazie a dei plugin ulteriori, supporta anche AMR e iLBC
  - Il video può essere impostato con i seguenti codec: H263, H262-1998, MPEG4, theora e H262 (grazie ad un plugin basato su x264), con risoluzione da QCIF(176x144) a SVGA(800x600)
  - Audio conferencing
  - Supports SRTP and ZRTP (per criptare voce e video)
  - Supporta qualsiasi webcam con driver V4L o V4L2 sotto Linux e driver Directshow sotto Windows
  - Supporta la cancellazione dell’eco
  - Fornisce la gestione della banda in modo efficiente: i limiti di banda sono segnalati usando SDP, risultando la sessione dell’audio e del video stabiliti con il bitrate della capacità della rete dell’utente
  - Fornisce algoritmi Adaptive audio & video bitrate per l’adattamento al bandwidth di rete disponibile
  - Si possono usare altri plugin: si possono aggiungere nuovi codec, o nuove funzionalità al core come, ad esempio, la ricerca di indirizzi sip in modo remoto.

2.2.1 Architettura

Linphone è costituita da una separazione interna tra la user interfaces e la core engine, che permette di creare vari tipi di user interface nel livello di astrazione piu alto, cioè di integrare qualsiasi applicazione grafica.

I frontend dell'interfaccia utente sono:

- Gtk+/glade interface
- La console interface. Linphone mette a disposizione degli utenti due tipi di interfacce grafiche o più precisamente due tool tramite console. Grazie a queste interfacce a linea di comando è possibile usare Linphone anche da riga di comando rendendo questo software più portabile. Il primo tool è ***"linphonec"*** che è un'interfaccia grafica a linea di comando che può essere usata per gestire completamente Linphone attraverso dei comandi dal prompt dell'utente. L'altro tool da linea di comando per il controllo da remoto del demone di linphonec è ***"Linphonecsh"***.
- iPhone application integrato dell' objective C
- Android application eseguito in Java

La libreria(core engine) che implementa tutte le funzionalità di linphone è **Liblinphone** [14]. Si tratta di un potente SIP VoIP video SDK che può essere usato per aggiungere funzionalità di chiamata audio/video ad una applicazione. Fornisce una API ad alto livello per avviare, ricevere, terminare le chiamate. Liblinphone e tutte le sue dipendenze sono scritte in C e si basa sulle seguenti componenti software:

- *mediastreamer2*: E' un potente e leggero motore di streaming specializzato per applicazioni telefoniche voce/video. E' la libreria responsabile di tutti la ricezione e l'invio di flussi multimediali in Linphone, compresa la cattura voce/video, codifica e decodifica e il rendering. Si occupa della codifica/decodifica di diversi formati audio, lettura/scrittura da/su file wav,cancellazione dell'echo, controllo del volume e del gain.

- *oRTP*: Una libreria RTP, scritta in C che implementa l'RFC3550 (RTP). Include uno scheduler (opzionale) di pacchetti per l'invio e la ricezione di pacchetti on time, in base al loro timestamp. Supporta il multiplexing I/O in modo che un singolo thread possa gestire diverse sessioni RTP. Inoltre supporta parte del protocollo RFC2833.

- ***eXosip2***: Una SIP user agent library rilasciata con licenza GPL. Il suo scopo è quello di implementare un semplice strato di API per stabilire e controllare sessioni SIP e le comuni estensioni (gestione chiamata, messaggistica). eXosip2 fornisce il supporto per:

  - registrazioni **REGISTER**
  - inizializzazione chiamata e modifica chiamata **INVITE, re-INVITE**
  - altri metodi per le chiamate **INFO, OPTION, UPDATE**
  - trasferimento chiamate **REFER**
  - affidabilità nella risposta provvisoria **PRACK**
  - sip event package **SUBSCRIBE/NOTIFY**
  - pubblicazione dello stato degli eventi **PUBLISH**
  - messaggi istantanei **MESSAGE**

eXosip2 si basa sulla libreria *libosip2*. Non si tratta di uno stack SIP completo, ma di una libreria che si occupa di formire un limitato numero di features comuni a qualsiasi SIP Agents. Può essere usato per implementare SIP End-Point e Proxy. Le caratteristiche che implementa possono essere così sintetizzate:

- SIP parser: la capacità di parsare e formattare richieste e risposte SIP

- SIP transaction state machine:

  - Invite Client Transaction *(ICT)*
  - Non Invite Client Transaction *(NICT)*
  - Invite Server Transaction *(IST)*
  - Non Invite Server Transaction *(NIST)*

- Metodi astratti per la gestione di thread, semafori e mutua esclusione.

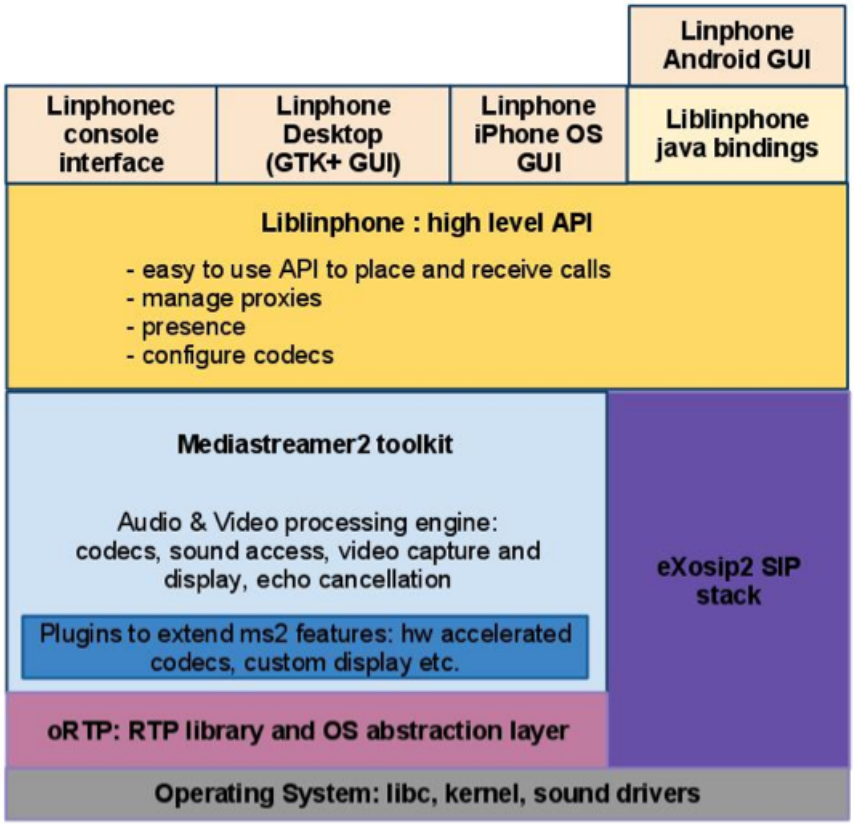

Figura 2.3: Architettura Linphone

L'applicazione si presenta con il seguente layout utente:

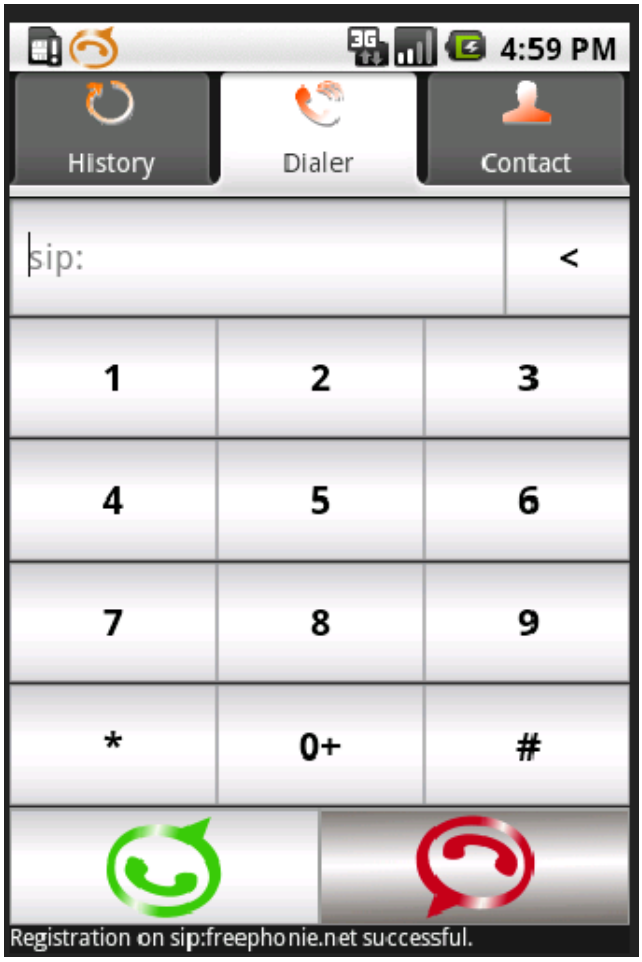

Figura 2.4: Linphone Main

2.3 CSipSimple

CSipSimple [15] è un free SIP client esclusivo per Android OS. E' un progetto open source rilasciato sotto licenza GNU General Public License. La prima stable release è stata rilasciata nel Novembre 2011 ed è scritto in Java, C/C++. Il progetto è complessivamente composto di 3 parti:

- *CSipSimple* che è l'applicazione Java per Android;
- *pjsip-android/jni* che rappresenta il porting di PJSIP per la piattaforma Android;
- *pjsip* che viene estratto a tempo di compilazione;

Le caratteristiche principali di questo software sono:

- Supporto multi-codec: *Speex* (narrow band, wideband), *G.711* (u-law, a-law), *GSM*, *iLBC*, *G.722*, *AMR* (narrow band), *iSAC*. Un plugin che permette l'utilizzo di *SILK* (narrow band, wideband, ultra wideband), *CODEC2*, *G.726*, *G.722.1*.
- Supporto multi-account : possono essere attivati fino a 10 account contemporaneamente.
- Può usare driver audio nativi.
- *STUN* e *ICE NAT* traversal.
- Integrazione con il sistema operativo Android con filtri e regole di riscrittura
- Meccanismi di encryption come *SRTP* (Secure Real Time Protocol), SIP su *TLS (Transport Layer Security)* e *ZRTP* (protocollo per la negoziazione di chiavi di cifratura tra due end-point in Voip).
- SIP-SIMPLE messaging
- Disponibilità di API.

L'applicazione si presenta con il seguente layout utente:

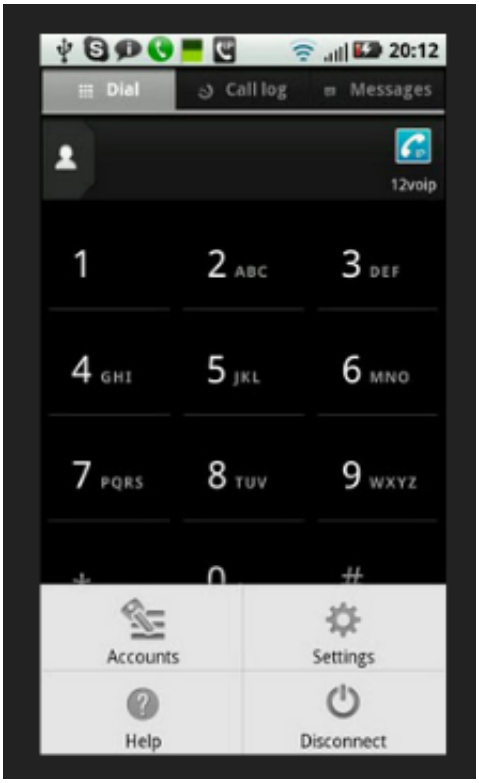

Figura 2.5: CSipSimple Main

La libreria sulla quale poggia l'applicazione *CSipSimple* è PJSIP, che adesso vedremo nel dettaglio.

2.3.1 PJSIP

PJSIP [16] è un gruppo di librerie scritte in C per la creazione di applicazioni scalabili, dunque la gestione delle comunicazioni multimediali attraverso il protocollo SIP . Sono pensate per essere estremamente portabili e performanti, adatte per questo ad essere eseguite su architetture critiche, come ad esempio i dispositivi mobili, dove il consumo energetico e le risorse hardware sono ridotte. Di seguito verrà descritto lo stack delle principali tecnologie che ne fanno parte e le particolarità che hanno fatto di PJSIP un backend ideale per le applicazioni VoIP su dispositivi mobili.

- **PJLIB** PJLIB è la libreria su cui tutte le altre si appoggiano, si occupa di garantire funzionalità di base quali l'astrazione rispetto al sistema operativo sottostante. Contiene una replica pressochè completa della libreria LIBC ed una serie di feature aggiuntive come la gestione dei socket, delle funzionalità di logging, dei threads, della mutua esclusione, dei semafori, delle critical section, funzioni di timing, nonchè un costrutto per la gestione delle eccezioni, e la definizione di varie strutture dati di base, come ad esempio liste, stringhe e tabelle di hash.

- **PJMEDIA** PJMEDIA è una libreria complementare e si occupa del trasferimento e della gestione dei dati multimediali. Tra le sue caratteristiche principali vi è la gestione degli stack RTP/RTCP, buffer adattivo per ridurre i problemi legati al jitter, rimozione degli echi, silence detector, generazione dei toni, supporto ad un'ampia varietà di codec tra cui speex/iLB-C/GSM/G.711. Supporta l'encoding e il decoding di frequenze di 16/32 Khz e la conversione di qualunque campione verso queste frequenze, essa inoltre riesce a tollerare bene i problemi classici legati alle reti basate su IP che sono il jitter e la perdita di pacchetti.

- **PJSUA** PJSUA è una libreria di alto livello che fa da wrapper verso le altre librerie e consente di scrivere più agevolmente le applicazioni.

- **PJLIBUTIL** PJLIBUTIL è invece una libreria ausialiaria che fornisce supporto a PJMEDIA e PJSIP. Alcune sue funzioni sono: small footprint di parsing xml, caching asincrono di risolver DNS, funzioni di hashing e di crittografia.

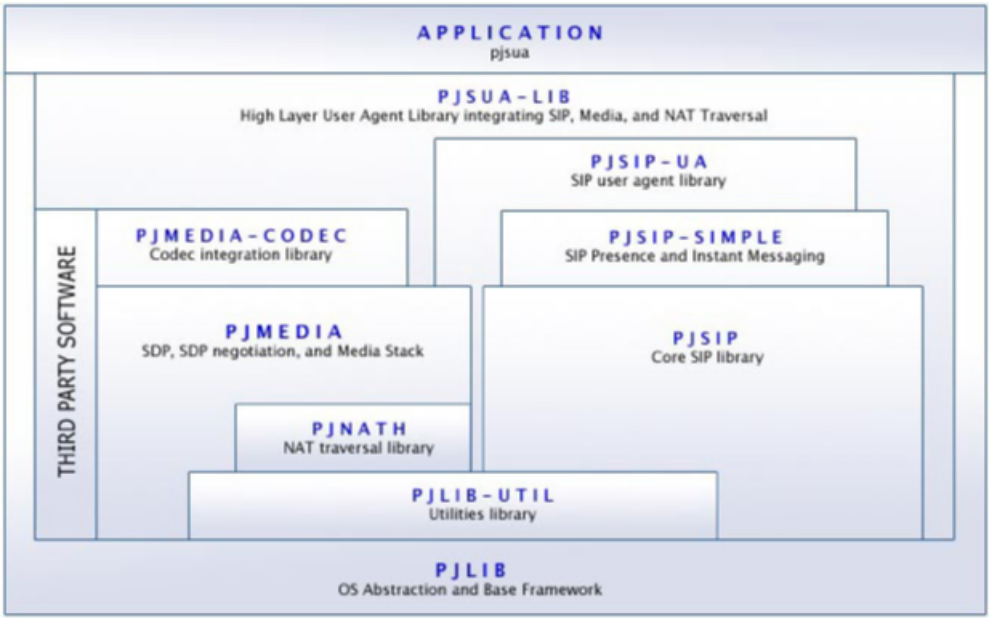

Figura 2.6: Architettura PJSIP

I vantaggi offerti da questa libreria possono essere così sintetizzate:

- *Portabilità:* può essere eseguita in diversi sistemi operativi che siano a 16, 32 o 64 bit, little o big endian, singolo o multi-processore, con o senza capacità di calcolo in virgola mobile e supporto multi-threading del S.O.

- *Performance:* può gestire più transazioni SIP contemporaneamente mantenendo minimo il consumo delle risorse, come la CPU;

- *Astrazione dal sistema opertivo:* La libreria PJLIB astrae alcune delle caratteristiche proprie dei vari sistemi operativi quali threads, sezioni critiche, semafori, eventi, manipolazione di date, ecc.. che di solito rendono complesso lo sviluppo di applicazioni multipiattaforma.

- *Gestione della memoria ottimale:* Viene evitata l'allocazione dinamica della memoria, pensi viene utilizzato un pool preallocato. Una gestione di questo tipo può essere, in certi sistemi, fino a 30 volte superiori rispetto all'utilizzo classico di malloc e free. :
  - alloc() ha costo computazionale O(1);
  - si assume che la sincronizzazione venga usata ad un livello di astrazione più alto;
  - la funzione free() (per la pulizia della memoria) nn viene utiizzata;
- *Dimensioni ridotte:* PJSIP è caratterizzata da avere ridotte dimensioni in modo da essere particolarmente adatta ai dispositivi con risorse limitate, come gli smartphone. E' possibile infatti aggiungere o rimuovere funzionalità non necessarie in determinati ambiti o architetture, arrivando ad avere funzionalità VoIP minimali in appena 100Kb di spazio.
- *I/O di rete a basso livello:* PJLIB ha un set proprio di API per gestire le comunicazioni attraverso rete. Queste ultime rendono possibile ulteriori astrazioni quali:
  - Socket, l'astrazione della socket permette di aggiungere nuove funzionalità di rete mantenendo la compatibilità sui sistemi supportati;
  - Funzioni di risoluzione dell'indirizzo;
  - Strutture dati comuni;
  - Gestione delle eccezioni;
  - Sistema di Logging;
- *Altre features:* Un vasto insieme di altre funzionalità quali messaggistica istantanea, trasferimento di chiamata, ecc..

Capitolo 3

Test e Analisi

Per rendere chiari gli obiettivi della tesi è importante definire quali sono stati gli step seguiti nella terminazione della stessa e capire bene il contesto in cui si è andati ad operare. Molto sinteticamente si possono riassumere i passi che hanno portarto a riscontrare diverse problematiche tutto aperte nel campo delle mobile network.

- **Open source:** Il dominio che è stato preso in considerazione e che avrebbero permesso lo sviluppo successivo. Come vedremo successivamente, sono diverse le applicazione costruite per android che mettono a dispozione tutta quella suite di servizi che sono stati analizzati, ma sono poche quelle che permettono, anzi ne favoriscono il libero studio e l'apporto di modifiche da parte di altri sviluppatori.

- **Problema dei Firewall e NAT**: Alcuni dei problemi legati al Voip riguarda proprio il particolare scenario di rete dove si va ad operare. La presenza di Firewall e NAT causa alcune complicazioni che in seguito andremo ad affrontare.

  - Configurabilità con il server proxy ABPS
  - Uso di librerie "light" e performanti

- **Problema di Usabilità:** l'efficacia, l'efficienza con la quale gli utenti riescono ad interagire con l'applicazione(interfaccia grafica, semplicità del menù e del settaggio)

Partendo da queste considerazioni andremo adesso a vedere come si è proceduto e i vari step che si sono susseguiti. Il punto di partenza del lavoro è stato:

- *Trovare, se esiste, un'applicazione Voip Android, open source, in grado di garantire un "buon funzionamento".*

Per "buon funzionamento" si è scelto di misurare una serie di parametri ritenuti necessari per garantire delle chiamate Voip di buona qualità , sia audio che video. Le caratteristiche delle applicazioni prese come riferimento, introdotte nel capitolo precedente, vengono riassunte nella seguente tabella:

Program	Operating System	Licence	Protocols	Encryption	Other Capabilities	Latest realease
SIPDROID	Android	GNU GPLv3	SIP	Unknown	Uses WIFI, 3G or EDGE	2.7 (May 2012)
LINPHONE	Android	GNU GPLv2	SIP	Unknown	Eaudio codecs speex, G.711(ulaw, alaw), GSM.	01.02.02 (December 2011)
CsipSimple	Android v1.6+	GNU GPLv3	SIP	SRTP / ZRTP / TLS	VoIP over Wi-Fi or mobile data; Audio codecs G.711, Speex, GSM, iLBC, G.722, AMR, iSAC, SILK, CODEC2, G.726, G.722.1; Conference; Android integration with filtering/rewriting rules	0.03-01 (November 2011)

Tabella 3.1: Applicazioni esaminate

Sono state esaminate 3 applicazioni Voip open source, sviluppati per diverse piattaforme mobili, ma esistono anche delle versioni di softphone per i vari sistemi operativi desktop.

Vengono sintetizzate di seguito le caratteristiche rilevanti degli strumenti utilizzati per effettuare i *"Test"*:

- **Samsung Galaxy S**:
  - *CPU:* 1 GHz (ARM Cortex A8)
  - *RAM:* 512 MB
  - *Sistema Operativo:* Android 2.3.6 Gingerbread
  - *Connettività:* Wi-Fi, 3G, GSM/GPRS/EDGE
- **HTC Wildfire**
  - *CPU:* 528 MHz
  - *ROM:* 512 MB
  - *RAM:* 384 MB
  - *Sistema Operativo:* Android 2.3.6 Gingerbread
  - *Connettività:* Wi-Fi, 3G, GSM/GPRS/EDGE

Si tratta di due smartphone con caratteristiche hardware molto diverse tra loro, ma non software, in quanto la funzionalità VOIP è stata implementata nativamente sulla versione 2.3 Gingerbread. Le diverse caratteristiche prestazionali dei due dispositivi ha condizionato notevolmente la bontà dei test ma ha reso realistico la validità dei test stessi: macchine diverse e con prestazioni diverse rappresentano meglio lo scenario reale. Su entrambi sono stati istallati le stesse release delle diverse applicazioni, in particolare le ultime elencate nella precedente tabella.

Per rendere più verosimili le valutazioni si sono utilizzati diversi account provider Voip e come vedremo, i test saranno caratterizzati anche da questi dettagli. Ognuna delle applicazioni mette a disposizione un menù in cui è possibile scegliere tra diversi domini, suggerendone alcuni tra i più utlizzati (Es. Ekiga, Eutelia, ecc.), ma dando la possibilità di impostare altri presenti in rete.

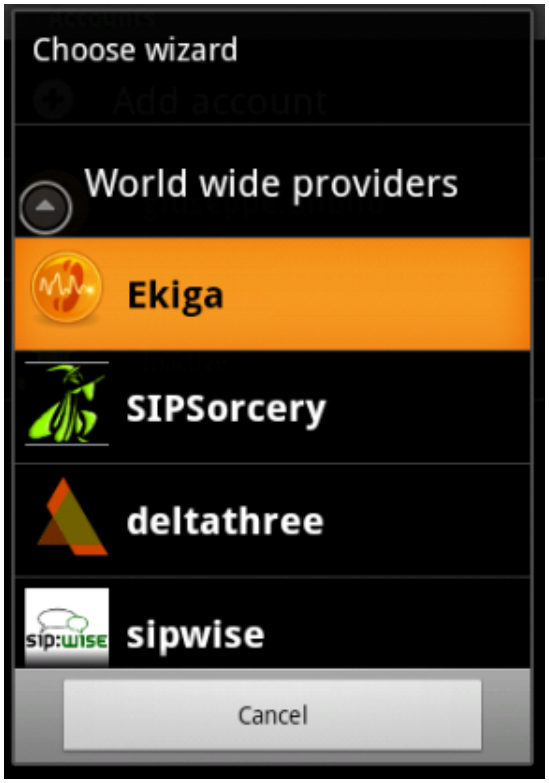

Figura 3.1: Wizard Account CSipSimple

In particolare *CSipSimple* mette a disposizione la possibilità di impostare ben 10 account distinti, dando la possibilità di selezione dell'account nel momento della chiamata. Nella figura precedente è possibile vedere il wizard per la gestione degli account

3.1 Livelli di valutazione

Per descrivere la qualità del software esaminato si è deciso di creare una valutazione a livelli, in modo che si ha una certa coerenza dei criteri di giudizio e risulta facile comprendere la formulazione dei dati nella varie tabelle, dando una visione dettagliata dei risultati ottenuti.

* ***Livello 0(L.0): Scarsa*** La videochiamata lamenta di una pessima qualità sia audio che video. Vi è una pessima sincronizzazione audio/video e i ritardi sono considerevoli.

* ***Livello 1(L.1): Discreta*** La videochiamata è accettabile relativamente all'audio, ma pessima considerando la sessione video. I ritardi sono considerevoli.

* ***Livello 2(L.2): Buona*** La videochiamata garantisce un' ottima qualità audio ma difetta in qualche ritardo video. I ritardi tuttavia sono accettabili e nn influiscono sulla terminazione della chiamata.

* ***Livello 3(L3): Ottima*** La videochiamata garantisce un' ottima qualità audio/video. I ritardi sono trascurabili e c'è perfetta sincronizzazione tra l'audio e il video.

Nella prima fase di Test sono state effettuate sessioni prendendo in considerazione solo parametri riguardanti la connettività e che inizialmente abbiamo definito essere essenziali. Possiamo riassumerli in questo ipotetico scenario di network:

- **Terminale 1:** Uno dei due dispositivi è connesso alla rete mediante 3G, lo standard di telefonia mobile.

- **Terminale 2:** L'altro dispositivo è connesso ad una WLAN che non configura alcun firewall o NAT.

I risultati delle sessioni sono riassunte nella seguente tabella.

	SIPDROID	LINPHONE	CSIPSIMPLE
3G	*L.3*	*L.3*	*L.3*
WIFI	*L.3*	*L.3*	*L.3*

Tabella 3.2: Comunicazione minimale 3G e Wi-Fi

Si può notare che non sembrano esserci particolari problemi in questo tipo di scenario ed è possibile definirlo "minimale".

Come accennato precedentemente i software che sono stati valutati, hanno avuto diversi comportamenti derivanti dall'utilizzo di account Voip appartenenti a diversi service prodiver. In questa tabella sono riportati i risultati del test in una connessione di tipo Wi-Fi.

Wi-Fi without NAT	SIPDROID	LINPHONE	CSIPSIMPLE
Same Provider	*L.3*	*L.3*	*L.3*
Different Provider	*L.3*	*L.2*	*L.3*

Tabella 3.3: Utilizzo di account differenti in una connessione Wi-Fi

In quest'altra tabella invece, con gli stessi criteri di giudizio, vengono valutate in un scenario di connessione 3G.

3G	SIPDROID	LINPHONE	CSIPSIMPLE
Same Provider	*L.3*	*L.3*	*L.3*
Different Provider	*L.3*	*L.3*	*L.3*

Tabella 3.4: Utilizzo di account differenti in una connessione 3G

Si osserva come, considerando lo stesso scenario, in particolare in un contesto Wi-Fi l'applicazione non garantisce stabilità. Un motivo di tale comportamento potrebbe essere attribuito a vari problemi che caratterizzano questa tipologia di rete, quali:

- **Grado di congestione della rete:** I router potrebbero essere sovraccaricati da troppo traffico. Questa situazione potrebbe verificarsi in corrispondenza del provider di servizi Internet o della rete LAN (Local Area Network), considerando che stiamo considerando streaming audio e video.

- **Interferenze:** numerosi dispositivi wireless sono connessi ai punti di accesso. Può inoltre verificarsi in presenza di un'interferenza con altri dispositivi, ad esempio altri cellulari.

- **Configurazione hardware.**

Ne risulta che in situazioni critiche è possibile che si verifichino uno o più degli effetti seguenti:

- Audio metallico: La voce dell'interlocutore sembra innaturale e con un tono più alto del previsto.

- Perdita di parole: Parti della conversazione vengono perse.

- Video discontinuo: L'immagine video non ha un movimento uniforme.

- Ritardo prolungato: Intercorre un lungo ritardo tra il momento in cui un'utente parla e quello in cui l'altra sente l'audio.

3.2 Firewall e NAT

Il questa fase di valutazione sono stati considerati gli scenari di network in cui vi è la presenza di firewall. A tal fine è necessario analizzare il contesto e prendere atto delle problematica derivanti.

- *Cosa è un **Firewall?**:* [17] un firewall è un componente passivo di una rete informatica che può anche svolgere funzioni di collegamento tra due o più tronconi di rete e che garantisce dunque una protezione in termini di sicurezza informatica della rete stessa. Usualmente la rete viene divisa in due sotto-reti: una, detta esterna, comprende l'intera rete Internet mentre l'altra interna, detta LAN (Local Area

Network), comprende una sezione più o meno grande di un insieme di terminali (fissi o mobili) locali. Lo scopo principale dei firewall è quello di proteggere una rete interna dagli accessi di utenti non autorizzati. Normalmente il traffico in entrata da host esterni viene consentito solo se la sessione è stata avviata dalla rete interna, pertanto le chiamate in entrata, provenienti da una rete esterna (WLAN), verranno filtrate e l'applicazione non riuscirà a stabilire una connessione con gli utenti finali.
Il firewall agisce sui pacchetti in transito da e per la rete interna, grazie alla sua capacità di aprire il pacchetto IP per leggere le informazioni presenti nel suo header, e in alcuni casi anche di effettuare verifiche sul contenuto del pacchetto stesso. Una tipica soluzione al problema è configurare il firewall in modo tale da usare solo specifiche porte per le connessioni VoIP, ma questo approccio presenta un forte rischio per la sicurezza, poiché un malintenzionato che è a conoscenza delle porte o che può arrivare a conoscerle, ne potrebbe approfittare. In più questa è una soluzione che richiede una configurazione da parte di tecnici o amministratori, visto che l'utente medio non hanno le conoscenze necessarie per farlo, oppure, peggio ancora, possono essere i provider stessi a non permettere questa configurazione.

- *Cosa è **NAT?**:* [18] Il network address translation , ovvero traduzione degli indirizzi di rete è una tecnica che consiste nel modificare gli indirizzi IP dei pacchetti in transito su un sistema che agisce da router all'interno di una comunicazione tra due o più host.
  Dunque i NAT effettuano la traduzione di indirizzi e/o porte fra spazi di indirizzamento differenti: i due spazi sono scorrelati: gli indirizzi dell'uno sono inutilizzabili nell'altro. In sintesi:

  - Modificano gli indirizzi IP e/o le porte dei pacchetti
  - Mantengono informazioni di stato relativamente al mapping fra gli spazi (binding cache)
  - Permettono di risparmiare indirizzi, inquanto solo chi accede alla rete pubblica ha un ID pubblico.

Si può distinguere tra source NAT (SNAT) e destination NAT (DNAT), a seconda che venga modificato l'indirizzo sorgente o l'indirizzo destinazione del pacchetto che inizia una nuova connessione.

- Nel source NAT, le connessioni effettuate da uno o più computer vengono alterate in modo da presentare verso l'esterno uno o più indirizzi IP diversi da quelli originali. Quindi chi riceve le connessioni le vede provenire da un indirizzo diverso da quello utilizzato da chi effettivamente le genera.
- Nel destination NAT, le connessioni effettuate da uno o più computer vengono alterate in modo da venire redirette verso indirizzi IP diversi da quelli originali. Quindi chi effettua le connessioni si collega in realtà con un indirizzo diverso da quello che seleziona.

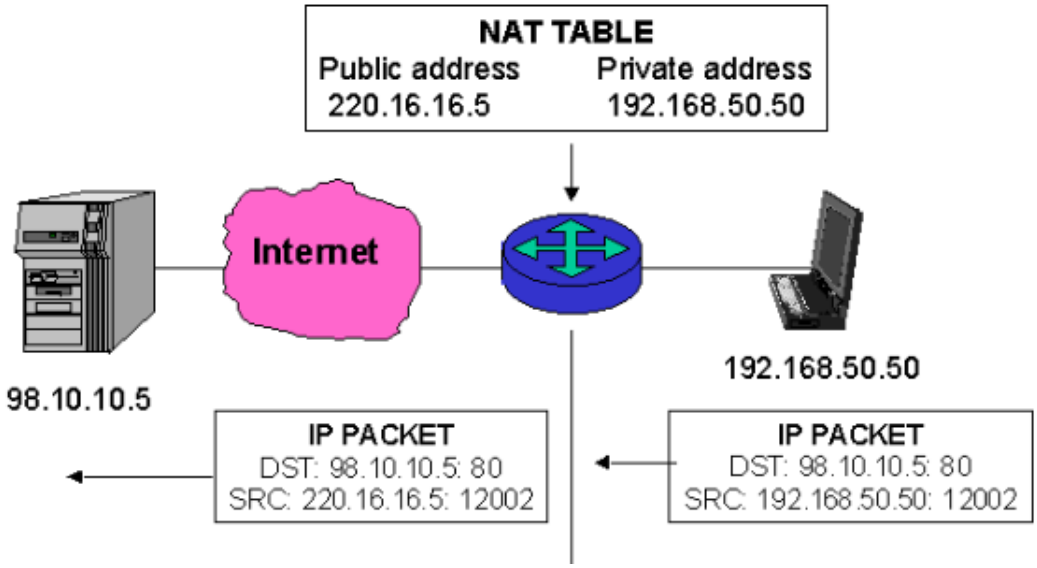

Figura 3.2: Scenario NAT

- *Quali soluzioni adottare?:* 2 possibili soluzioni:
  - *Soluzione strutturali:* Il supporto del protocollo viene incluso all'interno della periferica di NAT oppure il client apre un "buco nel NAT". Richiedono il supporto da parte della periferica di NAT

    - **ALG:** [19] Application Level Gateway (gateway al livello di strato di applicazione fungono da collegamento tra due reti; sono delle entità capaci di "guardare dentro" i messaggi di protocollo e di permettere di passare solo a quei messaggi conformi a certe politiche predefinite.

  - *Soluzione non strutturale:* Il client determina il tipo di NAT mediante STUN e usa un host esterno con indirizzi pubblici, che diventa un server STUN/TURN.

    - **STUN:**[20] E' un protocollo client-server che permette a un client di scoprire se si trova dietro un NAT e permette di determinare il tipo di servizio NAT esaminando lo scambio di informazioni tra l'host (su cui si attiva un client STUN) ed un server STUN sul lato pubblico del NAT. Il server STUN è generalmente al di fuori del firewall / NAT quindi fa parte del dominio Internet pubblico. Il client invia dei messaggi STUN esplorativi al server STUN. Il server utilizza tali messaggi per determinare l'indirizzo IP pubblico e le porte utilizzate e poi notifica il client. Il protocollo si basa su una semplice sequenza di domande-risposte con lo scambio di pacchetti UDP tra client e server.
    - **TURN:** [21] Traversal Using Relays around NAT è stato progettato per affrontare il problema dei NAT simmetrici. Permette ad un qualsiasi host dietro un NAT o un Firewall di ricevere dati in ingresso su connessioni TCP o UDP. Il server TURN lavora quindi da relay per il traffico. Provvede ad un meccanismo di mutua autenticazione tra client e server prima della fase di domande-risposte.
    - **ICE:** [22] Interactive Connectivity Establishment definisce le modalità di instradamento, scegliendo quali tra i protocolli STUN e TURN.

Delimitati alcuni dei problemi con i quali ci si impatta andando a considerare il contesto del Voip, possiamo definire un ulteriore quesito:

- *L'applicazione deve essere in grado di operare in presenza di firewall e NAT*

Da questa analisi sono sorti diversi problemi, nonostante nella documentazione dei rispetti software risultino tutte "NAT friendly" e vengono rilasciate con supporto STUN, con la possibilità di scegliere tra vari STUN server. *CSipSimple* permette addirittura l'utilizzo di ICE per la soluzione ai NAT.

Vediamo nella tabella seguente quale è stato il riscontro del test.

Wi-Fi with NAT	SIPDROID	LINPHONE	CSIPSIMPLE
Same Provider	*L.2*	*L.1*	*L.2*
Different Provider	*L.1*	*L.1*	*L.2*

Tabella 3.5: Risultati Test Wi-Fi in presenza di NAT

Nonostante questo supporto, i risultati ottenuti non sono ottimali. Osservando l'esito dei test, ci si riconduce a quello che è stato il punto di partenza di questo lavoro di tesi, cioè estendere una applicazione VoIP su dispositivi mobile Android, che fosse preferibilmente implementata con le librerie PJSIP, tramite l'introduzione di un SIP Proxy Client in grado di reindirizzare le connessioni VoIP verso l' architettura ABPS [23]. In particolare l'idea era quella di integrare ed estendere l'applicazione con delle funzionalità di proxy, quindi "meccanismo di proxy", incaricato di instradare i messaggi VoIP tra l'interfaccia virtuale e le reali interfacce fisiche.

3.2.1 ABPS: Always Best Packet Switching

- **Scenario:** Lo scenario a cui fa riferimento tale architettura è quello di un device mobile, equipaggiato con interfacce wireless eterogenee (NIC - Network interface cards), in grado di sfruttare completamente il vantaggio offerto dalle interfacce multiple a disposizione. Le attuali tecnologie infatti permettono al device di usare selettivamente solo una NIC. In alternativa, il modello ABPS estende le capacità di un device mobile con più interfacce di rete, fornendogli la possibilità di instradare

ciascun datagramm attraverso l'interfaccia più opportuna, consentendo in questo modo un uso simultaneo di tutte le NIC disponibili. Il vantaggio di avere a disposizione interfacce di rete eterogenee sullo stesso dispositivo mobile è evidente soprattutto nel caso di interfacce wireless, perché rende possibile la connettività nel caso venga meno una delle connessioni.

Un device mobile equipaggiato con NIC wireless eterogenne viene chiamato multi-homed Mobile Node (MN).

L'architettura fa uso di:

- Un server proxy fisso che svolge il ruolo di relay, con lo scopo di superare limiti di connettività dovuti alla presenza di firewall e NAT.
- Su ogni nodo mobile viene installato un proxy client SIP e RTP, che mantiene un tunnel multi-path con il server proxy, attraverso tutte le interfacce disponibili. Il canale logico così stabilito consente, attraverso un'estensione dei protocolli SIP e RTP/RTCP, che ogni pacchetto possa essere identificato anche quando cambia l'indirizzo IP di provenienza.

Nella seguente figura viene illustrato uno scenario composto da un terminale VoIP multihomed equipaggiato con due o più interfacce wireless (Es. Wi-Fi e 3G), collocato in una rete che fornita di copertura 3G e WiFi.

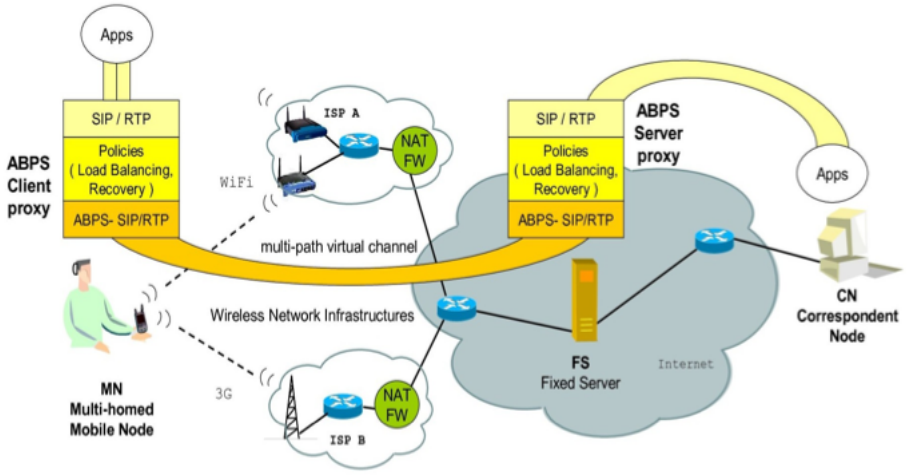

Figura 3.3: Architettura ABPS

Come ultimo criterio di giudizio è stato tenuto conto degli aspetti riguardanti l'usabilità e la praticità d'uso dell'applicazione stessa, oltre le features che mette a disposizione:

- **Flessibilità ed efficenza d'uso**
- **Interfaccia "user friendly" (semplice ed intuitivà)**
- **Sicurezza e robustezza agli errori**

Dopo aver considerato tutti i precedenti risultati ottenuti, l 'applicazione che risulta essere "più performante" è CSipSimple, un vero e proprio porting delle librerie PJSIP su Android con opportune modifiche relativamente alla gestione di codec audio e video (come definito nel capitolo precedente). Di seguito viene mostrato un report dei diversi test effettuati, in cui è possibile vedere tutti gli scenari possibili presi in considerazione e avere una visione più completa delle problematiche in cui ci si è impattati.

		SIPDROID		LINPHONE		CSIPSIMPLE	
		Same provider	Different provider	Same provider	Different provider	Same provider	Different provider
CONNETTIVITA'	3G	*L.3*	*L.3*	*L.3*	*L.3*	*L.3*	*L.3*
	Wi-Fi without NAT	*L.3*	*L.3*	*L.3*	*L.2*	*L.3*	*L.3*
	Wi-Fi with NAT	*L.2*	*L.1*	*L.1*	*L.1*	*L.2*	*L.2*
	STUN Support	√		√		√	
	ICE Support	—		—		√	
CHIAMATA	Instant Messaging	—		√		√	
	Addressbook	√		√		—	
	Multi-codec Support	√		—		√	
	ZRTP/SRTP Support	—		√		√	
	Multi-account Support	*2 accounts*		—		√	

Tabella 3.6: Valutazione globale dei Test

Capitolo 4

Progettazione

4.1 La piattaforma Android

Android [24] è un vero e proprio stack di strumenti e librerie per la realizzazione di applicazioni mobili. Esso ha come obiettivo quello di fornire tutto ciò di cui un operatore, un vendor di dispositivi o uno sviluppatore hanno bisogno per raggiungere i propri obiettivi. Ha la fondamentale caratteristica di essere open, dove il termine assume diversi significati:

- è open in quanto si appoggia sulla versione 2.6 di Linux per servizi del sistema centrale, come sicurezza, gestione della memoria, esecuzione, network stack e driver model.
- è open in quanto le librerie e le API che sono state utilizzate per la sua realizzazione sono esattamente le stesse che vengono usate per la creazione dell'applicazioni.
- è open in quanto il suo codice è open source, consultabile da chiunque possa contribuire a migliorarlo, lo voglia documentare. E' distribuito sotto Open Source Apache License 2.0.

E' un'architettura che comprende tutto lo stack degli strumenti per la creazione di applicazioni mobili di ultima generazione, tra cui un sistema operativo, un insieme di librerie native per le funzionalità core della piattaforma, una implementazione della DVM (Dalvin Virtual Machine) e un

insieme di librerie Java. Si tratta di un'architettura a layer, dove i livelli inferiori offrono servizi ai livelli superiori offrendo un più alto grado di astrazione.

L'esame della presente architettura, descritta in Figura 1.4, ci permetterà di comprendere al meglio l'utilizzo delle diverse funzionalità.

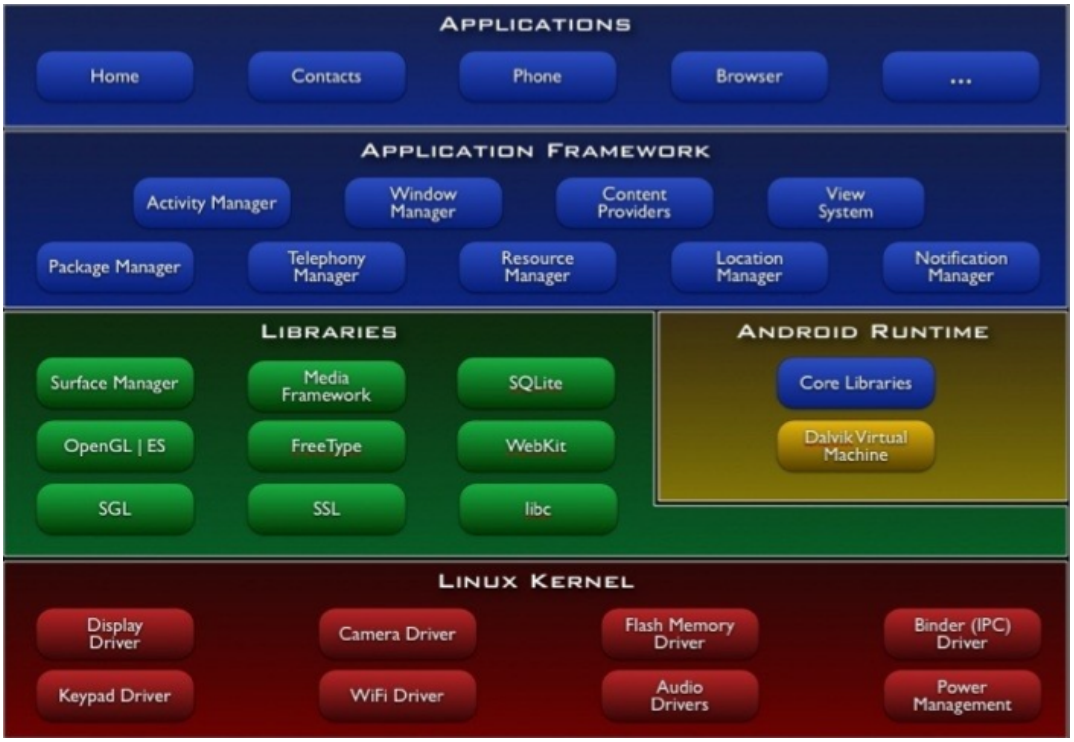

Figura 4.1: Architettura di Android

- ***Linux Kernel.*** Il layer di più basso livello è rappresentato dal kernel Linux nella versione 2.6. La necessità era infatti quella di disporre di un vero e proprio sistema operativo che fornisse gli strumenti di basso livello per la virtualizzazione dell'hardware sottostante attraverso la definizione di diversi driver, il cui nome è completamente esplicativo. In particolare, si possiamo notare la presenza di driver per la gestione delle periferiche multimediali, del display, della connessione Wi-Fi e dell'alimentazione. La scelta verso l'utilizzo di un kernel Linux è stata conseguenza della necessità di avere un SO che fornisse tutte le feature di sicurezza,gestione della memoria, gestione dei processi, power management e che fosse affidabile e testato.

- ***Native Libraries.*** Sopra il layer costituito dal kernel di Linux 2.6 abbiamo un livello che contiene un insieme di librerie native realizzate in C e C++, che rappresentano il core vero e proprio di Android. Si tratta di librerie che fanno riferimento a un insieme di progetti Open Source e che hanno le seguenti funzionalità:

  - *Surface Manager:* è un componente fondamentale in quanto ha la responsabilità di gestire le view, ovvero ciò da cui un'interfaccia grafica è composta. Il compito del SM è infatti di coordinare le diverse finestre che le applicazioni vogliono visualizzare sullo schermo. Ciascuna applicazione è in esecuzione in un processo diverso e disegna quindi la propria interfaccia in tempi diversi. Il compito del SM è di prendere le diverse finestre e di disegnarle sul buffer da visualizzare poi attraverso la tecnica del double buffering. In questo modo non si avranno finestre che si accavallano in modo scoordinato sul display. Notiamo quindi che si tratta di un componente di importanza fondamentale, specialmente in un'architettura che basa sulla capacità di creare interfacce interattive molta della sua potenza.Il Surface Manager avrà quindi accesso alle funzionalità del display e permetterà la visualizzazione contemporanea di grafica 2D e 3D dalle diverse applicazioni:
  - *Open GL ES:* un'importante caratteristica di Android è la possibilità di utilizzare sia grafica 3D sia grafica 2D all'interno di una

stessa applicazione. La libreria utilizzata per la grafica 3D è quella che va sotto il nome di OpenGL ES, la quale permette di accedere alle funzionalità di un eventuale acceleratore grafico hardware. Si tratta di una versione ridotta di OpenGL specializzata per dispositivi mobili. Si tratta di un insieme di API multipiattaforma che forniscono l'accesso a funzionalità 2D e 3D in dispositivi embedded.

- *SGL:* la Scalable Graphics Library è una libreria in C++ che insieme alle OpenGL costituisce il motore grafico di Android. Mentre per la grafica 3D ci si appoggia all'Open GL, per quella 2D viene utilizzato un motore ottimizzato chiamato appunto SGL. Si tratta di una libreria utilizzata principalmente dal Window Manager e dal Surface Manager all'interno del processo di renderizzazione grafica.
- *Media Framework:* questo componente è in grado di gestire i diversi CODEC (MPEG4, H.264, MP3, AAC, AMR) per i vari formati di acquisizione e riproduzione audio e video, che nell'architettura di Android si chiama Media Framework.
- *FreeType:* per la gestione dei font, android utilizza il motore di rendering dei font FreeType (http://freetype.sourceforge.net). Si è deciso di utilizzare questo motore perché è di piccole dimensioni, molto efficiente, altamente customizzabile e soprattutto portabile. Attraverso FreeType, le applicazioni di Android saranno in grado di visualizzare immagini di alta qualità. Il suo punto di forza è quello di fornire un insieme di API semplici per ciascun tipo di font in modo indipendente dal formato del corrispondente file.
- *SQlite:* è una libreria in-process che implementa un DBMS relazionale caratterizzato dal fatto di essere molto compatto, diretto, di non necessitare alcuna configurazione e soprattutto essere transazionale. SQLite è compatto in quanto realizzato completamente in C in modo da utilizzare solo poche delle funzioni ANSI per la gestione della memoria. È diretto in quanto non utilizza alcun processo separato per operare ma "vive" nello stesso processo

dell'applicazione che lo usa, da cui il termine in-process. Viene utilizzato spesso in sistemi embedded perché non necessita di alcuna procedura di installazione. Nonostante la compattezza e la semplicità d'uso, permette di gestire gli accessi ai dati in modo transazionale.

- *WebKit:* si tratta di un browser engine open source basato sulle tecnologie HTML, CSS, JavaScript e DOM. Essendo un browser engine e non un semplice browser, andrà integrato in diversi tipi di applicazioni.
- *SSL:* si tratta della libreria per la gestione dei Secure Socket Layer.
- *Libc:* si tratta di un'implementazione della libreria standard C libc ottimizzata per i dispositivi basati su Linux embedded come Android.

- ***Core Libraries.*** Per le applicazioni Android in fase di compilazione c'è bisogno del jar (di nome android.jar) per la creazione del bytecode Java, mentre in esecuzione il device metterà a disposizione la versione dex del runtime che costituisce appunto la core library. Nel dispositivo non c'e' infatti codice Java, in quanto non potrebbe essere interpretato da una JVM,ma solamente codice dex eseguito dalla DVM.

- ***Application Framework.*** Tutte le librerie viste vengono poi utilizzate da un insieme di componenti di più alto livello che costituiscono l'Application Framework (AF). Si tratta di un insieme di API e componenti per l'esecuzione di funzionalità ben precise che si possono sintetizzare cosi:

  - *Activity Manager:* si occupa dell'organizzazione delle varie schermate di un'applicazione in uno stack a seconda dell'ordine di visualizzazione delle stesse sullo schermo dei diversi dispositivi.
  - *Package Manager:* la responsabilità del Package Manager è di gestire il ciclo di vita delle applicazioni nei dispositivi, analogamente a quanto avviene in ambiente J2ME da parte del Java Application Manager (JAM).

- *Window Manager:* permette di gestire le finestre delle diverse applicazioni, gestite da processi diversi, sullo schermo del dispositivo. Si può quindi considerare come un'astrazione dei servizi nativi del Surface Manager descritto in precedenza.
- *Telephony Manager:* permette un'ottima interazione con le funzionalità caratteristiche di un telefono come la semplice possibilità di iniziare una chiamata o di verificare lo stato della chiamata stessa.
- *Content Provider:* ha la responsabilità di gestire la condivisione di informazioni tra i diversi processi. Il suo funzionamento è simile a quello di un repository condiviso con cui le applicazioni possono interagire inserendo o leggendo informazioni.
- *Resource Manager:* ha la responsabilità di gestire informazioni che riguardano i file di configurazione, file di definizione del layout, attraverso una serie di API. Dunque fornisce ottimizzazione delle risorse.
- *View System:* consente la gestione della renderizzazione dei componenti nonchè la gestione degli eventi associati.
- *Location Manager:* contiene una serie di API che gestiscono la localizzazione e tutte le funzionalità legate alla location, tra cui le operazioni di georeferenzazione.
- *Notification Manager:* mette a disposizione un insieme di strumenti che l'applicazione può utilizzare per inviare una particolare notifica al dispositivo.

4.2 Android SDK

Le applicazioni di Android sono sviluppate all'interno di un framework, una struttura dati specifica. Se si utilizza l'ambiente di sviluppo(SDK) [25] con Eclipse, la struttura del framework è molto chiara. Tramite l'SDK (o meglio: tramite gli strumenti utilizzati mediante l'SDK) trasformiamo la nostra applicazione Android in un codice intermedio chiamato bytecode; questo è esattamente quello che accade abitualmente in Java. Questo bytecode viene eseguito dalla Dalvik Virtual Machine (DVM) descritta nel paragrafo precedente. Ogni terminale Android ha la sua DVM, come definito nell'architettura del sistema; il suo compito è solo questo: eseguire il bytecode.

- *Creazione, Compilazione, Emulazione:* Tramite l'SDK possiamo passare dalla descrizione dell' applicazione alla sua effettiva esecuzione sia in emulazione, sia sul dispositivo. Per descrivere l'applicazione al dispositivo prescelto si utilizza il file Manifest.xml Possiamo quindi affermare che un'applicazione è descritta completamente da:
  - Codice Java
  - Risorse statiche xml
  - Manifest.xml

L'SDK di Android è composto da pacchetti modulari che è possibile scaricare separatamente utilizzando il Android SDK Manager. Ci sono diversi pacchetti disponibile per l'SDK, tra i più importanti:

- SDK Tools: Contiene gli strumenti per il debug e il test, più altre utilità che sono necessarie per sviluppare un app.
- SDK Platform-tools: Contiene i tools platform-dependent per lo sviluppo e il debugging delle applicazioni.
- Documentazione, APIs, Google APIs, SDK Platform, ecc..

4.3 Ciclo di Vita di un Activity

La piattaforma Android organizza le activity secondo una struttura a stack dove quella posta più in alto è attiva in un certo momento. Visualizzare una nuova schermata, dunque l'avvio di una nuova activity, porterà quest'ultima in cima allo stack mettendo in uno stato di pausa le altre. Terminato il proprio lavoro, le eventuali informazioni raccolte saranno passate all'attività precedente, la quale diventerà nuovamente attiva. Ottimizzare le risorse prevede che una activity non visualizzata possa essere eliminata dal sistema per poi essere eventualmente ripristinata successivamente. Gli stati possibili per una activity sono rappresentati nella figura seguente:

Figura 4.2: Ciclo di vita di un Activity

- *Running:* L'activity è in cima allo stack, è visibile e ha il focus; può ricevere gli eventi da parte dell'utente.

- *Paused:* l'activity non è attiva ma è ancora visibile per la trasparenza di quelle superiori o perché queste non occupano tutto lo spazio a disposizione. Insensibile agli eventi da parte dell'utente viene eliminata dal sistema solo in caso di estrema necessità.
- *Stopped:* E' lo stato delle activity non attive né visibili; è insensibile agli eventi dell'utente ed è fra le prime candidate a essere eliminata.
- *Inactive:* Una activity si trova in questo stato quando viene eliminata o prima di essere creata.

4.4 Compilazione ed Installazione di CSipSimple

Per ottenere i sorgenti di CSipSimple basta digitare su una shell il seguente comando:

svn checkout http://csipsimple.googlecode.com/svn/trunk/CSipSimple/ csipsimple

Viene creato all'interno della directory dalla quale è stato lanciato il comando la cartella *csipsimple* che verrà poi popolata con il sorgente dell'applicazione. I codec e la libreria pjsip, non sono stati tradotti in java ma sono stati lasciati in linguaggio nativo (C, C++). A tal fine è utilizzato il kit NDK (Native Development Kit) [26] che permette di compilare sorgenti in linguaggio nativo in modo da creare delle librerie utilizzabili mediante il meccanismo JNI di Java. Per compilare questi sorgenti occorre modificare il file Application.mk contenuto all'interno della cartella /path-to-cisipsimple/jni inserendo al posto della prima riga:

APP_PROJECT_PATH := /path-to-csipsimple/

dove */path-to-csipsimple/* è il cammino assoluto alla directory *csipsimple*.

Dopo questo step, bisogna entrare nella cartella dell'NDK ed eseguire il seguente il comando:

./ndk_build_C/path-to-csipsimple/

Le librerie generate verranno salvate all'interno della cartella */path-to-csipsimle/libs/armeabi*.

Da qui è possibile procedere alla compilazione dell'intera applicazione importando il progetto con Eclipse. Una volta importato il progetto, Eclipse provvederà in automatico a lanciare il processo di compilazione.

4.5 Creazione Rubrica

Dopo aver esaminato ed investigato su quali fossero le features di cui dispone CSipSimple, si è deciso di implementare un' addressbook "ad hoc" per l'applicazione. In realtà non si tratta di una vera e propria rubrica che permette le operazioni classiche(inserisci contatto, modifica contatto, cancella contatto, ecc..), quanto di un "filtro" di contatti Sip (*Sip_Address*).

```
1  Uri uri = ContactsContract.Data.CONTENT_URI;
2  String[] projection = new String[] {
3          ContactsContract.Data._ID,
4          ContactsContract.Data.DISPLAY_NAME,
5          ContactsContract.CommonDataKinds.SipAddress.SIP_ADDRESS,
6          ContactsContract.CommonDataKinds.SipAddress.TYPE,
7          };
8  String selection = ContactsContract.Data.MIMETYPE+"='"+ContactsContract.
       CommonDataKinds.SipAddress.CONTENT\_ITEM\_TYPE+"'";
9  String[] selectionArgs = null;
10 String sortOrder = ContactsContract.Contacts.DISPLAY_NAME+ "
       COLLATE_LOCALIZED_ASC";
11 Cursor cursor = managedQuery(uri, projection, selection, selectionArgs,
       sortOrder);
```

L'idea di questa scelta implementativa è sostanzialmente dovuto al fatto che sarebbe un consumo di risorse disporre di due rubriche separate, di cui una esclusivamente per i contatti SIP. D'altronde una delle politiche di android è la gestione ottimale delle risorse, il cui il principio di base è:

- *Sfruttare al massimo le funzionalità di un'activity per eseguire distinte operazioni*

Quindi anzichè avere la lista di tutti i contatti interni al *Dialer* proprio di Android (Numero Telefonico, Contatto Email,), all'utente vengono visualizzati solo quelli che dispongono di un contatto per le *internet call*. Il funzionamento è semplice:

- All'utente viene visualizzata la lista dei soli contatti Sip
- L'utente seleziona il contatto desiderato
- L'utente effettua la chiamata Voip con CSipSimple

La nuova Rubrica è inserita all'interno del package *com.csipsimple* quindi aggiunta al *Manifest.xml* , vale a dire il file che descrive l'applicazione al dispositivo. Il *Manifest* elenca la lista delle necessità del programma per poter operare nel sistema. Il codice seguente mostra il contenuto del file, relativamente alla Main dell'applcazione:

```
<intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
<intent-filter android:priority="10">
        <action android:name="com.csipsimple.phone.action.DIALER" />
        <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
<intent-filter android:priority="10">
        <action android:name="com.csipsimple.phone.action.RUBRICA" />
        <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
<intent-filter android:priority="10">
        <action android:name="com.csipsimple.phone.action.CALLLOG" />
        <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
<intent-filter android:priority="10">
        <action android:name="com.csipsimple.phone.action.MESSAGES" />
        <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

Nelle seguenti figure vengono mostrati i layout di CSipSimple, con la nuova rubrica e la seguenza di passi per effettuare le chiamate.

Figura 4.3: Nuovo Main CSipSimple

Il Dialer mostra tutta la lista di contatti presenti sul dispositivo, sia SIP che non.

Figura 4.4: Dialer CSipSimple

La nuova Rubrica filtra i contatti SIP:

Figura 4.5: Rubrica

Un volta selezionato il contatto, la procedura di chiamata termina come avveniva il precedenza:

Figura 4.6: Dialer dopo il “filtro” dei contatti SIP

La chiamata può essere effettuata:

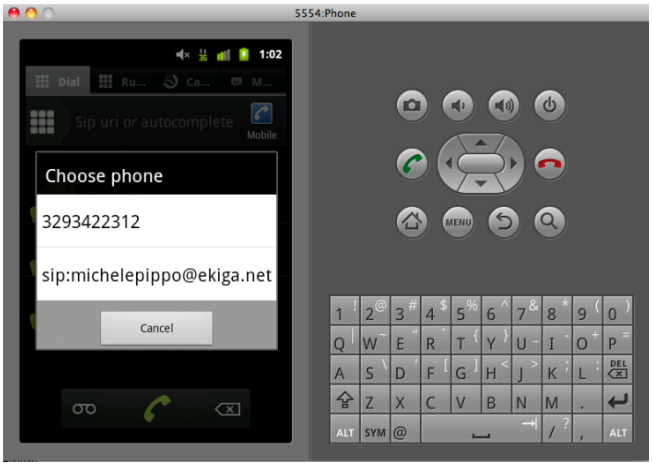

Figura 4.7: Chiamata CSipSimple

Per realizzare tale rubrica è stata aggiunta una nuova classe **Rubrica** che estende *ListActivity*, dunque si tratta di una vera e propria activity. L'idea di questa scelta implementativa è dovuta al fatto che si vuole sfruttare la logica "a stack" delle activity, mediante i vari metodi di callback.

3

Il seguente diagramma di sequenza illustra le operazioni e il principio di esecuzione dell'applicazione:

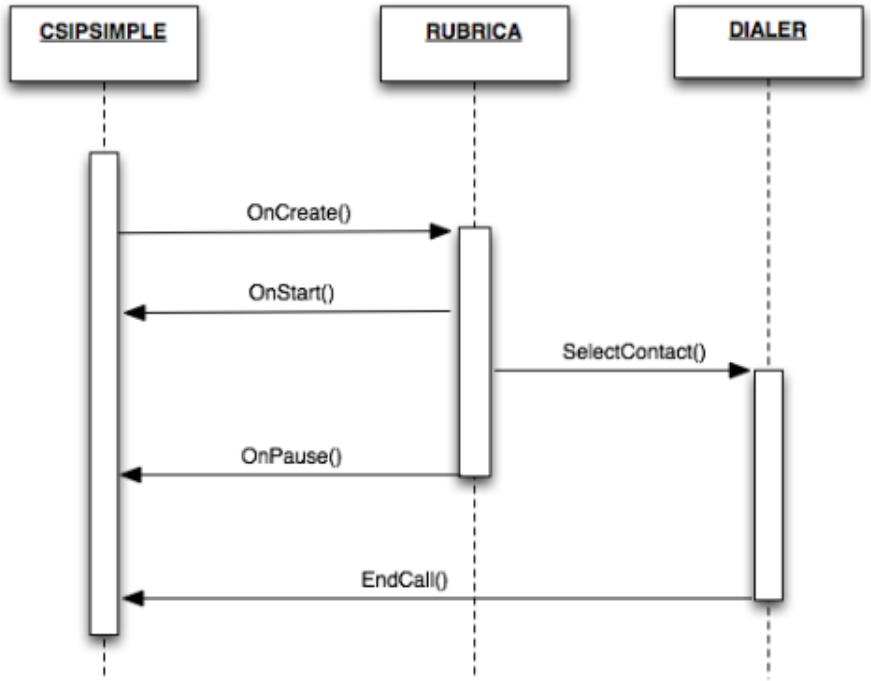

Figura 4.8: Sequenze Operazioni della Rubrica

Si nota come, una volta che è stato selezionato il contatto, l'activity **Rubrica** va in uno stato di pausa e sarà successivamente in cima allo stack delle activity (quindi *onStart()*) quando si vorrà effettuare una nuova chiamata.

La gestione delle chiamate è compito del *Dialer*, così come la gestione dei contatti, esattamente come avveniva nella applicazione prima delle modifiche.

L'interfaccia della Rubrica è stata resa coerente con tutto il resto dell'applicazione, aggiungendo semplicemente un nuovo wizard al Menù di *CSipSimple*.

Conclusioni

Oggi il VoIP é diventato un enorme business sia per le compagnie telefoniche tradizionali sia per nuove aziende specializzate che offrono servizi dedicati a privati ed aziende, dunque oggetto di studio continuo.

Con lo sviluppo di questa tesi si è voluto rappresentare in termini pratici alcune delle problematiche relative al mondo del Voip e tutte quella serie di tecnologia legate a questo contesto, passando dalla valutazione prestazionale di applicazioni open source, allo studio degli aspetti implementativi delle quali queste sono composte.

Sono stati studiati e mostrati i protocolli sui quali poggiano i servizi Voip, e suggerite delle soluzioni per garantire QoS nelle applicazioni che ne fanno uso.

E' stato introdotto l'architettura Android, una piattaforma innovativa e in costante evoluzione, toccando con mano le fasi di sviluppo di una tipica applicazione, mediante la realizzazione di una nuova funzionalità per l'applicazione CSipSimple. In questa applicazione si è avuto un riscontro pratico dell'utilizzo della libreria PJSIP che era il vero punto di partenza di questo lavoro. Data la complessità del progetto su cui ci si è trovati a lavorare e le diverse problematica conseguenti, il risultato ottenuto non è stato dei migliori, ma sicuramente stimolante e ha garantito la comprensione dei concetti sui quali Android stesso si fonda.

Molte delle difficoltà riscontrate in fase di studio e di sviluppo sono derivate dal fatto che l'applicazione CSipSimple stessa ha subito notevoli cambiamenti durante questo periodo, quello più evidente è l'adozione di una libreria grafica che si occupa di tutta la gestione dei layout (molto più efficiente rispetto alla versione sulla quale si è operato, a discapito di una struttura più articolata).

Nella fase di Test delle varie applicazioni si è avuto la prova di come, su dispositivi mobili, quindi con risorse limitate, sia più performante la libreria *PJSIP* rispetto a *MjSip* utilizzata in *Sipdroid* e *Liblinphone*, la core engile di *Linphone*.

E' stata presentata l'architettura ABPS e proprio a tal proposito, in futuro sarebbe utile permettere la configurabilità di CSipSimple con il proxy server ABPS, oltre che con un generico provider VoIP pubblico (come avviene attualmente), che era sostanzialmente il vero obiettivo prefissato.

Bibliografia

[1] **VoIP:** http://en.wikipedia.org/wiki/Voice_over_IP

[2] **H.323** http://en.wikipedia.org/wiki/H.323

[3] **J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley and E. Schooler. SIP: Session Initiation Protocol. IETF RFC 3261, Giugno 2002.**

[4] **H. Schulzrinne, A. Rao, R. Lanphier, RTSP: Real Time Streaming Protocol. IETF RFC 2326, Aprile 1998**

[5] **J. Postel, UDP: User Datagram Protocol, IETF RFC 768, Agosto 1980**

[6] **M. Handley, V. Jacobson, C. Perkins, SDP:Session Description Protocol IETF RFC 4566, Luglio 2006**

[7] **H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: A Transport Protocol for Real Time Applications. IETF RFC 3550,Luglio 2003**

[8] **M. Baugher, D. McGrew, M. Naslund, E. Carrara, K. Norrman, The Secure Real-time Transport Protocol (SRTP), IETF RFC 3711, Marzo 2004**

[9] **P. Zimmermann, A. Johnston, Ed., J. Callas, ZRTP: Media Path Key Agreement for Unicast Secure RTP, IETF RFC 6189, Aprile 2011**

[10] **Sipdroid:** http://sipdroid.org/

[11] **JSTUN: Java Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translation (NAT)** http://http://jstun.javawi.de/

[12] **MjSip** http://www.mjsip.org/doc/index.html

[13] **Linphone:** http://www.linphone.org/

[14] **Liblinphone:**http://nongnu.askapache.com/linphone/docs/liblinphone/index.html

[15] **CSipSimple:**http://code.google.com/p/csipsimple/

[16] **PJSIP:** http://www.pjproject.net

[17] **Firewall:** http://it.wikipedia.org/wiki/Firewall

[18] **NAT:** Livio Torrero, Problematiche di NAT Traversal

[19] **Application-level gateway(ALG):** http://en.wikipedia.org/wiki/Application-level_gateway

[20] **J. Rosenberg, J. Weinberger, C. Huitema, R. Mahy, STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs), IETF RFC 3488, Marzo 2003**

[21] **R. Mahy, P. Matthews, J. Rosenberg, Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN), IETF RFC 5766, Aprile 2010**

[22] **J. Rosenberg, Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols, IETF RFC 5245, Aprile 2010**

[23] **V. Ghini, S. Ferretti, F.Panzieri, The "Always Best Packet Switching" architecture for SIP-based mobile multimedia services.**

[24] **Massimo Carli. Android - Guida per lo sviluppatore**

[25] **Android SDK:** http://developer.android.com/sdk/index.html

[26] **Android NDK:** http://developer.android.com/tools/sdk/ndk/index.html