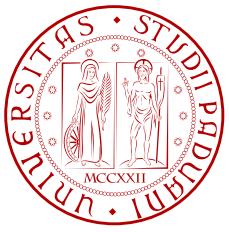


UNIVERSITÀ DEGLI STUDI DI PADOVA
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
CORSO DI LAUREA IN INGEGNERIA INFORMATICA
TESI DI LAUREA

STUDIO DI FATTIBILITÀ PER UN SISTEMA COLLABORATIVO SU WEB BASATO SU VOIP

RELATORE: Ch.mo Dr. Ing. Ferro Nicola

LAUREANDO: *Peruzzo Simone*

Padova, 7 Dicembre 2010

Indice

Elenco delle figure

Sommario

Il *Web 2.0* è un termine largamente impiegato per individuare quell'insieme di tecnologie che consentono l'aumento dell'interazione fra gli utenti del Web.
Voice over IP, la tecnologia impiegata per effettuare chiamate su rete IP, ben si adatta a questo tipo di esigenza.
In ambito aziendale, il centralino telefonico basato su questa tecnologia è uno strumento indispensabile per migliorare l'interazione con i propri clienti: inoltre, se integrato nel concetto di Customer Relationship Management, può dare vita ad un sistema in grado di aumentare la fidelizzazione del cliente sino a renderlo un promotore dei prodotti aziendali.

Il presente lavoro ha lo scopo di illustrare il sistema di assistenza telefonica progettato e prototipizzato in *Zucchetti*, un'azienda ICT che opera prevalentemente in ambito Nazionale ed Europeo; il progetto si basa su *Skype* un protocollo VoIP proprietario che offre ai programmatori la possibilità di accedere alle sue funzionalità attraverso le API(Application Programming Interface) pubbliche.
Nella parte conclusiva dell'elaborato vengo forniti degli spunti di sviluppo per migliorare il sistema sotto il punto di vista tecnologico e per meglio adattarlo alle esigenze dell'azienda.

Ringraziamenti

Ora che mi trovo a dover ringraziare tutte le persone che sono state importanti nella mia vita, mi rendo conto che una pagina non riuscirà a comprenderle tutte.

La prima persona a cui esprimo la mia riconoscenza e che ringrazio infinitamente è il Dott. Ing. Nicola Ferro, un docente di cui ho potuto apprezzare, oltre alle indiscutibili capacità professionali, le doti umane: le sue preziose indicazioni hanno reso possibile la produzione di un buon lavoro.

Una ringraziamento lo devo all'ing Michel Glèlè Ahanhanzo di Zucchetti che, nonostante non fosse il mio tutor aziendale, si è sempre reso disponibile per sciogliere i miei dubbi.

Un grandissimo grazie lo devo ai miei genitori Mario e Pierina e a mia sorella Eleonora, per tutta la pazienza e i sacrifici che hanno dovuto sopportare per sostenere le ingenti spese per il compimento del percorso di studi: vi sarò per sempre riconoscente. Un saluto lo rivolgo ai mia nonna Nisia, e ai miei nonni Teresina Cesio e Giuseppe che da qualche anno sono in cielo. Un ringraziamento per tutto il sostegno morale che mi stato dato da mia zia Rosanna, Pino, Toni, Renso, Lauretta, Romeo e Leda e a tutte le mie cugine.

Un ringraziamento lo devo a Daniele e Carlo, le persone con cui ho condiviso gioie e dolori della vita universitaria: è anche grazie a loro se sto scrivendo questa pagina.

Un grazie va inoltre a tutti gli amici di Cittadella e Curtarolo che mi hanno sostenuto nei momenti difficili. Il primo pensiero va a Stefano, un amico con cui ho condiviso la mia vita sin da bambino: un ringraziamento va anche a tutti gli altri, a partire dal mio vicino Matteo e poi a Dario, Daniele, Flavio, Fabio, Riccardo, Federica, Rami, Mike, Mirco, Tella, Brocca, Martino, Fris e a tutti coloro che non ho dimenticato di menzionare in questa lista.

Un grande ringraziamento è rivolto ai miei cugini Marco e Federico e ai miei grandissimi amici Vanny e Loris: con loro ho condiviso momenti indimenticabili.

L'ultima dedica la rivolgo alla Laura che da ormai sette mesi è al mio fianco e che in questo periodo ho trascurato per dedicare tutto me stesso allo studio . . . grazie per avermi capito.

Introduzione

La storia del web, per quanto recente, è fatta di continue novità e scoperte: proprio per la sua natura, non ha mai smesso di crescere e di svilupparsi, di attirare a sé gente e di proporre novità. Da quando è stato ideato si trova in una perpetua fase di miglioramento ed innovazione, oltre che di espansione grazie al sempre maggiore numero di persone che utilizzano questo strumento.

Il web 2.0 fa parte di questa crescita e può essere visto come il prodotto (non ancora finale) di anni di continuo sviluppo del web. L'incremento di versione del web non si riferisce ad un aggiornamento delle specifiche del World Wide Web, bensì ad una vera e propria rivoluzione, dal lato concettuale, dell'utilizzo della piattaforma da parte degli sviluppatori e degli utenti stessi.

Il punto cardine del concetto di web 2.0 è l'acquisita centralità dell'utente nel processo di partecipazione alla crescita del Web. L'utente riveste un ruolo da protagonista in quanto insostituibile fonte di informazioni: l'idea fondamentale è pensare l'utente di un servizio web non più come passivo utilizzatore di un contenuto, ma attivo e collaborativo. Attorno a ciò ruotano tutti i concetti innovativi della logica del web 2.0 come la collaborazione e la condivisione delle informazioni.

In questo contesto si inserisce la tecnologia per effettuare le chiamate sul web, nota sotto il nome di Voice over IP (VoIP). VoIP non è solo un modo per risparmiare denaro; sta diventando il combustibile che alimenta il Web 2.0 in quanto, grazie ad esso, il grado di interattività della rete aumenta.

Molte aziende che hanno sviluppato applicazioni web 2.0 stanno cogliendo la nuova opportunità offerta dal VoIP per migliorare la collaborazione con gli utenti,a vendo come obiettivo l' ottimizzazione del processo di assistenza diretta al cliente. Un caso di studio che verrà proposto in quasto lavoro è un centralino VoIP che sfrutti le funzionalità offerta da Skype, un software di Instant Messaging (IM) e VoIP diffusissimo della rete.

Anche gli ambienti di ricerca stanno cogliendo l'importanza della maggiore interattivà offerta da un sistema Voice over IP (VoIP) per aumentare la collaborazione fra utenti e promotori di pubblicazioni nei portali di divulgazione scientifica.

Anche per questo caso viene proposto un caso di studio di un servizio denominato Ask the Expert, in cui un utente interessato ad una particolare tematica può contattare la persona più adatta a fornire spiegazioni su quel particolare argomento.

Il presente lavoro viene suddiviso in sei capitoli.

Nel primo capitolo viene fornita un'introduzione storica sull'evoluzione della rete Internet e del Voice over IP, il servizio di trasporto delle comunicazioni vocali sulla rete. Il capitolo si conclude con un confronto fra il servizio di telefonia offerto dalla rete PSTN (Public Switched Telephony Network) e il VoIP.

Nel secondo capitolo viene fornita una descrizione dell'architettura a livelli di un sistema VoIP, illustrando i protocolli sviluppati da enti di standardizzazione che sono più diffusi.

Nel terzo capitolo viene approfondito il protocollo VoIP di Skype, che viene impiegato come base per i casi di studio introdotti precedentemente

Nel quarto capitolo si parlerà del software di Instant Messaging e VoIP di Skype, e delle funzionalità offerte. Verranno inoltre descritte le Application Programming Interface (API) rese disponibili dagli sviluppatori di Skype per la creazione di applicazioni esterne che possano estendere le funzionalità offerte dal client.

Il lavoro si concluderà con due casi di studio del VoIP per aumentare la collaborazione fra utenti di un' applicazione Web.

In particolare, nel quinto capitolo si descriverà il caso di studio del servizio di assistenza clienti progettato e parzialmente implementato durante il periodo di stage presso l'azienda Zucchetti S.p.a. Verranno inoltre proposti alcuni sviluppi futuri per migliorare il servizio.

L'azienda Zucchetti

L'azienda Zucchetti [zuc] annovera oltre 1.800 addetti operanti nello sviluppo di soluzioni software, hardware e servizi innovativi per soddisfare le specifiche esigenze di:

- aziende di qualsiasi settore e dimensione, banche e assicurazioni;
- professionisti (commercialisti, consulenti del lavoro, avvocati, curatori fallimentari, notai ecc.), associazioni di categoria e CAF;
- pubblica amministrazione Locale e Centrale (Comuni, Province, Regioni, Ministeri, società pubbliche...).

Diverse sono le sedi distribuite sul territorio nazionale: a Padova, Zucchetti è presente con una piccola sede (10 dipendenti) specializzata nello sviluppo di soluzioni innovative.
Per ulteriori informazioni si visiti il sito di riferimento dell'azienda.

Capitolo 1

Storia delle reti per i servizi vocali

La rete telefonica tradizionale o Public Switched Telephony Network (PSTN) è stata oggetto di un'evoluzione considerevole dal momento in cui nel 1876 *Alexander Graham Bell* realizzò la prima trasmissione vocale su cavo[1], attraverso un circuito denominato *ring-down*. Questo semplice meccanismo si è evoluto nel tempo da una trasmissione unidirezionale (*one-way*) dove un utente poteva solamente parlare ad una trasmissione bidirezionale (*two-way*) dove entrambi gli utenti potevano parlare.
Il mercato iniziale del mondo della telefonia riguardò la vendita dei telefoni: gli apparecchi erano venduti a coppie e spettava al cliente tirare un cavo tra i due telefoni. L' impatto del business dei telefoni sul territorio fu devastante. In poco tempo le città furono coperte da cavi che passavano alla rifusa sopra le abitazioni: questa conseguenza fu dovuta al fatto che, se un utente voleva parlare con altri n proprietari di telefoni, doveva tirare cavi separati tra la sua e le altre n abitazioni. Divenne immediatamente chiaro che questo modello non poteva funzionare.

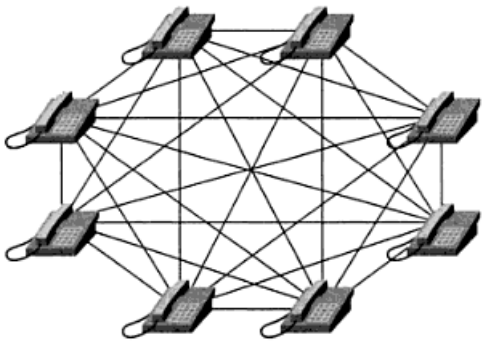


Figura 1.1: grafo completo del modello iniziale della telefonia

[1] la paternità del telefono è una questione ancora aperta

Bell se ne accorse, e costituì la Bell Telephone Company, azienda che aprì il suo primo ufficio di commutazione (anche conosciuto con il termine inglese *switch* [2]) nel 1878.

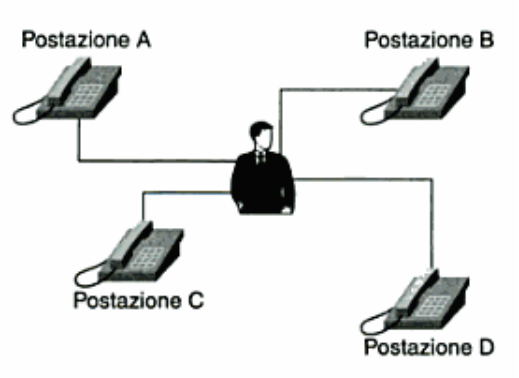


Figura 1.2: modello di telefonia con commutazione effettuata da un operatore umano

In breve tempo apparvero uffici di commutazione un po' ovunque e l'utenza incominciò a richiedere comunicazioni interurbane: per rispondere a questa nuova esigenza, la società decise di collegare tra di loro tutti i suoi uffici di commutazione. Ciò causò un problema già affrontato in precedenza: collegando ogni ufficio di commutazione a tutti gli altri si ottenne una nuova invasione di cavi. Il problema fu sostanzialmente risolto allo stesso modo, introducendo centrali di commutazione di secondo livello. Le sempre più grandi esigenze dell'utenza telefonica hanno portato alla crescita della gerarchia della rete, fino a raggiungere i cinque livelli. Già nel 1890 le tre parti principali del sistema telefonico erano state preparate: gli uffici di commutazione, i cavi tra clienti e centrale e le connessioni su lunghe distanze. Da allora sono stati apportati miglioramenti a tutti e tre i settori, ma il modello di base del sistema è rimasto sostanzialmente lo stesso per 100 anni.

Attorno agli anni '50 furono poste le premesse per la progettazione di una rete di nuova generazione, che 50 anni dopo sarebbe diventata la base tecnologica della telefonia. Nel pieno della Guerra Fredda, il dipartimento della difesa USA commissionò una rete che fosse più robusta della PSTN: come riportato in [DP01] e [Cas01], doveva essere un sistema di comunicazione invulnerabile ad un attacco nucleare. All'epoca tutte le comunicazioni militari usavano la rete telefonica pubblica, considerata vulnerabile: il motivo di questo timore si può capire osservando la figura 1.4(a).

I punti neri rappresentano le centrali di commutazione telefonica, ognuna delle quali era connessa a migliaia di telefoni. A loro volta queste centrali erano colle-

[2] Inizialmente fu un operatore umano ad agire come switch: chiedeva al chiamante il destinatario della comunicazione e manualmente connetteva i due percorsi di rete. Successivamente l'operatore umano venne sostituito da apparati elettrici di commutazione

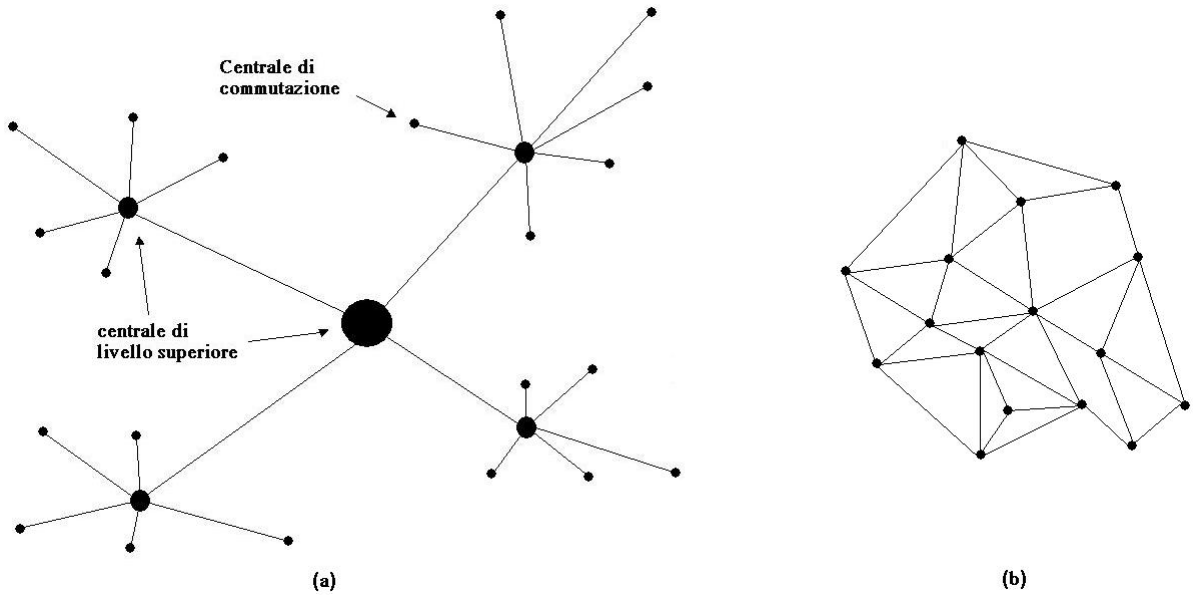


Figura 1.3: (a) Struttura della rete telefonica (b) Il sistema distribuito a commutazione proposto da Baran

gate da centrali di commutazione di livello superiore per formare una gerarchia nazionale con poca ridondanza. La vulnerabilità del sistema stava nel fatto che la distruzione di poche centrali di alto livello avrebbe frammentato la rete in molte isole separate. Attorno al 1960 il dipartimento della difesa assegnò il compito di trovare la soluzione alla RAND Corporation: un suo dipendente, *Paul Baran*, sviluppò un progetto altamente distribuito e resistente ai guasti, mostrato in figura 1.4(b). Nel suo modello veniva proposto l'uso della *tecnologia digitale a commutazione di pacchetto*: il percorso tra due centrali di commutazione qualunque era diventato molto maggiore di quello che i segnali analogici potevano percorrere senza distorsioni. Le sue idee furono apprezzate dagli ufficiali del Pentagono a tal punto da richiedere ad AT&T[3] la costruzione di un prototipo. AT&T rigettò le idee di Baran: secondo [DP01] il motivo del rigetto è da rintracciarsi nel fatto che un colosso della telefonia dell'epoca non poteva farsi dare lezioni su come costruire reti da un "giovane esaltato". Questa idea riprese vigore nel 1967 grazie a *Larry Roberts*, portando alla realizzazione di una rete che divenne noto sotto il nome di *ARPANET* Quattro anni più tardi la rete contava quindici nodi: la maggior parte erano centri di ricerca universitari.

Nei tardi anni '70 l'organismo statunitense National Science Foundation (NSF) vide l'enorme impatto che ARPANET aveva sulla ricerca universitaria, permettendo a scienziati di tutto il paese di condividere i dati e collaborare a progetti di ricerca. Tuttavia per collegarsi ad ARPANET un'università doveva avere un contratto di ricerca aperto con il ministero della difesa, e molte non lo avevano.

[3]monopolista della rete telefonica USA fino al 1984

NSF reagì a questa situazione progettando una rete aperta a tutti i gruppi di ricerca universitari: venne costruita una backbone per collegare i suoi centri che ospitavano supercomputer. Ad ogni supercomputer fu affiancato un microcomputer chiamato *fuzzball*. I fuzzball erano collegati tra loro formando una subnet con la stessa tecnologia hardware usata da ARPANET. La tecnologia software era invece differente: i fuzzball usarono sin dall'inizio TCP/IP, costituendo così la prima WAN TCP/IP. Dopo il 1983, quando ARPANET e NSFNET furono interconnesse e TCP/IP divenne l'unico protocollo ufficiale, la crescita del numero di utenti divenne esponenziale; in seguito si unirono altre reti regionali e furono creati collegamenti dagli USA verso il Canada e l'Europa. Nella metà degli anni '80 alcuni iniziarono a vedere l'insieme di reti come una internet, che più tardi venne considerata l'Internet per eccellenza. Il collante che tiene insieme Internet è formato dal modello di riferimento TCP/IP e dalla corrispondente pila di protocolli. Uno studio sulla crescita di Internet può essere trovata in [Com03]

All'inizio degli anni '90 con l'enorme diffusione di Internet e la conseguente affermazione a livello mondiale della suite di protocolli TCP/IP, si incominciò a parlare di VoIP: il termine venne coniato per identificare un nuovo modello di servizio telefonico. Questo nuovo modello di telefonia prevedeva la trasmissione della voce attraverso la rete, sfruttando il modo proprio di Internet di spedire dati e informazioni. L'idea di base era semplice: convertire la voce catturata da un microfono in pacchetti dati da inviare sulla rete al destinatario; una volta giunti a destinazione, i pacchetti venivano riconvertiti permettendo così la trasmissione della voce. VocalTec lanciò nel 1995 il primo prototipo di servizio di voce sulla rete, denominato *InternetPHONE*. Il sistema era composto da un software installato sui computer che effettuava la codifica e la decodifica della voce; i computer si collegavano ad un server, che dietro richiesta li metteva in collegamento. Nonostante la tecnologia fosse ancora in fase pionieristica, la novità ebbe un grosso successo fra gli utenti di Internet perché permetteva collegamenti a grandi distanze a costi decisamente inferiori a quanto normalmente si pagava con la tradizionale rete telefonica.

Successivamente, tra il 1999 e il 2000 c'è stato un altro momento importante, quando il mercato dell'Information Tecnology, spinto dai venti di Borsa, ha incoraggiato la nascita di diverse società di comunicazione basate sul VoIP: Dialpad, Mediaring, FreePhone sono alcuni dei nomi che nascono e si muovono nel mercato in quel periodo. Dal 1995 la tecnologia VoIP ha fatto passi avanti notevoli: il nuovo modo di effettuare chiamate tramite computer diventa di consuetudinario, specialmente per quelle a lunga distanza, dove il modo di funzionare proprio di Internet permette risparmi enormi. Passati i venti di Borsa e scomparse molte delle società nate da idee brillanti, ma senza solide fondamenta economico - finanziarie, sono rimasti sul mercato alcuni gruppi che sono cresciuti arrivando ai giorni nostri: ne sono esempi Skype, VoipStunt, Skypho, Eutelia e Tele2.
In [Par06] e [Lon05] vengono descritti in modo dettagliato i servizi offerti da alcuni operatori VoIP

Anche le società che operavano sulla tradizionale PSTN hanno compreso già da

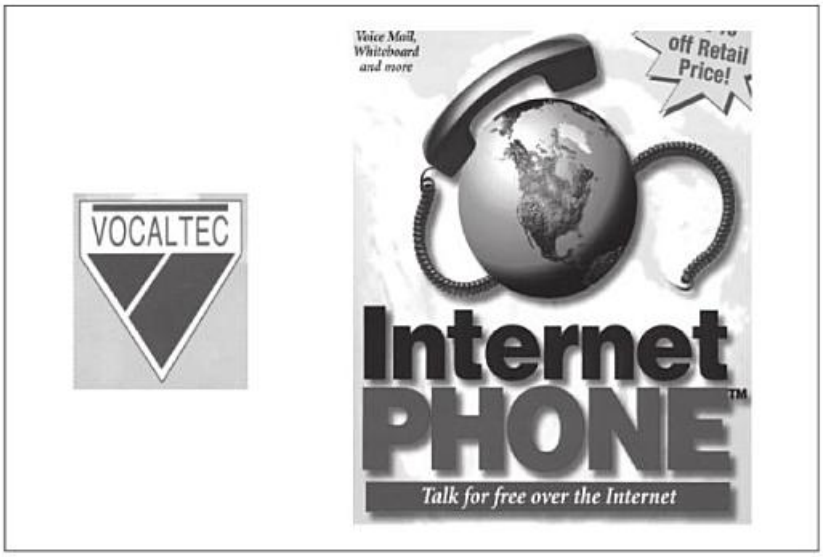


Figura 1.4: Logo di VocalTec e immagine di InternetPhone

qualche anno l'importanza della trasmissione della voce sotto forma di pacchetti dati, e hanno cominciato a sfruttare il VoIP per rendere maggiormente efficienti i loro sistemi. Telecom Italia usa il VoIP non direttamente sui clienti finali, ma per trasmettere le chiamate su backbone. Fastweb va oltre: insieme alla voce offre la televisione, anch'essa trasmessa con il medesimo sistema. Per approfondimenti, si veda [Lon05].

In questo breve viaggio nella storia della telefonia, è stato detto che la vecchia PSTN è basata su comunicazione a *commutazione di circuito*, mentre l'innovativa VoIP su un servizio a *commutazione di pacchetto*. La parte rimanente di questo capitolo viene impiegata con lo scopo di comprendere le diverse modalità con cui avviene la comunicazione su queste reti.

1.1 La commutazione di circuito

[LF02] definisce una rete a commutazione di circuito come l'insieme dei nodi di commutazione, dei collegamenti tra i nodi, degli apparati di connessione utente - nodo che consente uno scambio di informazioni tra gli utenti tramite l'instaurazione di connessioni fisiche dedicate temporanee. Sostanzialmente, ad una richiesta di comunicazione, i commutatori del sistema creano un percorso fisico sino al terminale dell'utente ricevente.
Questa tecnica, impiegata nelle reti telefoniche PSTN, è mostrata schematicamente in figura 1.5.

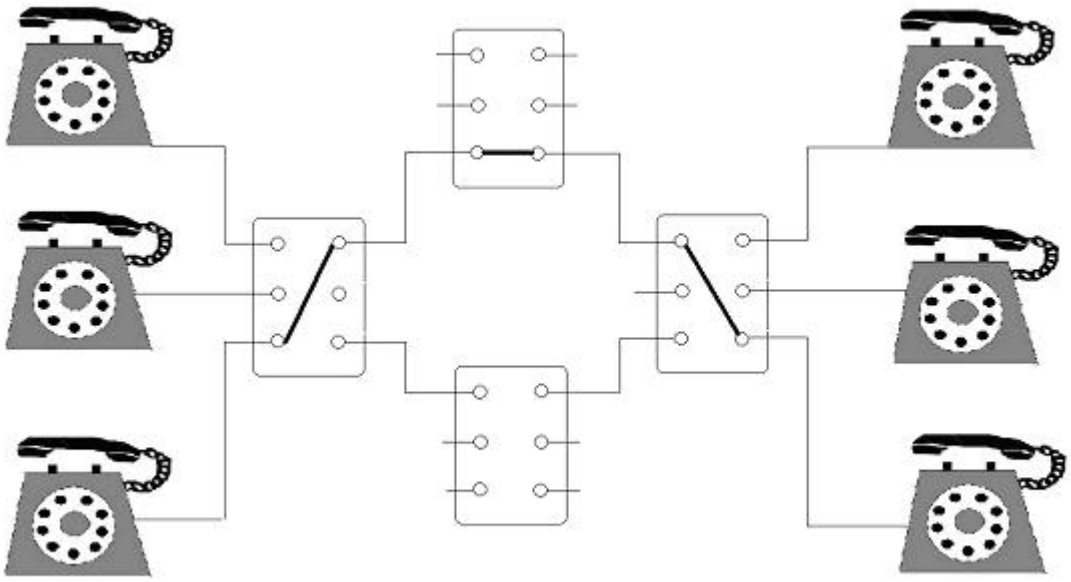


Figura 1.5: *schema di commutazione di circuito*

L' evoluzione della comunicazione attraverso una rete a commutazione di circuito viene descritta in [PD08] come una sequenza di tre fasi.
Nella prima fase[4], antecedente alla comunicazione, viene stabilito un circuito dedicato fra due terminali attraverso l'instaurazione di connessioni end - to - end fra nodi intermedi (*switch*) della rete. Questa fase viene completata attraverso un test che valuti se il ricevente è occupato o disposto ad accettare la connessione.
Successivamente avviene la fase di comunicazione (*Data Transfer*), in cui viene inviato un flusso di bit al destinatario sfruttando il percorso costruito nella fase precedente. Infine avviene la fase di conclusione della comunicazione (*Circuit disconnect*): l'autore della chiusura della connessione invia la segnalazione di tale evento al nodo a cui è collegato. La segnalazione viene quindi propagata verso i nodi intermedi del circuito, in modo che possano deallocare le risorse dedicate a questa comunicazione.

L'alternativa alla commutazione di circuito è la commutazione di pacchetto.

[4]nota come *Circuit establishment*

1.2 Commutazione di Pacchetto

Il principio di funzionamento della commutazione di pacchetto è ben spiegato in [PD08] e [Tan04].
Diversamente dal caso della commutazione di circuito, in questa forma di comunicazione, non viene stabilito a priori alcun percorso fisico fra chiamante e ricevente. Il blocco di dati che il mittente deve inviare, viene passato alla prima centrale di commutazione (*router*) assieme alle informazioni sul destinatario, utilizzando un elemento noto sotto il nome di *pacchetto*.
Le reti a commutazione di pacchetto adottano la strategia nota sotto il nome *store - and - forward*: ogni pacchetto completo ricevuto da un router attraverso una linea di connessione, viene memorizzato per poi essere esaminato al fine di verificare la presenza dei errori. Successivamente viene trasmesso verso la destinazione utilizzando le informazioni allegate al pacchetto e la sua conoscenza della rete.

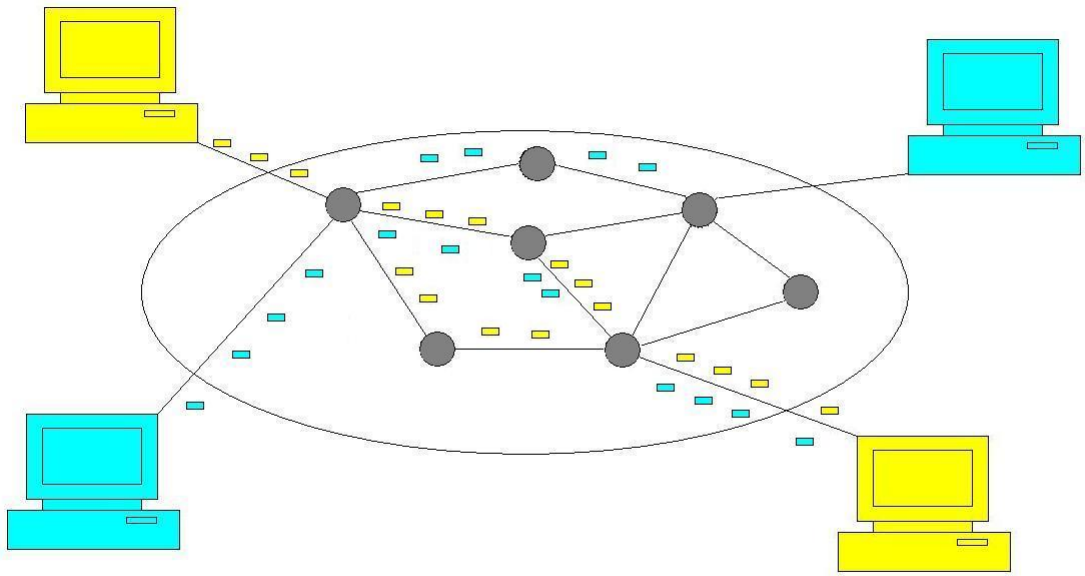


Figura 1.6: *schema di commutazione di pacchetto*

Secondo [PD08] il motivo principale per cui nelle reti di calcolatori viene usata la commutazione di paccheto invece della commutazione di circuito è l'efficienza. Nella prossima sezione verrà approfondito i confronto fra queste due modalità di comunicazione.

1.3 Commutazione di Pacchetto e Commutazione di Circuito a confronto

La commutazione di circuito e di paccheto, come spiegato in [Tan04] sono molto diverse. Innanzitutto mentre la commutazione di circuito necessita della configurazione di un circuito prima dell'inizio della comunicazione, la commutazione di pacchetto non richiede alcuna preimpostazione; il primo pacchetto può essere inviato appena disponibile. La configurazione della connessione usata nella tecnica a commutazione di circuito permette di riservare l'ampiezza di banda per tutto il percorso che unisce il trasmettitore e il ricevitore. Tutti i pacchetti seguono questo percorso, quindi i dati arrivano già nell'ordine corretto. Nel caso della commutazione di pacchetto non c'è alcun percorso predefinito, perciò diversi pacchetti possono seguire percorsi differenti che dipendono dalle condizioni della rete al momento della trasmissione; inoltre, i dati possono arrivare non in ordine.

La commutazione di pacchetto è inoltre più resistente agli errori: [Tan04] sostiene che è stata inventata proprio per questo motivo. Se un commutatore si blocca, tutti i circuiti che lo utilizzano vengono chiusi e nessun dato può essere inviato; nel caso della commutazione di pacchetto, i pacchetti possono essere instradati aggirando i commutatori bloccati.

Un' ulteriore differenza fra le due strategie consiste acquisizione della banda per la trasmissione. La preimpostazione di un percorso offre anche la possibilità di riservare in anticipo ampiezza di banda: in questo modo, all'arrivo di un pacchetto i dati possono essere inviati immediatamente attraverso la banda riservata. La commutazione di pacchetto grneralmente non riserva alcuna ampiezza di banda, perciò ogni pacchetto deve aspettare il proprio turno. Quando si riserva l'ampiezza di banda in anticipo, i pacchetti non possono sovraccaricare la linea (a meno che non si presentino più pacchetti di quelli attesi). D'altra parte, il tentativo di stabilire un circuito potrebbe non andare a buon fine se la la linea è sovraccarica: così il sovraccarico si presenta in momenti diversi con la commutazione di circuito (durante l'impostazione iniziale della connessione) e con la commutazione di pacchetto (durante la trasmissione dei pacchetti). Se un circuito è stato riservato per un particolare utente e non c'è traffico da trasmettere, l'ampiezza di banda di quel circuito è sprecata in quanto non può essere utilizzata per gestire altro traffico. La commutazione di pacchetto non spreca l'ampiezza di banda e perciò è più efficiente dal punto di vista generale del sistema. Comprendere questo compromesso è cruciale per capire la differenza tra la commutazione di circuito e quella di pacchetto. Una soluzione garantisce il servizio ma spreca risorse, l'altra non garantisce il servizio ma non spreca le risorse. In [JE03] si parla di *allocazione dinamica* della banda per le reti a commutazione di pacchetto e di allocazione statica per quelle a commutazione di circuito

L' ultimo elemento di confronto è l'introduzione di ritardi. La tecnica store and forward aggiunge ritardo durante la comunicazione: quando un pacchetto viene accumulato nella memoria del router, per poi è inviato al router successivo, viene

introdotto un ritardo. Con la commutazione di circuito, il tempo viene perso nalla costruzione del percorso dedicato: questo ritardo è noto sotto il nome *call setup delay*.

Dopo un'introduzione sull'evoluzione della telefonia Voice over IP, nei prossimi capitoli ne verrà descritta la struttura su cui si basa e i protocolli che gestiscono il suo funzionamento.

Capitolo 2

Il Voice over IP

In questo capitolo verrà descritta l'architettura a livelli su cui si basa Voice over IP. Successivamente verranno approfonditi il protocollo Real Time Protocol (RTP), progettato per effettuare trasmissioni real - time e i protocolli H.323 e Session Initial Protocol (SIP), utilizzati nella gestione del controllo della chiamata per la tecnologia Voice over IP

2.1 Il modello a strati di Voice over IP

Il modello proposto da [DP01], rappresentato in figura 2.1, si basa su una struttura a tre livelli.

La voce viene trasmessa tramite un'infrastruttura a commutazione di pacchetto, il livello di controllo della chiamata è distinto dal livello del mezzo e le interfacce API aperte consentono agli Independent Software Vendor di creare nuovi servizi.

Il livello standard dell'infrastruttura a pacchetti

Questa infrastruttura è rappresentata da IP (Internet Protocol), anche se questo modello può funzionare quando si utilizzano altri protocolli: a tale proposito [Har03] e [K$^+$01] utilizzano il termine **VoX** per indicare la classe di tecnologie adibite al trasporto della voce.
Vo è l'abbreviazione di *voce*, mentre *X* identifica il particolare protocollo utilizzato per il trasporto: fra questi, i più citati in letteratura sono Frame, ATM, ..., e ovviamente IP.

VoIP è la tecnologia che ormai monopolizza il mercato del trasporto della voce su rete a commutazione di pacchetto, grazie alla grandissima diffusione di IP e dell'architettura di cui ne è il protocollo fondamentale: l'*architettura TCP/IP*.

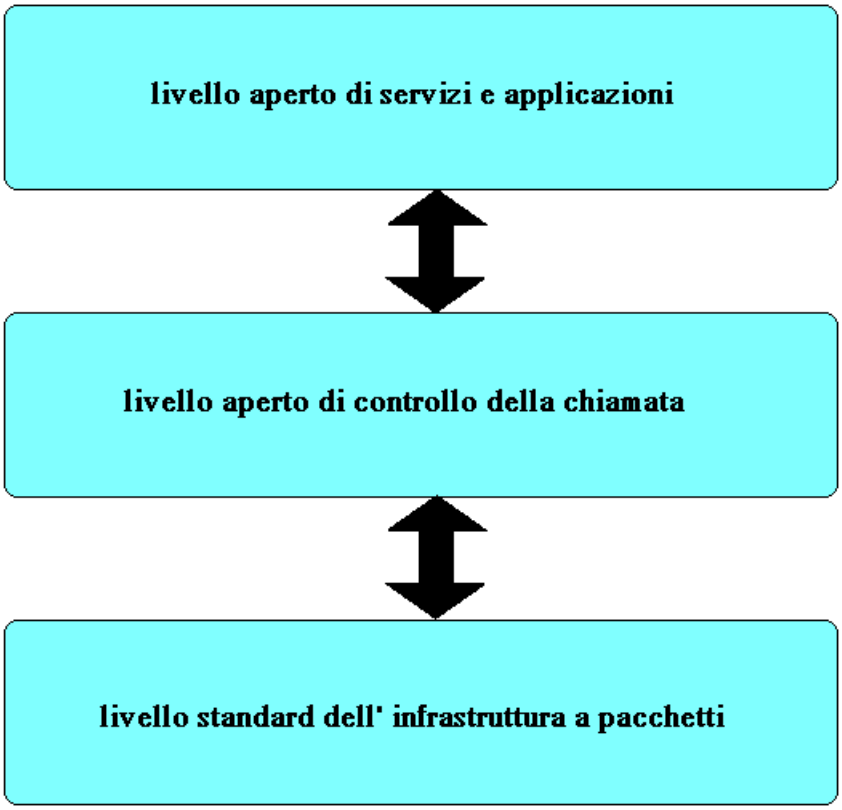


Figura 2.1: Il modello di Voice over IP

Quest'architettura, rappresentata in figura 2.2, prevede quattro strati:

- Il livello Host - network
- Il livello Internet
- Il livello Transport
- Il livello Application

Il livello Host - network

Nel livello Host - network è presente una grande varietà di protocolli, rappresentati in figura 2.2 con NET1, . . . , NETn; questi protocolli sono implementati come combinazione di hardware (un adattatore di rete) e di software (un componente del sistema operativo per la gestione dell'adattatore *network device driver*).
In questo strato si collocano protocolli come Ethernet e FDD. L'architettura Internet effettua una sola ipotesi in merito a ciò che avviene in questo livello: l'host deve connettersi alla rete utilizzando un protocollo che consenta il trasporto di pacchetti IP su di essa.

Il livello Internet

Il livello successivo nella gerarchia TCP/IP è denominato *livello Internet*: in [Tan04] viene definito *il perno che tiene assieme l'intera architettura.* Il suo compito è di permettere ad un host di inserire pacchetti in una qualsiasi rete in modo che questi viaggino indipendentemente verso la destinazione (potenzialmente anche su reti diverse). Essi possono arrivare anche in ordine diverso rispetto a quello di partenza; in tal caso, l'operazione di riordino verrà effettuata da qualche protocollo dei livelli superiori dell'architettura. Il livello Internet definisce un formato di pacchetto ufficiale e il protocollo IP (Internet Protocol) (vedi appendice).

Il livello Transport

Il livello superiore al livello internet nel modello TCP/IP è noto come livello Transport: questo strato ha lo scopo di permettere ad entità pari livello presenti sugli host sorgente e destinazione di portare avanti una comunicazione.
Due protocolli di collegamento sono definiti in questo livello.
Il primo, Transmission Control Protocol (TCP) è un protocollo orientato alla connessione affidabile che permette a sequenze di byte originate su una macchina di essere consegnate senza errori su una qualsiasi altra macchina della rete. Sulla destinazione, il processo TCP ricevente riassembla i messaggi ricevuti nella sequenza in uscita. Il protocollo TCP gestisce anche il flusso di controllo per essere sicuro che un mittente veloce non possa sovraccaricare un ricevente lento con più messaggi di quelli che è in grado di gestire.
Il secondo protocollo di questo livello, User Datagram Protocol (UDP) è un protocollo inaffidabile, privo di connessione, per applicazioni che non desiderano la

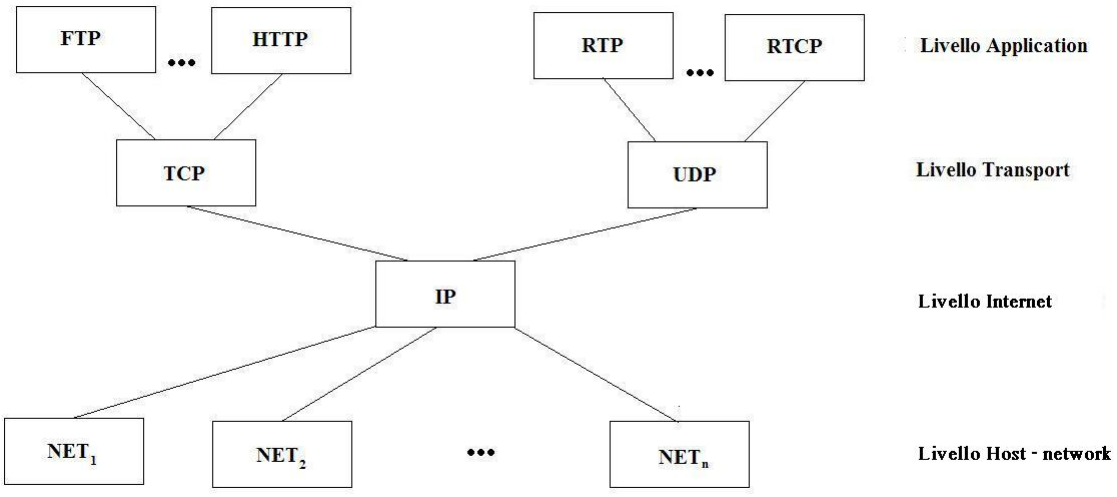


Figura 2.2: il grafo dei protocolli di Internet

sequenzializzazione o il del flusso del protocollo TCP e che desiderano gestire tutto questo in modo completamente autonomo. É spesso utilizzato per comunicazioni veloci, per richieste e risposte fra un cliente e un servente, o applicazioni in cui la prontezza della consegna è più importante che la sua accuratezza, come nel caso di trasmissioni audiovisive.

Il livello Application

Subito sopra al livello al livello di trasporto si trova il livello di application. Esso contiene tutti i protocolli ad alto livello. Per quanto riguarda le "applicazioni tradizionali", afferiscono a questo livello File Transfer Protocol (FTP) per il trasferimento di file, la posta elettronica Simple Mail Transfer Protocol (SMTP) e Hyper Text Transfer Protocol (HTTP), utilizzato per caricare pagine sul World Wide Web:

Per le applicazioni specializzate per il trasporto di dati si dati real - time il protocollo più conosciuto è RTP, che verrà approfondito in un'apposita sezione.

Il livello aperto del controllo di chiamata

Il controllo della chiamata, in poche parole, è il processo di decisione che riguarda sia l'indirizzamento delle chiamate che il modo in cui si deve svolgere una chiamata. Uno dei principali compiti del protocollo di controllo della chiamata è quello di dire agli stream RTP il punto in cui terminare e il punto in cui iniziare. Il controllo della chiamata svolge questo compito eseguendo la traduzione tra indirizzi IP e i piani di numerazione telefonica. Il modello di telefonia VoIP prevede la separazione dei *Bearer* (gli steam RTP) dal livello di controllo della chiamata e la separazione del livello di controllo della chiamata dai servizi è necessario garantire la possibilità di impiegare protocolli basati su standard. In questo senso, le reti dati sono peculiari in quanto nella stessa rete possono coesistere più protocolli che possono essere adattati a seconda delle esigenze. Ad esempio esistono vari protocolli di instradamento IP, ognuno dei quali è progettato in modo specifico per un determinato tipo di rete. Ognuno di essi risolve un problema simile: gli aggiornamenti delle tabelle di routing, ma ovviamente usando metriche e criteri di valutazione differenti. La stessa cosa si può affermare per i protocolli di controllo di chiamata Voice over IP, in quanto tutti risolvono lo stesso problema: la trasformazione di un numero telefonico in un indirizzo IP.

I protocolli più diffusi di questo strato Voice over IP sono H.323, SIP e Skype. H.323 e SIP, che saranno trattati nei prossimi paragrafi, utilizzano un modello architetturale *client - server*; essi sono inoltre riconosciuti come standard in quanto progettati da organismi di standardizzazione internazione. Skype, a differenza dei due protocolli precedentemente citati, si basa su un'architettura *peer - to - peer*; un'ulteriore caratteristica che lo contraddistingue è il fatto di essere un pro-

tocollo proprietario: è ritenuto uno *standard de facto* grazie alla sua grandissima diffusione.

La descrizione dei modelli *client - server* e *peer - to - peer* viene qui trattata brevemente

Il modello architetturale client - server

Il modello di comunicazione client - server, chiamato anche *sistema client - server*, è caratterizzato da due processi: il *client* e il *server*.

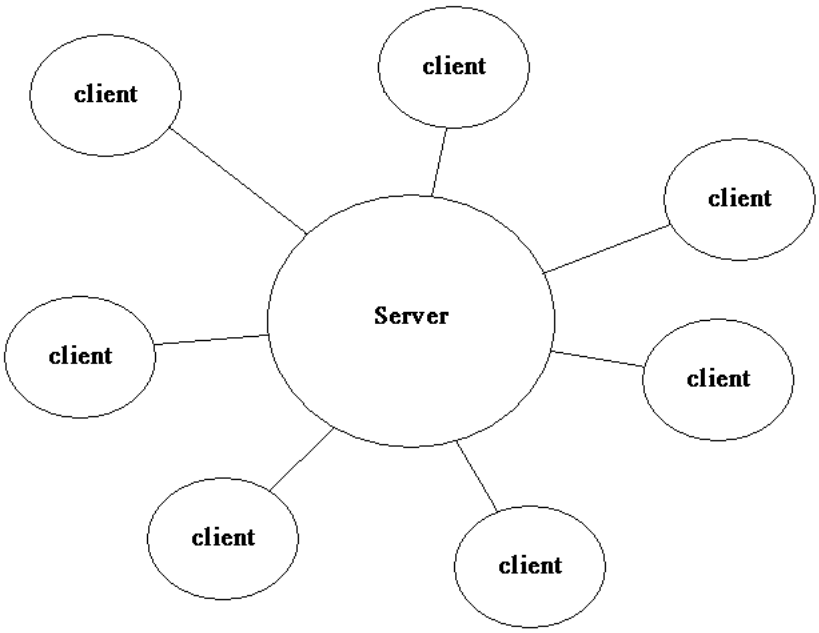


Figura 2.3: schema dell'architettura client - server

Questo modello è caratterizzato dalla relazione di servizio secondo la quale più processi cliente possono avviare un dialogo richiedendo servizi forniti da processi server corrispondenti. Il dialogo fra processi client e server, rappresentato in figura 2.4, è generalmente di tipo sincrono, ovvero:

- il processo server è generalmente inattivo, in attesa che gli pervenga una richiesta di servizio;
- il processo client, quando genera una richiesta di servizio, rimane bloccato in attesa della risposta

Il modello client - server è inoltre caratterizzato dall'accentramento delle risorse: le risorse necessarie per fornire il servizio sono concentrate nel server. Diretta

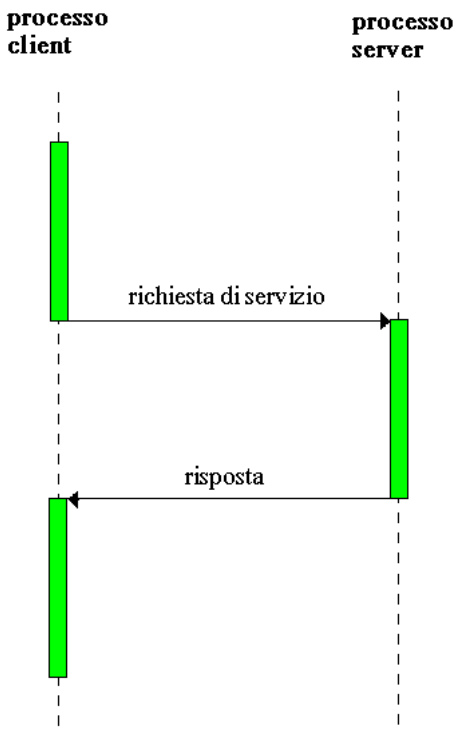


Figura 2.4: Relazione di servizio fra processo client e processo server

conseguenza di questa distribuzione delle risorse è la scara scalabilità del modello: all'aumentare del numero di richieste, le prestazioni di un sistem client - server degradano. Questo si verifica perché le prestazioni del sistema dipendono dal server; i miglioramenti possono avvenire solamente investendo risorse per l'aggiornamento della configurazione del server.

Infine, questo modello si caratterizza per un basso grado di *fault-tolerance* (resistenza ai guasti): se il server subisce un guasto, viene a mancare la disponibilità del servizio e il funzionamento dell'intera struttura è compromessa.

Il modello architetturale peer - to - peer

Il modello peer - to - peer è stato introdotto per identificare la possibilità di collaborare e condividere risorse migliorando la scalabilità (evitando la dipendenza da punti centralizzati) ed eliminando la necessità di un'infrastruttura specifica.
Il peer - to - peer è un modo per implementare sistemi basati sulla nozione di aumento della decentralizzazione del sistema: lo scopo per cui è stato concepito è lo sfruttamento dalle risorse che sono distribuite. Concettualmente è un'alternativa al modello client-server.

Come descritto in [AST07], i processi che costituiscono un sistema peer - to - peer sono tutti uguali: di conseguenza l'interazione tra essi è quasi tutta simmetrica

poiché ogni processo può agire sia da client che da server. Dato questo comportamento simmetrico, le architetture peer - to - peer si sviluppano attorno al problema dell'organizzazione dei processi im una rete *overlay* (*overlay network*), ovvero una rete in cui i nodi sono costituiti dai processi e i collegamenti rappresentano possibili canali di comunicazione (di solito realizzati con connessioni TCP). In generale un processo non può comunicare direttamente con un altro processo arbitrario, ma deve inviare messaggi attraverso i canali di comunicazione disponibili.

Sostanzialmente un'overlay network (reti sovrapposte) sfrutta le funzionalità della rete sottostante per svolgere le proprie elaborazioni; è possibile immaginarla come una rete logica realizzata al di sopra di una rete fisica, come viene mostrato in figura 2.5.

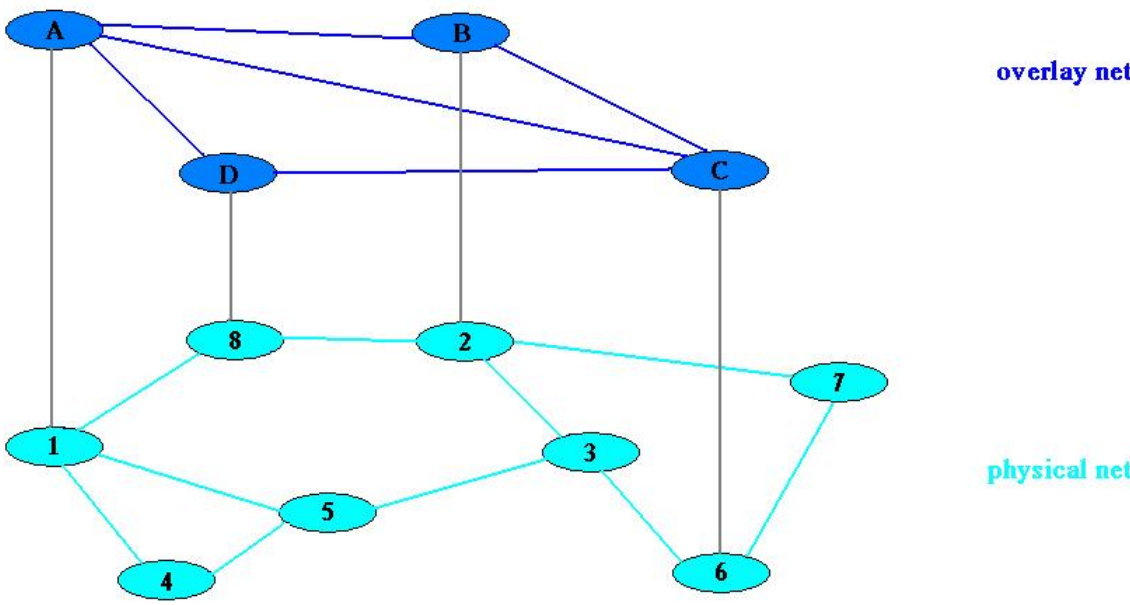


Figura 2.5: esempio di rete overlay

Una possibile classificazione dei sistemi peer - to - peer è mostrata in figura 2.6. Un sistema peer - to - peer può essere classificato in *puro* e *ibrido*: di seguito vengono descritte queste due tipologie.

Sistemi peer - to - peer ibridi

In un modello ibrido un server centralizzato contiene metadati riguardanti l'identità dei peer e le informazioni che memorizzano. Il server si comporta come un

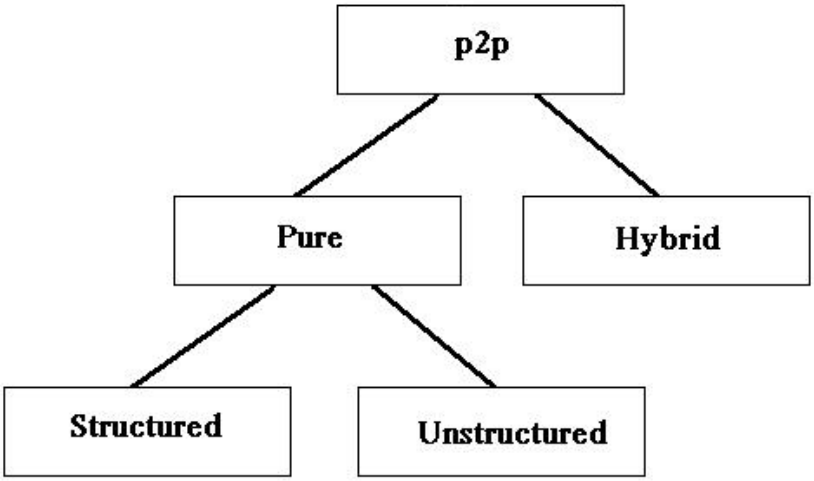


Figura 2.6: Classificazione dei sistemi peer - to - peer

'broker' e fa da intermediario in una comunicazione fra due peer. All'inizio la comunicazione avviene tra il server ed il richiedente per ottenere la locazione o l'identità di un peer che contiene l'informazione ricercata. Da questo momento in poi, la comunicazione procede direttamente fra il peer richiedente ed il peer che contiene l'informazione ricercata.

Sistemi peer - to - peer puri

In un modello puro non esiste un server centralizzato e tutti i peer ricoprono lo stesso ruolo nel sistema. La ricerca delle informazioni sulla rete avviene inviando domande ai peer vicini i quali, se non sono in grado di rispondere, inoltrano tali domande ai loro vicini e così via. I sistemi peer - to - peer puri si suddividono a loro volta in:

- peer - to - peer non strutturati
- peer - to - peer strutturati

I **sistemi peer - to - peer non strutturati** si basano su algoritmi casuali per la costruzione della rete overlay. Questo modello prevede la costruzione di una rete overlay simile ad un *grafo disordinato* in cui ogni nodo mantenga una lista di vicini (*vista parziale*) costruita in modo casuale. Si suppone inoltre che i dati siano posizionati sui nodi in maniera altrettanto casuale.

Di conseguenza, quando un nodo ha bisogno di trovare un dato, non conoscendo l'esatto posizionamento della risorsa di cui necessita, l'unica cosa che può effettivamente fare è inondare la[1] rete con un'interrogazione per la ricerca.

Una doverosa considerazione riguardante questa tipologia di rete è che, se da una parte viene penalizzata l'operazione di ricerca delle risorse sulla rete dall'altra vengono favorite l'aggiunta e rimozione dei nodi: poiché il grafo dell'overlay è random, non essendoci alcun vincolo sulla sua costruzione, per un nuovo nodo risulta semplice collegarsi e lasciare la rete.

Al contario dei precedenti, i **sistemi peer - to - peer strutturati** sono progettati in modo tale che l'overlay si sviluppi secondo un grafo avente una struttura piuttosto particolare, che consente la localizzazione delle risorse in modo efficiente in cambio della complessità introdotta nella fase di costruzione e manutenzione della rete. Una tecnica piuttosto nota per la localizzazione delle risorse prevede l'utilizzo di *tabelle di hash*.

Esistono anche soluzioni intermedie ai due modelli sopra descritti come quella dei coordinatori (*SuperPeers*).

Le reti super - peer rappresentano un incrocio tra i sistemi ibridi e i sistemi puri.

Come già spiegati in precedenza, specialmente nei sistemi peer - to - peer non strutturati, al crescere della rete può diventare problematico localizzare dati rilevanti. La causa di questo problema è semplice: non essendoci un modo deterministico di instradare una richiesta di ricerca per un dato specifico, l'unica tecnica a cui un nodo può fare ricorso è l'inondazione della rete con la richiesta. Per cercare di stroncare l'inondazione della rete, possono essere impiegati dei nodi speciali (**Superpeer**) che mantengono un indice dei dati e agiscono da intermediari. Come suggerisce il nome, i superpeer sono anch'essi organizzati in una rete peer - to - peer; un semplice esempio di tale organizzazione è mostrata in figura 2.7.

In questa organizzazione, ogni peer regolare è connesso a un superpeer. Tutte le comunicazioni da e verso un peer regolare passano attraverso il superpeer associato. In molti casi la relazione peer regolare - superpeer è fissa: quando un peer regolare si unisce alla rete, si connette a uno dei superpeer presenti e vi rimane finché non lascia la rete. In questo caso ci si aspetta che i superpeer siano processi a lunga vita e altamente affidabili. Per compensare il comportamento potenzialmente instabile di un superpeer è possibile ricorrere a schemi di backup, come il raddoppio di ogni superpeer e la richiesta ai peer regolari di connettersi ad entrambi.
Ci sono casi in cui l'associazione fissa non è la soluzione migliore. Per esempio, nel caso di una rete per la condivisione di file, è meglio che un client si connetta a un superpeer che mantiene un indice dei file cui il client è generalmente interessato. In questo caso, ci sono maggiori possibilità che, quando un client cerca un file, il suo superpeer sappia dove si trova.

[1] in letteratura, l'operazione di ricerca su reti peer - to - peer pure non strutturate è conosciuta col nome di *flooding*

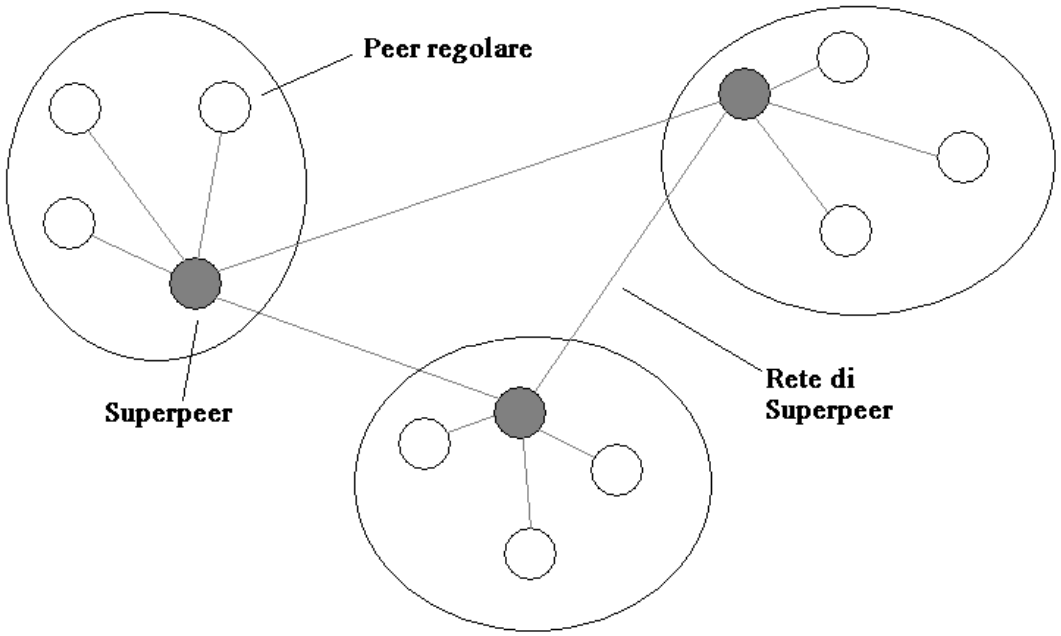


Figura 2.7: Organizzazione gerarchica dei nodi in una rete superpeer

Con le reti superpeer, se da una parte si limita il flooding della rete, dall'altra si introduce un nuovo problema: la modalità di selezione di nodi candidati a superpeer.
Il protocollo Voice over IP di Skype, come spiegato in [BS04] è un esempio di overlay network basato su superpeer in cui la cui scelta dei superpeer è ancora in fase di studio.

Il livello delle applicazioni e servizi

Questo livello è il più astratto, poiché non conta la specifica struttura della rete, ma soltanto le funzionalità che essa mette a disposizione. Secondo [DP01], con l'avvento dell'architettura VoIP, aumenta il mercato degli sviluppatori di applicazioni: quando si costruisce una rete che ha interfacce aperte fra il livello dei pacchetti e e il livello di controllo di chiamata e fra il livello di controllo di chiamata e il livello della applicazioni, le applicazioni non devono più essere sviluppate dai produttori: basta utilizzare l'API (Application Programming Interface) standard per avere accesso a tutti i servizi offerti dalla nuova infrastruttura. Proprio per questo motivo, vi è un grandissimo numero di Independent Software Vendor (ISV) che sono stati attratti da questa fascia di mercato. I servizi primari che un sistema VoIP deve offrire sono quelli messi a disposizione dalla vecchia PSTN: chiamata, identificazione del chiamante, attesa, numeri di emergenza, segreteria telefonica. Giunge a proposito la considerazione che viene fatta in [DP01]: quando si adotta una nuova infrastruttura, non è necessario replicare

tutte le funzionalità offerte dalla vecchia infrastruttura, ma solo le funzionalità e le applicazioni richieste dai clienti. Dopo aver convertito queste applicazioni, è possibile sviluppare migliaia di nuove applicazioni per le infrastrutture a pacchetti, per esempio interfacciandosi con altre applicazioni già sviluppate per una rete a commutazione di pacchetto: solo per citarne alcune, la posta elettronica e la messaggistica istantanea.

2.2 Problemi del VoIP

Una rete di comunicazione, per garantire a pieno una buona qualità del servizio, deve fornire un trasferimento affidabile dell'informazione e nello stesso tempo garantire l'assolvimento dei vincoli imposti da alcuni parametri, dalla particolare applicazione. Nel caso di comunicazioni VoIP, le costrizioni sono dettate dalla massima tolleranza ai *ritardi*, al *jitter* e alla *perdita dei pacchetti*. Le reti IP sono state progettate per il trasferimento interattivo di dati e non per applicazioni tempo reale, quali il trasporto della voce. I ritardi, il jitter e la perdita di pacchetti sono fenomeni tanto frequenti all'interno di Internet quanto difficili da contenere entro i limiti di accettabilità. Per le applicazioni classiche (per esempio FTP) la variabilità della QoS è un parametro tollerabile. Al contrario, per applicazioni di telefonia su IP la garanzia di una buona QoS si presenta come una grossa sfida. Qui di seguito vengono descritti i principali problemi precedentemente citati, che che affliggono il Voice over IP.

Ritardi

I ritardi, che inficiano il trasporto di voce su reti IP, sono di varia natura. Tali ritardi, se eccessivi, possono degradare molto la qualità della conversazione. Va tenuto presente che la natura dei ritardi è duplice: possono essere sia fissi che variabili. I ritardi fissi sono da attribuire ai tempi necessari per l'elaborazione del segnale ed in particolare agli algoritmi di codifica e di pacchettizzazione. Si tratta di ritardi predicibili la cui entità può essere facilmente quantificata priori.

I ritardi variabili sono meno predicibili e sono dovuti alla natura stessa della rete. Si è gia detto in precedenza che le modalità di trasmissione delle reti a commutazione di pacchetto non garantiscono a priori la necessaria quantità di banda e tanto meno un percorso dedicato. Per tale ragione, i ritardi introdotti dalla rete variano in dipendenza della disponibilità di risorse. Quest'ultima, a sua volta, è legata al carico della rete in un dato istante, il quale non è predicibile. Un router che deve inoltrare un pacchetto su una linea è costretto ad attendere finché, su quella linea, è disponibile banda sufficiente per la trasmissione.

Un'interessante soluzione, proposta per arginare il problema dei ritardi, è quella legata al protocollo RSVP (Resource ReSerVation Protocol): si tratta di un protocollo progettato per riservare una certa porzione di banda proprio ad applicazioni che richiedono risposte in tempo reale, quali quelle del VoIP.

E' evidente che l'implementazione di un sistema di comunicazione vocale su reti IP necessita di un costante monitoraggio dei ritardi associati ai pacchetti. Questa è una delle esigenze che ha dato vita al protocollo di trasporto RTP, il quale fornisce informazioni da cui è possibile risalire al ritardo relativo ad ogni pacchetto.

Jitter

Il peggior effetto della variabilità dei ritardi è il fenomeno del jitter. In sostanza, mentre in trasmissione i pacchetti sono generati ad un tasso costante, in ricezione arrivano secondo intervalli temporali casuali; questo fenomeno necessita di essere compensato attraverso l'implementazione di un jitter - buffer. L'implementare un sistema di voce su IP non si può prescindere da questo fenomeno: per la comprensione del parlato è necessario che l'ordine dei pacchetti in arrivo sia concorde con quello in trasmissione, mentre i dati non sempre rispettano questo vincolo.

L'algoritmo di decompressione vocale è fatto supponendo che le trame vocali arrivino ad intervalli costanti. Mentre il ritardo dovuto alla rete è aleatorio e la sua entità non è per niente predicibile. Ciò fa sì che la sequenza temporale, in trasmissione, non sia perfettamente mappata in ricezione. Dunque, non è possibile in alcun modo adattare l'algoritmo di decompressione al jitter dei pacchetti. La soluzione comunemente adottata è di realizzare un buffer all'interno del quale immagazzinare temporaneamente i pacchetti per poi presentarli al decodificatore secondo la giusta sequenza temporale e ad un tasso costante. Un corretto dimensionamento del jitter-buffer garantisce un ascolto dei pacchetti secondo la giusta sequenza ma introduce ulteriore ritardo. Questo ritardo introdotto è strettamente legato alla disponibilità di risorse della rete. Un approccio interessante è quello di implementare un buffer la cui dimensione vari secondo il carico della rete. A tale proposito sia il protocollo che il protocollo RTCP forniscono appositamente informazioni per la scelta del dimensionamento del buffer.

Perdita di pacchetti (Packet Loss)

La perdita di pacchetti è uno dei problemi più frequenti nelle reti dati ed esistono diverse soluzioni che mirano a porvi parziale rimedio. Contrariamente ai protocolli per il trasporto dati TCP, in VoIP l'obiettivo non è ricevere il messaggio assolutamente integro e corretto, ma riuscire a ricostruire il parlato con sufficiente fedeltà da soddisfare l'ascoltatore più attento. Pertanto la priorità assoluta è minimizzare il ritardo e il jitter. Grazie all'alta qualità raggiunta dai mezzi trasmessivi, i ritardi principali sono imputabili alle congestioni a livelli superiori. In caso di perdita di un solo pacchetto isolato, solitamente di distrugge 20 ms di parlato: l'ascoltatore medio non è in grado rilevare alcuna differenza nella qualità della voce. Per correggere questo errore, la soluzione più diffusa consiste nell'intraprendere una strategia di occultamento dell'errore (*Packet Loss Concealment*): dopo lo scadere del timeout del buffer di jitter, la coda del buffer attende al massimo per un certo tempo il pacchetto in ritardo, dopodiché cerca di mascherare l'errore ripetendo il campione precedente ed eventualmente scartando il pacchetto in ritardo, qualora arrivasse. In [DP01] si afferma che si può utilizzare la strategia di occultamento solo quando la perdita riguarda un solo pacchetto: se si perdono più pacchetti, viene comunque eseguita una volta sola.

Esistono altre strategie più complesse che interpolano il campione mancante sfruttando la forte correlazione fra i campioni vocali adiacenti.

2.3 Il protocollo RTP

RTP

Il protocollo RTP é stato definito dal Working Group Audio/Video Transport dell'Internet Engineering Task Force (IETF) nell'RFC[2] 1889(gennaio 1996) e successivamente in RFC 3550(luglio 2003).

Secondo [Tan04] e [PD08], la posizione di RTP nello stack dei protocolli dell'architettura TCP/IP è in un certo senso strana: nonostante questo protocollo sembri, dal nome, essere un protocollo dello strato di trasporto, viene trattato come protocollo del livello applicativo poiché contiene una notevole quantità di funzioni che sono specifiche per le applicazione multimediali. IETF ha quindi deciso di inserire RTP nello spazio utente e di eseguirlo appoggiandosi su UDP.

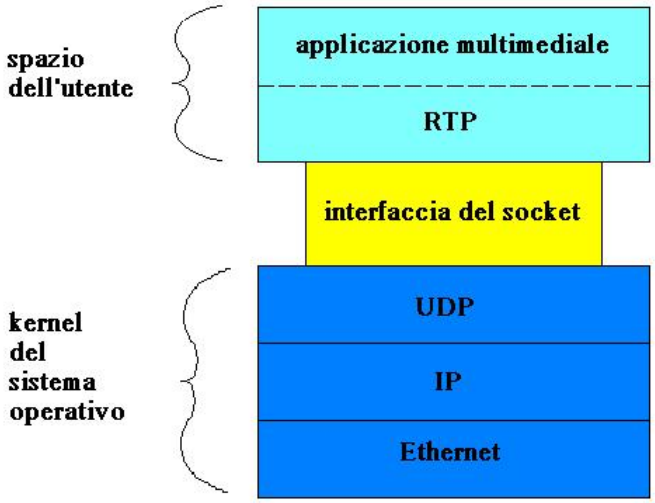


Figura 2.8: Lo stack di RTP

Un'applicazione multimediale è composta di flussi audio, video, di testo e di altro tipo. Questi flussi vengono passati alla libreria RTP che si trova nello spazio dell'utente assieme all'applicazione; la libreria esegue il multiplexing dei flussi e li codifica in pacchetti RTP, che vengono poi inseriti in un socket. All'altra estremità del socket (nel kernel del sistema operativo), vengono generati i pacchetti UDP e incorporati in pacchetti IP. Se il computer si trova su una Ethernet, i pacchetti IP vengono poi inserito in un frame Ethernet per la trasmissione. La pila di protocolli per questa situazione è mostrata in figura 2.8.

[2]é un documento che riporta informazioni o specifiche riguardanti nuove ricerche, innovazioni e metodologie dell'ambito informatico o, più nello specifico, di Internet. Attraverso l'Internet Society gli ingegneri o gli esperti informatici possono pubblicare dei memorandum, sottoforma di RFC, per esporre nuove idee o semplicemente delle informazioni che una volta vagliati dall'IETF possono diventare degli standard Internet.

La funzione di base di RTP è eseguire il multiplexing di flussi di dati in tempo reale in un singolo flusso di pacchetti UDP. Il flusso UDP può essere inviato a un singola destinazione (*unicasting*) o a più destinazioni (*multicasting*). Dal momento che RTP utilizza il normale UDP, i suoi pacchetti non sono trattati in modo speciale dai router, a meno che siano attivate alcune delle normali funzionalità Quality of Service (QoS) IP. In particolare, non vi sono garanzie speciali sulla consegna, su ritardi temporali e così via. A ogni pacchetto inviato in un flusso RTP è assegnato un numero incrementato di una unità rispetto al suo predecessore; la numerazione serve alla destinazione per scoprire se mancano alcuni pacchetti. In mancanza di un pacchetto, l'azione iegliore che la destinazione può compiere è approssimare il valore mancante per interpolazione. La ritrasmissione non è un'opzione praticabile[3], perché il pacchetto ritrasmesso arriverebbe probabilmente troppo tardi per essere utile. Il timestamp riduce gli effetti del jitter e consente la sincronizzazione di più flussi.

RTP è abbinato ad un protocollo di pari livello chiamato RTCP (Real Time Control Protocol). Questo protocollo gestisce il feddback, la sincronizzazione e l'interfaccia utente, ma non trasporta alcun dato.
La prima funzione può essere utilizzata per fornire alla sorgente del flusso una retroazione (*feedback*) su ritardi, jitter e altre proprietà di rete. Queste informazioni possono servire al processo di codifica per aumentare la velocità dei dati (e fornire una qualità superiore) quando la rete funziona bene per ridurre la velocità dei dati quando vi sono problemi nella rete. Fornendo un feedback continuo, gli algoritmi di codifica si possono adattare costantemente per fornire la migliore qualità permessa dalle circostanze.
RTCP gestisce anche la sincronizzazione tra flussi. Il problema è che flussi diversi possono utilizzare clock differenzi: questo protocollo può essere utilizzato per mantenere la sincronia fra i flussi.

Infine, RTCP fronisce un metodo per denominare le sorgenti: queste informazioni possono essere visualizzate sullo schermo del ricevitore per indicare chi sta parlando.

Un'altra caratteristica necessaria per molte applicazioni in tempo reale è il *timestamp*. L'idea sta nel consentire all'origine di associare un contrassegno temporale al primo campione di ogni pacchetto. I timestamp sono relativi all'inizio del flusso, per cui sono significative solo le differenze tra timestamp mentre i valori assoluti non forniscono alcun contributo informativo. Questo meccanismo consente alla destinazione di svolgere una piccola quantità di buffering e di riprodurre ogni campione nell'esatto istante richiesto, indipendentemente dall'istante di arrivo del pacchetto contenente il campione.

Un approfondimento delle intestazioni dei pacchetti di RTP e RTCP è rinviata all' appendice C. Per uno studio approfondito dei due protocolli si veda [HS03] .

[3]questo è fondamentalmente il motivo per cui RTP non è basato su TCP

2.4 Il protocollo ITU-T H.323

L'ITU (International Telecommunication Union) è, dal 1947, un'agenzia delle Nazioni Unite con il compito di standardizzare le telecomunicazioni internazionali; emette i propri standard sotto forma di indicazioni, meglio conosciute come *Raccomandazioni*.

L'ITU è organizzata in settori: il settore che si occupa degli standard per le telecomunicazione, e quindi di interesse per il mondo della telefonia, è il settore T. Le Raccomandazioni sono a loro volta organizzate in serie; la serie ITU-T H è interamente dedicata ai sistemi multimediali e audiovisivi. La Raccomandazione ITU-T H.323 comprende la definizione delle componenti tecniche necessarie per una comunicazione multimediale (audio, video, dati) che si trovi a lavorare, come sottorete di trasporto, su una rete a pacchetto che potrebbe non garantire una Qualità di Servizio (QoS - Quality of Service).

Nei prossimi paragrafi verranno illustrati i componenti principali del sistema H.323 ossia:

- gli elementi H.323
- la suite di protocolli H.323

Verrà proposto anche un esempio di comunicazione, per comprendere il funzionamento di questo protocollo.

Gli elementi H.323

La figura 2.9 illustra gli elementi fondamentali di un sistema H.323. Questi elementi includono terminali, gateway, gatekeeper, e unità Multipoint Control Units (MCU) I terminali sono spesso indicati come endpoint e corrispondono a quei dispositivi che consentono di effettuare conferenze punto-punto e multipunto di tipo audio, video e dati. I gateway consentono di effettuare l'interconnessione con reti telefoniche tradizionali o PSTN e con reti ISDN, allo scopo di definire opportunamente l'interazione fra endpoint H.323. I gatekeeper definiscono le operazioni di controllo e i servizi di risoluzione degli indirizzi impiegati da terminali e gateway. Le unità MCU sono dispositivi che consentono a due o più terminali e gateway di effettuare sessioni di multiconferenza audio e/o video.

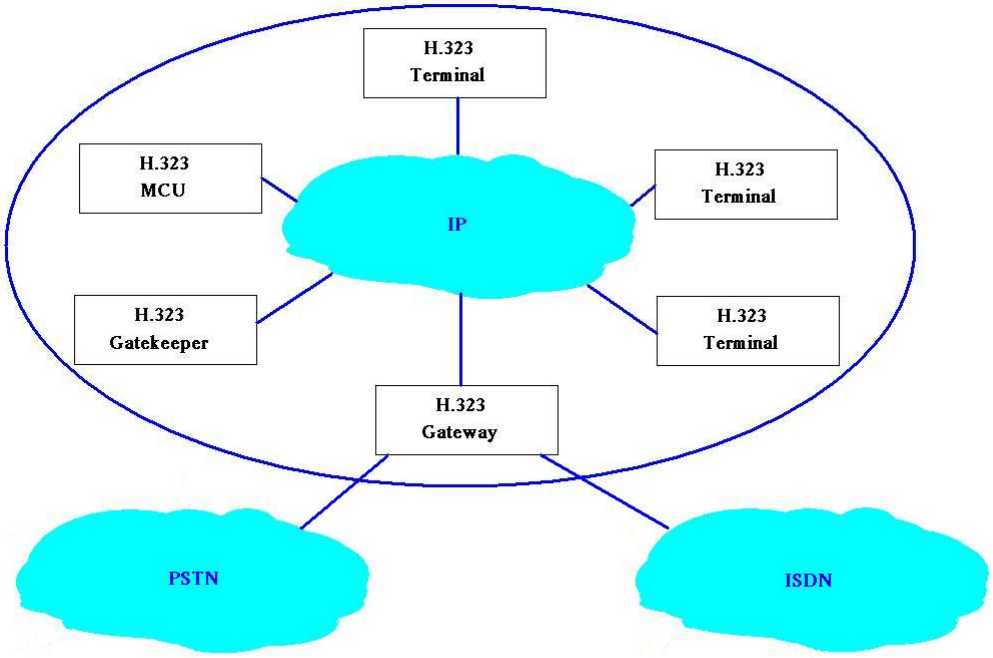


Figura 2.9: elementi di una rete H.323

Terminale o Endpoint

L'elemento di rete mostrato in figura è definito dalle specifiche H.323 come *terminale* di rete. I terminali H.323 devono avere un'unità di controllo del sistema, un'unità di trasmissione multimediale, un codec audio, un'interfaccia per la rete a commutazione di pacchetto. A questi requisiti necessari si affianca la presenza di codec video e di applicazioni relative a dati utente.Di seguito sono indicate le funzionalità e le unità che caratterizzano un terminale H.323:

- Unità di controllo del sistema. Definisce le funzionalità H.225 e H.245 di controllo della chiamata, di scambio delle operazioni, dei messaggi e dei comandi che consentono l'utilizzo stesso del terminale.

- Unità per le trasmissioni multimediali.Si occupa della formattazione relativa alle trasmissioni audio, video,dati, gli stream di controllo e i messaggi sull'interfaccia di rete. Queste unità elaborano, a loro volta i segnali audio, video, dati, gli stream di controllo e i messaggi sull'interfaccia di rete. Queste unità elaborano, a loro volta, i segnali audio, video, dati, gli stream di controllo e i messaggi provenienti dall'interfaccia di rete.

- Codec Audio. Codifica il segnale dell'apparecchiatura audio per la trasmissione in rete e decodifica il segnale audio digitale in ingresso.

- Interfaccia di rete. Interfaccia della rete a pacchetti in grado di utilizzare servizi di trasmissione TCP e UDP, in modalità unicast e multicast.

- Codec video. Se necessario, è possibile definire queste unità per la codifica e decodifica di segnali video.

Gateway

I Gateway sono caratteristici di qualsiasi comunicazione tra sottoreti diverse. La connessione avviene tramite traduzione dei protocolli applicativi relativi alle due reti e talvolta anche attraverso la traduzione dei protocolli di trasferimento e la conversione di codifica dei flussi audio/video. Un esempio è quello di un Gateway che vede sia la rete IP che la PSTN (Public Switched Telephone Network). Il Gateway termina i protocolli di comunicazione su entrambe le reti: esso viene quindi visto come un terminale H.323 sulla LAN e come un terminale PSTN sulla rete commutata. L'aggiunta del Gateway all'architettura ha dato ad H.323 la possibilità di interfacciarsi con tutte le altre realtà esistenti, e di conseguenza ha permesso allo standard di diffondersi in maniera capillare.

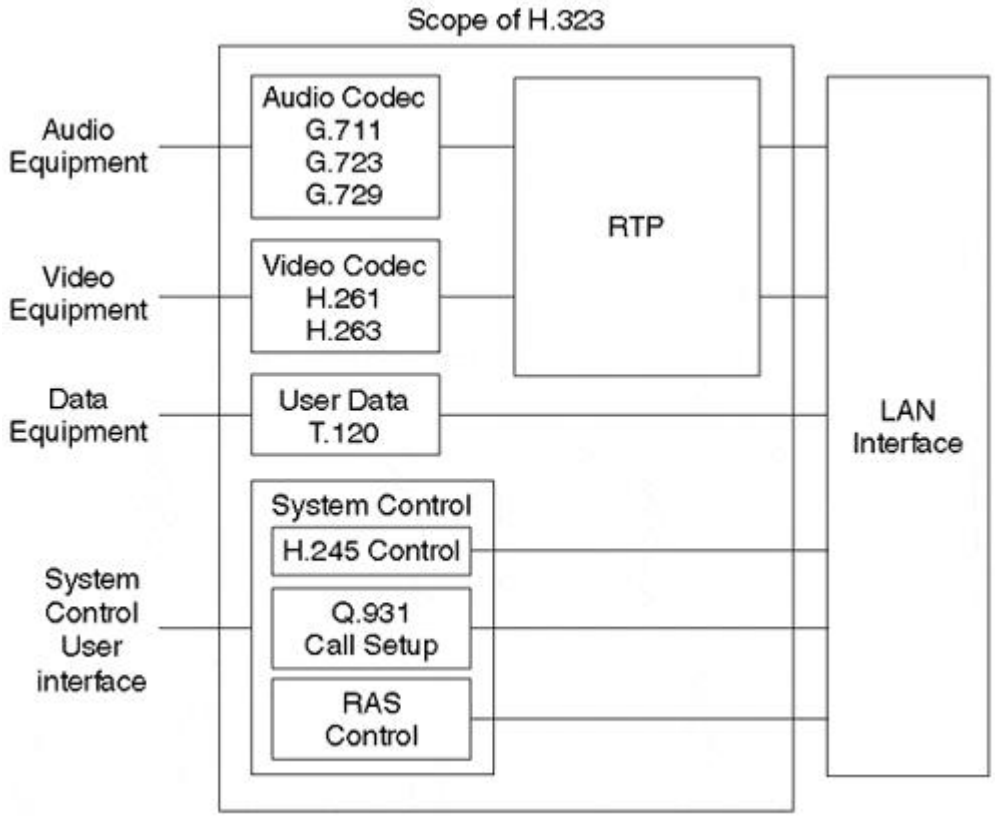


Figura 2.10: terminale H.323

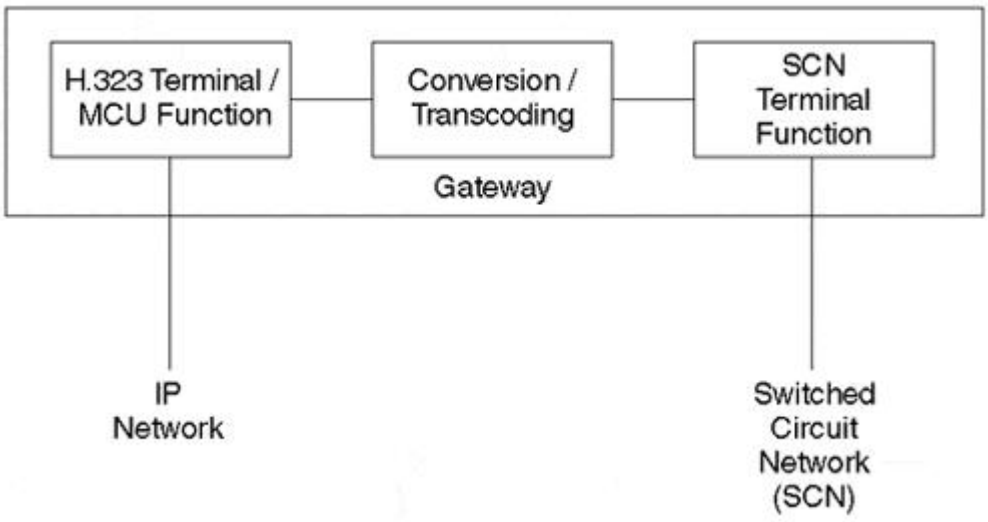


Figura 2.11: Gateway H.323

Gatekeeper

Il Gatekeeper, pur non essendo obbligatoriamente presente in una comunicazione multimediale, svolge, se utilizzato, numerose importanti funzioni e offre tutta una serie di sevizi a cui altrimenti si deve rinunciare. In particolare, esso può limitare l'accesso al servizio, secondo le disposizioni dell'amministratore dello stesso, e la quantità di capacità trasmissiva allocata per le applicazioni multimediali ad un livello tale da non degradare troppo la qualità offerta agli altri servizi (email,trasferimento dati ...). Lo standard definisce in maniera precisa le funzionalità offerte dal Gatekeeper,delle quali alcune obbligatorie e altre opzionali. Tra quelle obbligatorie bisogna citare:

- Address Translation: trasformazione di un numero telefonico (E.164) o di un indirizzo alias in un indirizzo IP e vice versa. Esiste, internamente al Gatekeeper, una tabella delle registrazioni, continuamente aggiornata attraverso i messaggi del protocollo RAS.
- Admission Control: controllo di ammissione di un endpoint in una Zona H.323. Attraverso i messaggi Admission ReQuest (ARQ), Admission ConFirm (ACF) e Admission ReJect (ARJ), ad un endpoint viene permesso o meno l'accesso alla LAN. L'accesso può essere basato su autorizzazioni alla chiamata, banda richiesta o altri criteri.
- Bandwidth Control: controllo di banda. Il Gatekeeper usa i messaggi Bandwidth ReQuest (BRQ), Bandwidth ConFirm (BCF) e Bandwidth ReJect (BRJ) per la gestione della banda disponibile.
- Zone Management: controllo dei componenti H.323 appartenenti alla Zona H.323; il Gatekeeper offre i propri servizi solo a Terminali, Gateway e MCU ad esso registrati.

Tra le funzionalità opzionali del Gatekeeper possiamo elencare:

- Call Control Signalling: in una conferenza punto-punto il Gatekeeper può processare i segnali di controllo H.225.0 oppure farli scambiare direttamente tra gli endpoints.
- Call Authorization: il Gatekeeper può rifiutare una chiamata per motivi ad esempio di mancata autorizzazione. I criteri per determinare queste autorizzazioni sono al di fuori della raccomandazione.
- Bandwidth Management: il Gatekeeper può rifiutare una chiamata senon c' è abbastanza banda disponibile, ma i criteri per definire la banda disponibile vanno al di fuori della raccomandazione. Tale funzione può operare anche dopo la fase di inizializzazione della chiamata, qualora gli utenti decidessero di cambiare il tipo di media che si stanno scambiando nella chiamata in atto.

- Call Management: il Gatekeeper può tenere, ad esempio, una lista degli utenti Occupati in comunicazioni.
- Routing della segnalazione di chiamata: il Gatekeeper si fa carico dell'inoltro agli endpoint della segnalazione di chiamata. Questo permette, ad esempio, a un service provider di monitorare le chiamate sulla rete a scopi di billing, oppure di inoltrare chiamate ad altri endpoint ove quelli chiamati non siano disponibili, ...
- Routing del controllo di chiamata: il canale di controllo H.245, invece di essere stabilito tra i due endpoint, passa attraverso il Gatekeeper (H.245 Routed Mode).

Multipoint Control Unit (MCU)

Uno dei punti di forza della Raccomandazione H.323 è il supporto che fornisce alle applicazioni multimediali multipunto. Il componente fondamentale per questo aspetto è il Multipoint Control Unit. Se da una parte può essere considerato un Terminale, in quanto può generare e terminare flussi audio e video, dall'altra la differenza è come agisce su questi. Infatti, l'MCU riceve tutti i flussi audio e video dai partecipanti alla conferenza e restituisce a tutti un unico flusso contenente i contributi di ognuno. Per quanto riguarda l'audio, le varie comunicazioni vengono sovrapposte come nella realtà, mentre, per quel che riguarda il video, viene spedita un'immagine di formato standard, ma comprendente al suo interno riquadri più piccoli con i partecipanti alla videoconferenza. Al suo interno sono comprese due parti:

- il Multipoint Controller (MC), che gestisce i segnali di controllo (H.245) e stabilisce dei codec audio e video comuni a tutti gli endpoints, come pure una larghezza di banda sostenibile da tutti i partecipanti alla conferenza. Esso non ha a che fare direttamente con i flussi voce/video.
- il Multipoint Processor (MP), che si occupa di processare e mescolare i flussi audio/video.

La suite di protocolli H.323

H.323 è una cosiddetta raccomandazione ombrello, cioè essa racchiude e collega insieme una serie di protocolli applicativi, specificati poi nel dettaglio in altre raccomandazioni: in letteratura questa suite di protocolli è anche conosciuta col nome famiglia H.323. La famiglia H.323 supporta le funzionalità di call admission, setup, teardown e trasporto di stream multimediali e messaggi impiegati all'interno di uno scenario di rete H.323. Questi protocolli sono supportati da opportuni meccanismi di trasporto dei dati all'interno della rete, sia robusti che best-effort.

Anche se al momento la maggior parte delle operazione H.323 utilizzano il protocollo TCP per trasportare le informazioni di segnalazione, il protocollo H.323 versione 2 fornisce anche il trasporto delle informazioni su UDP.

La suite protocollare H.323, come spiegato in [DP01], è contraddistinta da tre aree principali relative a funzioni di controllo:

- RAS Signaling. Definisce il controllo che precede una chiamata all'interno di un'architettura di rete H.323 basata su gatekeeper.
- Call Control Signaling. Meccanismo impiegato per impostare,mantenere e terminare conversazioni tra dispositivi di endpoint.
- Media Control and Transport. Meccanismo che definisce un opportuno canale di rete affidabile H.245 allo scopo di trasmettere i messaggi di controllo relativi ai dati multimediali.Meccanismi di trasporto avvengono mediante il protocollo UDP

La parte conclusive di questa sezione focalizza la propria attenzione su queste tre funzioni.

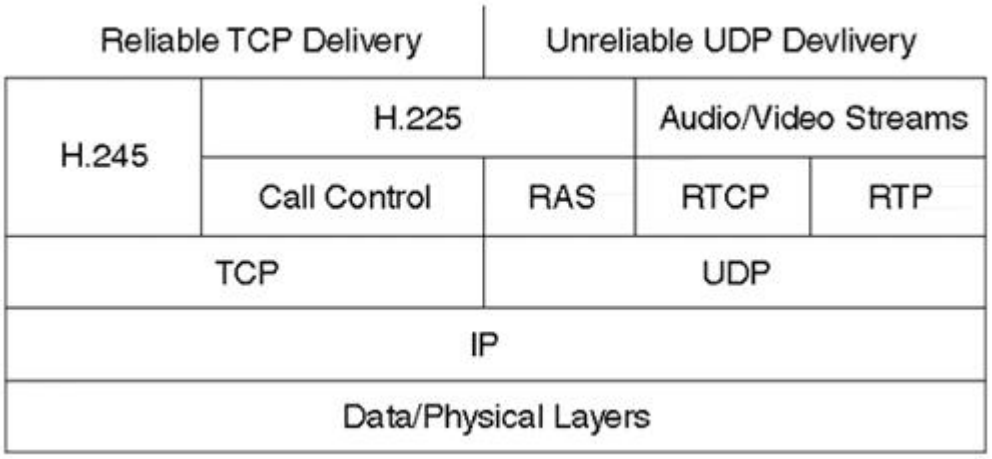


Figura 2.12: Stack protocollare H.323

RAS Signaling

Questo protocollo definisce i messaggi per lo scambio di segnalazione di controllo tra gli endpoints ed il Gatekeeper. I messaggi RAS sono scambiati su un canale inaffidabile, quindi utilizzano il protocollo UDP. Tale canale è aperto indipendentemente da una chiamata e, quindi, è aperto prima dell'instaurazione di qualunque altro canale. Mediante il protocollo RAS si realizzano le procedure di scoperta del Gatekeeper, registrazione, controllo di ammissione, modifica della banda assegnata ad una comunicazione, scambio di informazioni di stato, cancellazione di registrazioni.

Call Control Signaling (H.225)

È il protocollo che definisce i messaggi scambiati al fine di stabilire l'instaurazione e l'abbattimento di una connessione tra endpoints, sulla quale avverrà lo scambio di dati realtime. Lo scambio dei messaggi avviene su un canale affidabile, quindi TCP per le reti basate su IP. Questo protocollo si rifà, sostanzialmente, al protocollo Q.931, definito per il controllo di chiamata in reti ISDN. Infatti, i messaggi H.225.0 utilizzati nell'instaurazione e nell'abbattimento di una chiamata, sono identici a quelli usati per una chiamata ISDN. Nel caso di presenza di un Gatekeeper, possono verificarsi due scenari:

- Gatekeeper Routed Call Signalling: gli endpoints instaurano il canale di Call Signalling con il Gatekeeper, che, quindi, sarà attraversato da tutti i messaggi H.225.0. Tale procedura permette un forte controllo sulla chiamata da parte del Gatekeeper.
- Direct Call Signalling: il Gatekeeper non sarà attraversato dai messaggi H.225.0, che sono scambiati direttamente tra gli endpoints comunicanti.

Quale delle due procedure adottare è deciso, dal Gatekeeper, durante il dialogo RAS. Nel caso di assenza del Gatekeeper, ovviamente, i messaggi di Call Signalling sono scambiati direttamente tra gli endpoints. Tipici messaggi di Call Signalling sono: Setup, CallProceeding, Alerting, Connect, ReleaseComplete.

Media Control and Transport (H.245 and RTP/RTCP)

Tipici messaggi di Call Signalling sono: Setup, CallProceeding, Alerting, Connect, ReleaseComplete. H.323 fa riferimento alla Raccomandazione ITU-T H.245 per il protocollo di controllo delle comunicazioni; con H.245 si realizzano le procedure per attivare e gestire una comunicazione multimediale tra due o più entità H.323. Ogni entità deve instaurare un canale dedicato di tipo affidabile (TCP), detto H.245 Control Channel, per ogni comunicazione alla quale partecipa; tale canale sarà permanentemente aperto durante tutta la connessione. Si osservi che

nel caso una delle entità in questione sia un MCU, che può supportare molte chiamate contemporaneamente, è necessaria l'apertura di più canali H.245 di controllo. L'instaurazione degli H.245 Control Channel avviene dopo il dialogo H.225 (RAS e Call Signalling). Si menzionano le principali procedure descritte nella raccomandazione:

- Master-Slave Determination: procedura per risolvere le contese tra endpoints.
- Capabilities Negotiation: mediante tale procedura, le entità comunicanti negoziano le modalità con le quali si svolgerà la comunicazione; si stabiliscono, ad esempio, le codifiche audio e video, il numero di canali logici sui quali saranno trasmessi i media stream... Ciascun endpoint descrive le proprie capabilities come ricevente, abilitando l'altro endpoint a trasmettere nei limiti delle capabilities dichiarate.
- Opening/Closing of Logical Channels: con questa procedura, si instaurano e abbattono i canali logici sui quali viaggeranno i media stream, ovviamente in funzione delle capabilities negoziate. Ad esempio, all'atto dell'apertura di un nuovo canale logico, il trasmettitore indica al ricevitore il tipo di media che trasmetterà su quel canale, consentendo, in ricezione, di predisporre opportunamente le risorse necessarie, quali buffer o codec.
- Flow Control: procedura con cui il ricevitore indica al trasmettitore di ridurre il bit rate dello stream che sta trasmettendo.
- User Input Indication: trasmette all'altro endpoint le informazioni digitate dall'utente, in forma di stringa alfanumerica o di caratteri in codice DTMF (Dual Tone Multi Frequency) impostati mediante tastiera analogica.

L'instaurazione di un H.245 Control Channel richiede l'apertura di una nuova connessione TCP, con il tradizionale threewayhandshake che ne deriva. Nella seconda versione di H.323, per ridurre il ritardo in fase di instaurazione della connessione del tempo necessario al setup di una nuova connessione TCP da dedicare esclusivamente al dialogo H.245, è stato introdotto il meccanismo dell'H.245 Tunneling: i messaggi H.245 sono incapsulati in messaggi H.225.0. Questa soluzione, inoltre, migliora la scalabilità dei Gateway, dimezzando, di fatto, il numero di connessioni TCP simultaneamente aperte. Questa tecnica è particolarmente conveniente se abbinata ad un'altra, il cosiddetto Fast Start: il dialogo H.245 non fa più seguito al messaggio H.225.0 Connect, bensì la struttura dei messaggi di apertura dei canali logici è inserita in un campo dei primi messaggi di Call Signalling, quali Setup e Alerting. In questo modo, quindi, i canali logici sono predisposti ancor prima che la connessione sia completamente instaurata.

Un esempio di esecuzione di H.323

In [Tan04] viene proposto un semplice esempio di funzionamento di H.323. Per vedere come questi protocolli operano assieme, si consideri il caso di un terminale PC su LAN (con un gatekeeper) che chiama un telefono. remoto.
Innanzitutto il PC deve scoprire il gatekeeper, pertanto tramsmette in broadcast un pacchetto UDP di rilvamento del gatekeeper sulla porta 1718. Quando il gatekeeper risponde, il PC apprende il relativo indirizzo IP.
Ora il PC si registra presso il gatekeeper inviando un messaggio RAS in un pacchetto UDP. Dopo l'accettazione, il PC invia al gatekeeper un messaggio di ammissione RAS richiedendo un pò di banda. Solo dopo che la banda è stata garantita può iniziare l'impostazione della chiamata. La richiesta anticipata della banda consente al gatekeeper di limitare il numero di chiamate, per impedire che la linea in uscita sia sottoscritta in eccesso e quindi fornire la qualità del servizio necessaria. dopo l'allocazione della banda, il PC stabilisce una connessione TCP con il gatekeeper e gli invia un messaggio *SETUP Q.931*. Questo messaggio specifica il numero del telefono da chiamare[4]. Il gatekeeper risponde con un messaggio *CALL PROCEEDING Q.931* per confermare la corretta ricezione della chiamata. Il gatekeeper inoltra quindi il messaggio *SETUP* al gateway. Il gateway, che per metà e un PC e per metà è un commutatore telefonico, esegue una telefonata ordinaria al telefono desiderato. Quando il telefono chiamato squilla, il gateway invia un messaggio *ALERT Q.931* per segnalare questo evento al PC chiamante. Quando la persona risponde al telefono, il gateway invia un messaggio *CONNET Q.931* per segnalare al PC che possiede la connessione. Una volta stabilita la connessione il gatekeeper non si trova più nel ciclo. I pacchetti successivi scavalcano il gatekeeper e e raggiungono direttamente l'indirizzo IP del gateway. A questo punto si dispone di un semplice canale tra due parti:si tratta di una connessione allo strato fisico per lo spostamento dei bit. Ogni lato non conosce nulla del'altro.
Il protocollo H.245 ora è utilizzato per negoziare i parametri della chiamata. Utilizza il canale di controllo H.245, che è sempre aperto. Ogni lato inizia annunciando le sue capacità, per esempio se è in grado di gestire video o le chiamate in conferenza, quali codec supporta e così via. Quando ogni lato conosce ciò che l'altro può gestire, vengono impostati due canali dati unidirezionali, assegnando un codec e altri paramentri ad ognuno.

Una vota completata tuttle ne negoziazioni, può iniziare il flusso di dati che utulizza RTP. É gestito tramite RTCP che svolge un ruolo importante nel controllo delle congestioni. Se è presente il video, RTCP gestisce la sincronizzazione audio/video. I canali sono mostrati in figura 2.13 .

Quando le parti riagganciano, il canale di segnalazione della chiamata Q.931 viene utilizzato per abbattere la connessione. Una volta terminata la chiamata, il PC

[4]nel caso il terminale sia un PC, nel messaggio deve essere presente l'indirizzo IP e il numero di porta

contatta il gatekeeper con un messaggio RAS che chiede di rilasciare la banda assegnata. In alternativa può effettuare un'altra chiamata.

Un'interesssane osservazione effettuata in [Tan04] è che non si è parlato di qualità del servizio, anche se è un aspetto essenziale per il successo di Voice over IP. Il motivo è che la qualità del servizio ricade all'esterno dell'ambito di H.323. Se la rete sottostante è in grado di produrre una connessione stabile e priva di jitter dal PC chiamante al gateway, allora la qualità del servizio della chiamata sarà buono; in caso contrario sarà scarso; la parte telefonica è priva di jitter.

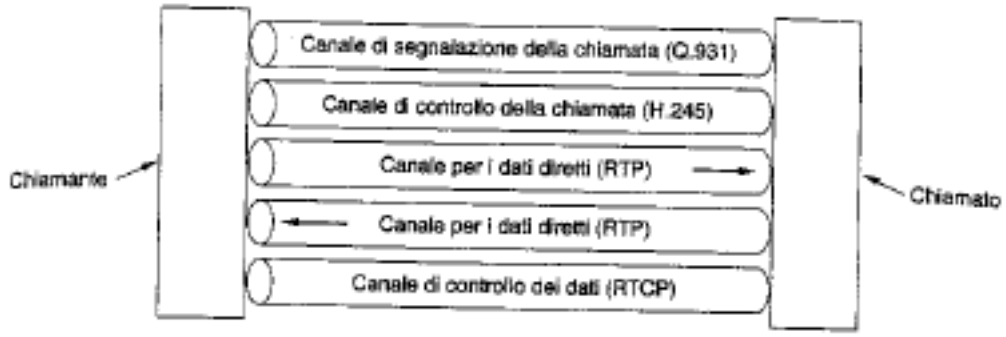


Figura 2.13: i canali logici tra chiamante e chiamato durante una chiamata

2.5 Il protocollo SIP

Il protocollo SIP [Jan03] [DP01] è stato standardizzato per la prima volta nel 1999 dall'IETF (Internet Engineering Task Force) nella RFC 2543, superata e resa obsoleta nel 2002 dalla nuova definizione data dalla RFC 3261. Si tratta di un protocollo di segnalazione e di controllo di livello applicativo utilizzato per stabilire, mantenere e terminare le sessioni [5] multimediali. Le sessioni multimediali includono la telefonia Internet, conferenze e altre applicazioni simili che coinvolgono media come audio, video e dati. Tre sono i protocolli che operano in collaborazione con SIP per rendere possibili numerose applicazioni basate sul concetto di sessione: RTP si occupa del trasporto di contenuti multimediali (audio, video, ecc.) in tempo reale, Session Description Protocol (SDP) ,che fornisce uno strumento di descrizione delle caratteristiche di una sessione, permettendo ai partecipanti di accordarsi su parametri fondamentali quali il tipo di flusso scambiato (es: audio o video), la codifica scelta o il protocollo di trasporto adottato (es: TCP o UDP) e il protocollo Session Announcement Protocol (SAP) , per la divulgazione di comunicazioni in multicast.

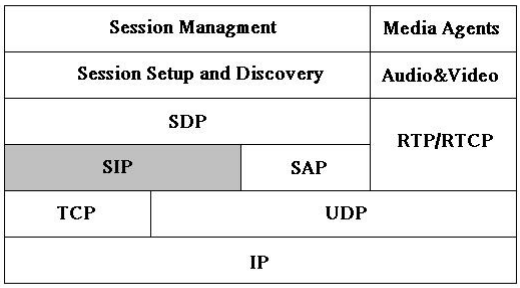


Figura 2.14: Stack protocollare di SIP

Il protocollo, inoltre, si integra bene con altre applicazioni IP quali world wide web e posta elettronica, grazie al principio del riuso adottato in fase di progettazione. SIP si basa infatti apertamente su HTTP, il protocollo più diffuso e popolare del mondo Internet, di cui cerca di sfruttare l'esperienza. Numerosi sono i punti in comune tra i due protocolli, a cominciare dal formato (testo ASCII in chiaro) e dalla sintassi , con i campi header e un corpo del messaggio opzionale. SIP, inoltre, sfrutta per il suo funzionamento tecnologie Internet consolidate come DNS (Domain Name Service), SDP (Session Description Protocol) e URL (Uniform Resource Locator) o URI (Uniform Resource Identifier).

È importante notare che SIP è in grado di operare in congiunzione con altri protocolli di segnalazione, come ad esempio H.323.

[5] con il termine sessione si intende il complesso delle entità comunicanti (mittenti e riceventi) e dello stato da esse mantenuto durante la comunicazione stessa

Elementi di una rete SIP

Nella figura seguente sono illustrati i componenti di una tipica architettura SIP.

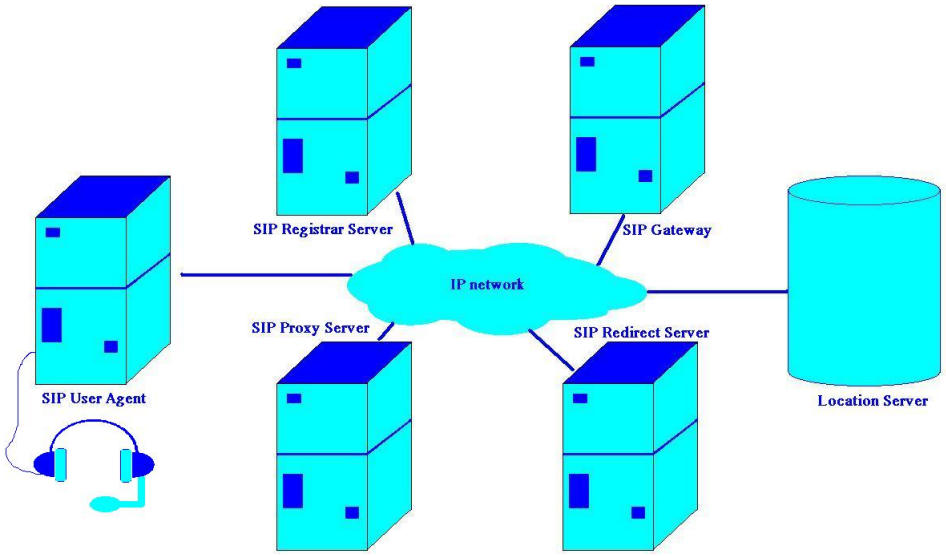


Figura 2.15: Elementi dell'architettura SIP

User Agent

Le entità che partecipano ad una sessione SIP sono note come user agent. Si tratta tipicamente di applicazioni in esecuzione su dei terminali, ma possono fungere da user agent anche dispositivi quali telefoni cellulari, gateway PSTN o PDA. Gli user agent si specializzano, per astrazione funzionale, in User Agent Client (UAC), applicazione client preposta a inviare messaggi di richiesta e ricevere risposte, e User Agent Server (UAS), applicazione server in grado di ricevere richieste e generare le relative risposte.

Uno user agent fungerà da client o da server a seconda dei casi: ad esempio, si comporterà da client al momento di intraprendere una sessione di comunicazione con un altro user agent, assumerà invece il ruolo di server se invitato da un'altra entità ad aprire una sessione di comunicazione. UAC e UAS si possono dunque vedere come due parti logicamente distinte della stessa applicazione.

Terminali

Si tratta di endpoint intelligenti che supportano la comunicazione in tempo reale e bidirezionale con entità analoghe. Tali dispositivi rappresentano l'implementazione fisica degli user agent con le funzionalità UAC e UAS.

Servers

SIP Proxy Server

Svolge il ruolo di intermediario tra terminale chiamante e chiamato nella fase di inizializzazione di una sessione di comunicazione, fungendo al tempo stesso da UAS e da UAC: quando riceve una richiesta di inizio sessione (invitation) da parte di un mittente, la consegna direttamente al destinatario, risolvendone subito il nome, se conosce il terminale su cui si trova, o in caso contrario la inoltra ad un altro proxy server che potrebbe conoscerla. Questa procedura termina quando la richiesta non raggiunge un proxy server in grado di risolvere il nome del destinatario. I proxy server possono essere passivi *Stateless* o attivi *Stateful*.

I server proxy passivi inoltrano i messaggi senza conservare uno stato, ignorando anche un'eventuale organizzazione dei messaggi stessi in transazioni. Essi sono molto semplici e veloci, ma non sono in grado di gestire le ritrasmissioni di messaggi e di realizzare forme avanzate di instradamento quali biforcazioni o attraversamenti ricorsivi.

I server proxy attivi, mantenendo uno stato per ogni transazione, sono più complessi e lenti, ma consentono di realizzare un tipo di instradamento più sofisticato e di gestire le ritrasmissioni, in quanto si tiene traccia di ogni messaggio processato;possono inoltre fornire altre funzionalità avanzate, tra cui un metodo di redirezione automatica di una chiamata in caso di mancata risposta.

SIP ReDirect Server

Questo tipo di server viene contattato dagli utenti in alternativa ad un proxy, ma si comporta in maniera diversa da questo. Anziché preoccuparsi di far arrivare direttamente un messaggio al destinatario, il redirect server ottiene informazioni sulla sua posizione dal location database di un registrar server e le comunica all'utente, redirezionando così la richiesta. Il chiamante potrà quindi inviare nuovamente la richiesta ad uno dei nodi indicati dal redirect server, che può essere un proxy server o l'utente chiamato. Dunque, il Re-Direct Server, a differenza del Proxy che deve occuparsi del routing delle richieste, permette una maggiore scalabilità della rete.

Location Server

Proxy e redirect server accedono a questi server tramite protocolli non SIP (es: Structured Query Language (SQL)) per ottenere informazioni sull'attuale posizione dell'utente chiamato. Il location server risiede tipicamente sulla stessa macchina di un registrar server e contiene un database aggiornato costantemente dalle registrazioni degli utenti. Esso può localizzare direttamente l'utente cercato o restituire gli indirizzi di proxy, gateway o altri location server che potrebbero conoscerne la posizione.

Registrar Server

Un registrar server consente agli utenti SIP di effettuare una registrazione, memorizzando in un location database la loro attuale posizione (data da un indirizzo IP e un numero di porta associati al nome utente). I proxy server sfruttano questo servizio di registrazione per localizzare gli utenti all'interno della rete. Ecco perché i registrar server rappresentano delle entità logicamente indipendenti ma risiedono molto spesso sulle stesse macchine dei proxy.

Indirizzi SIP

Gli indirizzi SIP, detti anche SIP URL (Uniform Resource Locator) o URI (Uniform Resource Identifier), si presentano nella forma sip: user@host, dove user può essere un nome utente o un numero telefonico e host rappresenta un dominio, a cui si dovrà associare (es. tramite un proxy o un redirect server) un indirizzo IP che individui l'attuale posizione dell'utente sulla rete. L'indirizzo SIP di Bob potrebbe essere ad esempio sip: bob@domain.com. Ma bob@domain.com potrebbe essere anche il suo indirizzo e-mail. Risulta allora evidente come un utente possa utilizzare per un'applicazione SIP (es. VoIP) lo stesso indirizzo usato per la posta elettronica. Inoltre, come già avviene per l'e-mail, anche per VoIP (o per altre applicazioni SIP) si può pensare di inserire un collegamento ipertestuale in una pagina web in modo da poter essere facilmente contattati da chi lo desideri.

Richieste

Ecco come si presenta un tipico messaggio di richiesta SIP:

INVITE sip:bob@domain.com SIP/2.0
Via: SIP/2.0/UDP 167.180.112.24:5060
From: sip:alice@wonderland.com
To: sip:bob@domain.com
Call-ID: a2e3a@wonderland.com
CSeq: 34 INVITE
Content-Type: application/sdp
Content-Length: 885

c=IN IP4 167.180.112.24
m=audio 38060 RTP/AVP 0

La start line indica che il messaggio è un invite, usato per avviare una sessione di comunicazione. Le altre informazioni in essa contenute riguardano il Request-URI, cioè l'indirizzo SIP dell'utente da contattare (bob@domain.com) e la versione del protocollo usata dal terminale che ha generato il messaggio (SIP/2.0). Il campo Via, che in questo caso è relativo al terminale chiamante, può essere più di uno e serve a tenere traccia del percorso compiuto dalla richiesta. From e To identificano il chiamante e il chiamato. Call-ID è, come sarà più chiaro in seguito, un identificativo di dialog: permette di individuare i messaggi appartenenti alla stessa chiamata. Content-Type definisce il contenuto del messaggio SIP (codificato tipicamente in SDP) e Content-Length definisce la dimensione in byte di tale carico utile. Cseq, definendo un ordine tra i messaggi, favorisce la gestione delle ritrasmissioni. Un altro campo, assente nell'esempio ma spesso usato, è Contact, con cui il chiamante indica al chiamato indirizzo IP e numero di porto su cui desidera essere contattato in futuro. Dopo la linea bianca di separazione troviamo il message-body codificato in SDP, con il quale, in questo caso, Alice dà indicazioni riguardo al suo indirizzo IP e al formato audio preferito in ricezione. Altri parametri analoghi, relativi in particolare ai flussi multimediali scambiati e al protocollo di trasporto usato, possono essere negoziati nel corpo del messaggio dai partecipanti alla sessione, tramite il protocollo SDP. Come visto, ciò che caratterizza una richiesta è la riga iniziale dell'intestazione, nota anche, in questo caso, come request line, che identifica un metodo di richiesta.

I metodi di richiesta fondamentali, definiti nella RFC 3261, sono:

- INVITE
- ACK
- BYE
- CANCEL
- REGISTER
- OPTIONS

Esaminiamone ora la semantica.

INVITE

Consente di invitare un utente a partecipare ad una sessione di comunicazione. Il chiamante può specificare nel corpo del messaggio, codificato in SDP, il tipo di dati che è in grado di ricevere (e.g. audio o video), le proprie preferenze riguardo al formato dei dati stessi, l'indirizzo IP su cui desidera riceverli e altri parametri della sessione.

ACK

Il chiamante invia un ACK per confermare al chiamato la ricezione di una risposta definitiva ad un precedente INVITE. Se tale risposta era positiva, si instaura così una sessione di comunicazione, grazie ad una procedura di handshake a tre vie. Il corpo dell'ACK può contenere una descrizione della sessione nella usuale codifica SDP. In caso contrario, il chiamato deve considerare ancora validi i parametri di sessione presenti nell'INVITE ricevuto precedentemente.

BYE

Questo messaggio può essere inviato indifferentemente dal chiamante o dal chiamato per terminare una sessione di comunicazione.

CANCEL

Con questo metodo un client può annullare una precedente richiesta per la quale non ha ancora ricevuto una risposta definitiva. Ciò avviene tipicamente in seguito all'invio di un INVITE che non riceve risposta per un certo tempo. Un proxy deve inoltrare una richiesta di CANCEL a tutte le destinazioni con richieste pendenti, mentre un redirect server si limita a rispondere con un codice di stato: 200 (OK) se l'operazione è ancora in corso, 481 (Transaction Does Not Exist) altrimenti.

REGISTER

Un client può ricorrere a questo metodo per registrare presso un registrar server l'utente specificato tramite l'indirizzo SIP presente nel campo To dell'intestazione. Tale utente può essere lo stesso che ha chiesto la registrazione o un altro: in quest'ultimo caso si parla di third-party registration. Il registrar server estrae dal messaggio le informazioni (indirizzo IP e numero di porto) che definiscono la posizione attuale dell'utente nella rete e le memorizza in un location database, a cui accedono proxy e redirect server per localizzare gli utenti. Ogni registrazione ha una scadenza, che viene definita dal server nel campo Expires della risposta o richiesta dal client nello stesso campo della REGISTER, e può essere aggiornata dall'utente che l'ha effettuata. Un UAC può anche annullare esplicitamente una registrazione prima della scadenza stabilita, semplicemente inviando una REGISTER con uno 0 nel campo Expires; per deallocare contemporaneamente tutti gli indirizzi SIP associati all'utente, basta inserire un asterisco nel campo Contact del messaggio.

OPTIONS

Un client si può servire di questo metodo per ottenere informazioni relative allo stato di un utente e alle funzionalità (e.g. metodi) da esso supportate. In particolare, un chiamante può verificare in questo modo la disponibilità dell'utente che ha intenzione di contattare ad accettare un INVITE.

Le primitive di richiesta fondamentali del protocollo SIP, chesono state qui brevemente presentate, sono state estese con nuovi metodi da RFC successive alla 2543 (prima definizione del protocollo), o alla 3261 (definizione aggiornata che ha sostituito la precedente).

Risposte

Ogni richiesta SIP deve ricevere una risposta, fatta eccezione per le richieste di tipo ACK, che rappresentano le conferme nell'handshake a tre vie. Ecco un tipico esempio di risposta SIP:

SIP/2.0 200 OK
Via: SIP/2.0/UDP 167.180.112.24:5060
From: sip:alice@wonderland.com
To: sip:bob@domain.com
Call-ID: a2e3a@wonderland.com
CSeq: 34 INVITE
Content-Type: application/sdp
Content-Length: 0

Come si vede, il messaggio è molto simile ad una richiesta se non per la start line, che presenta, oltre alla versione del protocollo, un codice di stato e una sua descrizione testuale. Si noti come i campi Via, From, To, Call-ID e CSeq abbiano lo stesso contenuto che avevano nella richiesta (riportata precedentemente): From e To non sono invertiti, come si potrebbe essere indotti a pensare. Ciò consente ai server SIP di mettere la risposta in relazione con la richiesta, mentre la funzione di distinguere i due messaggi è affidata alla prima riga dell'intestazione. I codici di stato, mutuati dall'HTTP, sono degli interi compresi tra 100 e 699 e identificano le risposte inviate da un server a un client. Essi si dividono in sei categorie, a seconda del contenuto informativo:

- INFORMATIONAL (1XX)
- SUCCESSFUL (2XX)
- REDIRECTION (3XX)
- REQUEST FAILURE (4XX)
- SERVER FAILURE (5XX)
- GLOBAL FAILURE (6XX)

Vediamo ora la semantica delle diverse classi di risposte SIP.

INFORMATIONAL (1XX)

Si tratta di risposte provvisorie inviate ad un client per informarlo che una richiesta è stata ricevuta ma la relativa elaborazione non è ancora terminata e quindi non è disponibile al momento una risposta definitiva. In seguito alla ricezione di una riposta provvisoria, che viene generata oltre un tempo di attesa di 200 ms, il client è tenuto ad astenersi da ritrasmissioni della richiesta e ad attendere ulteriori informazioni sull'esito dell'elaborazione.

SUCCESSFUL (2XX)

La richiesta è stata accolta con successo. La risposta definitiva inviata dal server è 200 OK.

REDIRECTION (3XX)

Questa classe di risposte fornisce informazioni riguardanti la nuova posizione dell'utente o i servizi alternativi che potrebbero portare a buon fine la chiamata.

REQUEST FAILURE (4XX)

Si tratta di precise risposte di fallimento fornite da un determinato server. Il client non dovrebbe riformulare la richiesta senza modifiche (e.g. aggiungendo le autorizzazioni del caso). Tuttavia la stessa richiesta potrebbe avere successo se inviata ad un server diverso.

SERVER FAILURE (5XX)

Il fallimento della richiesta è dovuto ad un errore del server.

GLOBAL FAILURE (6XX)

Il server ha informazioni definitive riguardo all'utente desiderato ma non quelle relative alla particolare richiesta effettuata.

Transazioni SIP

I messaggi SIP, pur viaggiando sulla rete ognuno in maniera indipendente, possono essere accorpati in transazioni dagli user agent e da alcuni tipi di server proxy. A tal proposito, si parla spesso di SIP come di un protocollo transazionale. Una transazione è una sequenza di messaggi scambiati tra entità di una rete SIP e si compone tipicamente di una richiesta e di tutte le risposte (provvisorie e definitive) a tale richiesta. Quando la richiesta è un INVITE, la medesima transazione comprende anche l'ACK del chiamante solo in caso di risposta negativa. Se la risposta definitiva è positiva (200 OK), l'ACK non fa invece parte della transazione: questo a sottolineare l'importanza della consegna di tutte le risposte positive provenienti dalle entità che hanno ricevuto l'INVITE (in caso di forking della richiesta). Due tipiche transazioni SIP sono illustrate dal diagramma di sequenza in figura. Le entità SIP (e.g. server proxy) in grado di riconoscere i messaggi appartenenti ad una stessa transazione sono dette stateful. Esse mantengono memoria delle transazioni in corso e cercano di associare ogni messaggio SIP ricevuto ad una transazione grazie ad un apposito campo identificativo presente nell'intestazione.

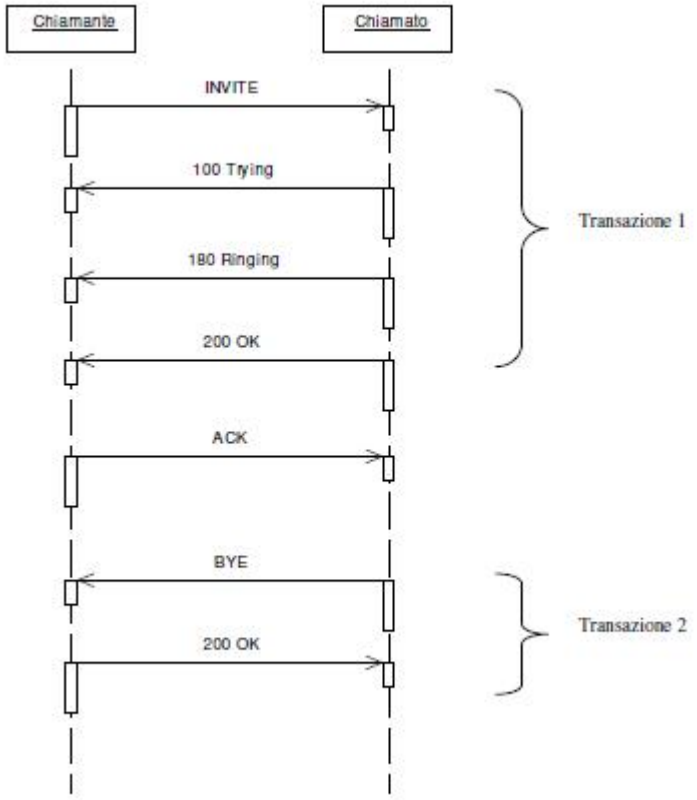


Figura 2.16: esempio di transizione SIP

Dialoghi SIP

Un dialogo è una relazione instaurata tra due user agent in esecuzione su terminali SIP. Essa è identificata dai campi di intestazione Call-ID, From e To. Call-ID è un identificativo di chiamata. Una chiamata può comprendere più dialoghi, nel caso in cui la richiesta che l'ha generata abbia subito un forking, cioè sia stata diramata a diversi terminali: ogni user agent che dà una risposta positiva alla richiesta stabilisce un dialogo distinto col chiamante. I campi From e To identificano un dialogo nel contesto del chiamante e del chiamato rispettivamente. Il campo CSeq identifica invece una transazione all'interno di un dialogo. Ne consegue che un dialogo può anche essere visto come una sequenza di transazioni, benché alcuni messaggi appartenenti ad un dialogo possano non far parte di alcuna transazione (è il caso dell'ACK nel dialogo di figura).Solo alcuni messaggi SIP, come ad esempio l'INVITE, permettono di stabilire un dialogo.

I dialoghi rivestono un ruolo fondamentale nell'instradamento dei messaggi SIP. Quando un utente genera una richiesta (e.g. INVITE) allo scopo di instaurare un dialogo, essa viene presa in consegna da un server proxy che, direttamente o tramite altri server proxy, provvede a recapitarla al destinatario.

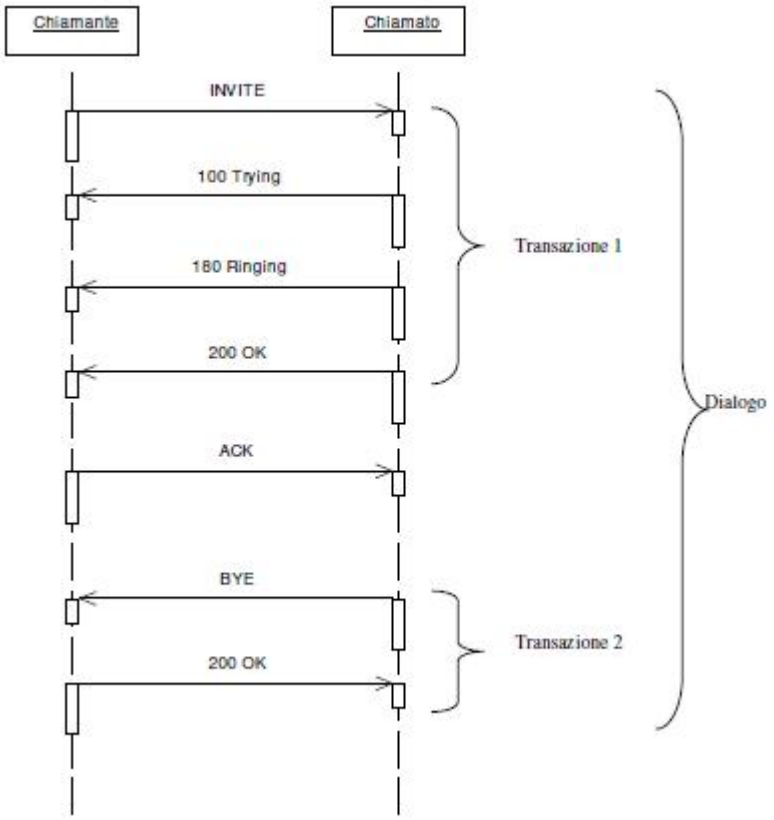


Figura 2.17: esempio di dialogo SIP

Questi risponderà, indicando nel campo di intestazione Contact la sua attuale posizione nella rete (indirizzo IP e porto). Poiché anche la richiesta conteneva un analogo campo Contact, ora chiamante e chiamato conoscono l'uno la posizione dell'altro e potranno pertanto comunicare direttamente e senza l'intermediazione di alcun server proxy per il resto del dialogo. Questo semplifica notevolmente l'instradamento e rende molto più veloce la consegna dei messaggi. A proposito di tale schema di comunicazione, illustrato graficamente di seguito, si parla talvolta di trapezoide SIP.

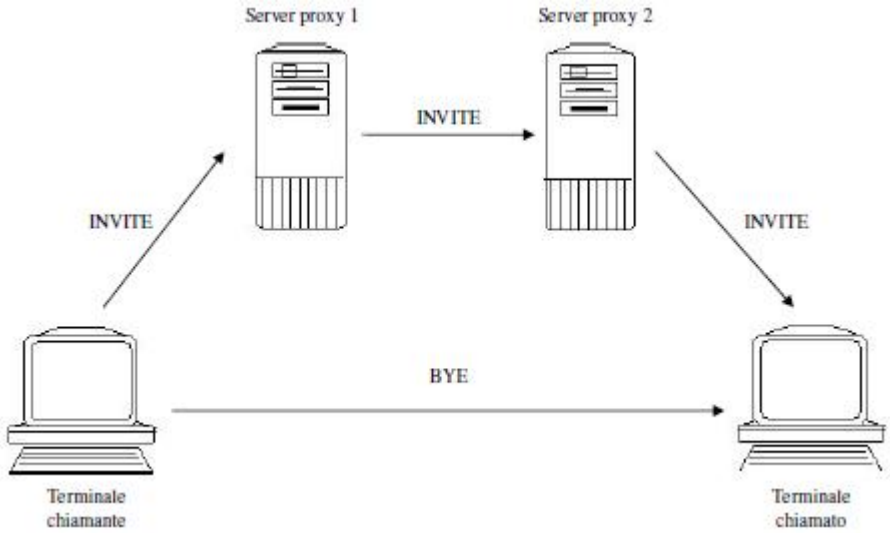


Figura 2.18: trapezoide SIP

Scenari tipici

Con l'ausilio di diagrammi di sequenza viene qui illustrata la dinamica di alcune comuni interazioni tra entità SIP.

Registrazione

Un utente deve registrarsi presso un registrar server per rendersi reperibile dai server proxy e poter essere così contattato dagli altri utenti SIP. La registrazione avviene grazie ad un messaggio di richiesta REGISTER a cui il server risponde con un codice di stato 200 OK solo se il client, dalle credenziali fornite (username e password), risulta autorizzato; se tali credenziali non vengono presentate o non sono valide la risposta sarà invece una 407 (Proxy Authentication Required). La figura seguente illustra una procedura di registrazione, ignorando per semplicità il ruolo del location server che, come visto, forma il più delle volte un tutt'uno col registrar server, da cui si distingue solo a livello logico. Tale scelta è giustificata peraltro dal fatto che l'accesso ai location server avviene tramite protocolli non SIP come SQL e quindi una sua descrizione esula dagli scopi della presente trattazione.

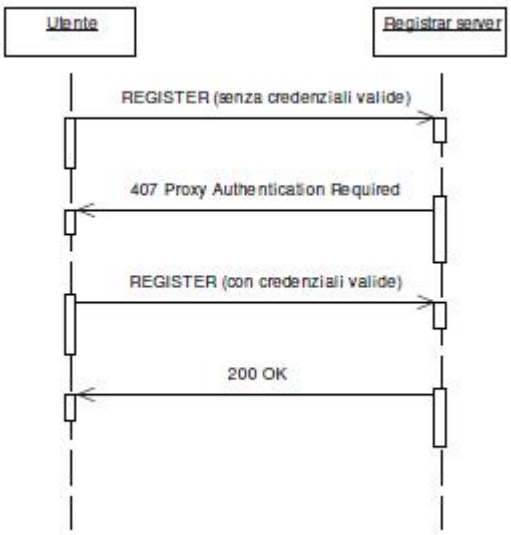


Figura 2.19: registrazione SIP

Instaurazione di una chiamata

Una chiamata o, più in generale, una sessione di comunicazione SIP, può essere stabilita inviando ad un server proxy una richiesta di INVITE con l'indirizzo dell'utente che si intende contattare. Il proxy provvederà, direttamente o tramite altri server proxy, a localizzare l'utente desiderato sulla rete e, in caso di successo, ad inoltrargli l'INVITE. Le risposte (provvisorie e definitive) del chiamato raggiungeranno il chiamante seguendo a ritroso lo stesso percorso della richiesta. I messaggi e i flussi multimediali successivi saranno scambiati direttamente dai due utenti. In alternativa il mittente può indirizzare la propria richiesta ad un redirect server allo scopo di ottenere informazioni sull'attuale posizione del destinatario. Sarà poi il chiamante stesso, eventualmente con l'aiuto di un server proxy, a contattare l'utente. Ecco uno scenario in cui una sessione di comunicazione viene stabilita con successo tramite un server proxy.

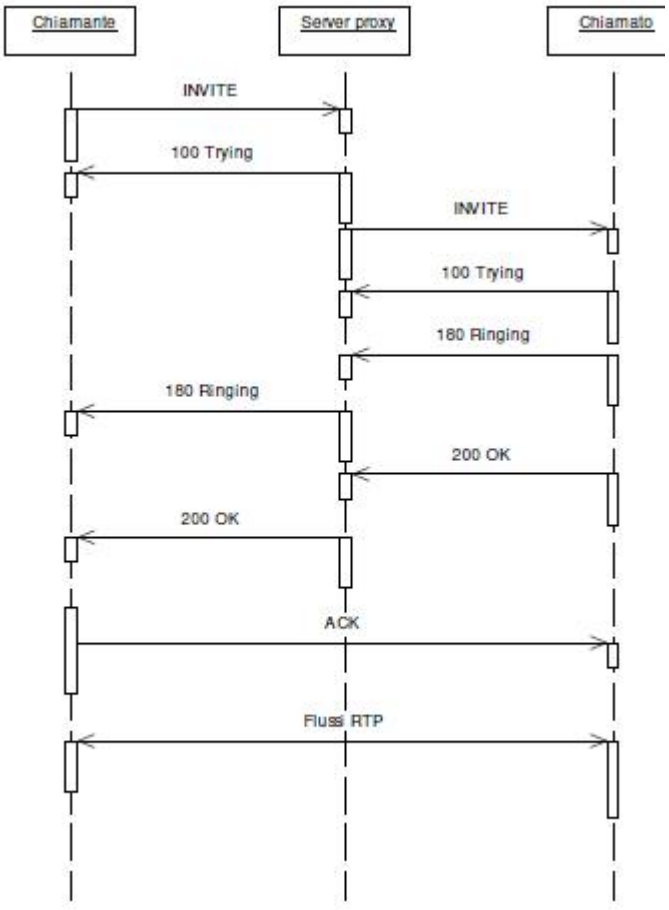


Figura 2.20: instautazione di chiamata tramite server proxy

Nell'esempio seguente il redirect server informa il chiamante che l'utente desiderato non è al momento raggiungibile all'indirizzo specificato nella richiesta,

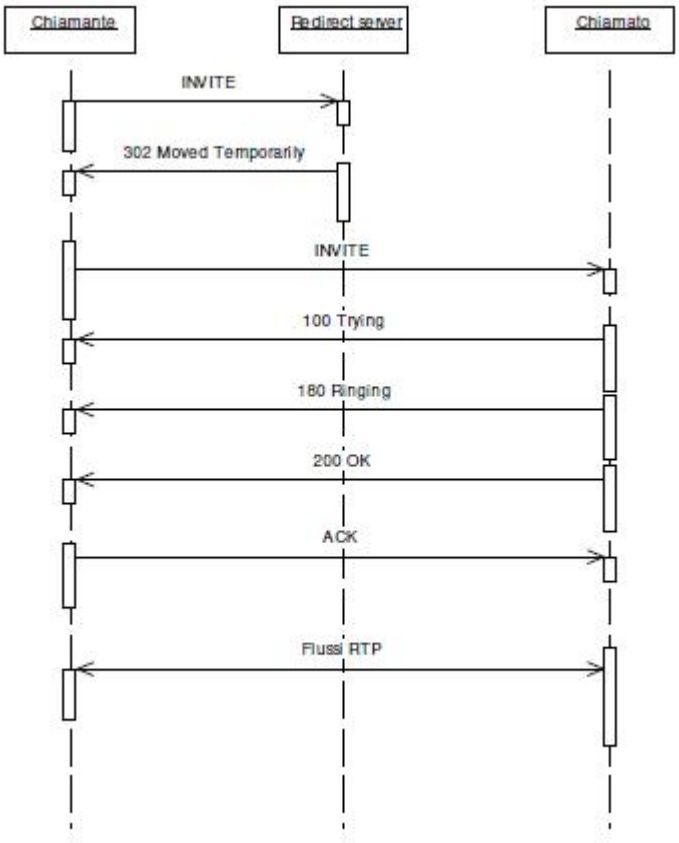


Figura 2.21: instaurazione chiamata SIP tramite redirect server

fornendogli al tempo stesso, nella risposta 302 (Moved Temporarily), l'indirizzo su cui va contattato.

I due scenari di instaurazione di chiamata contengono, come si vede, delle ipotesi semplificative: nel primo assumiamo che il proxy contattato sia in grado di localizzare direttamente il chiamato e di inoltrargli la richiesta; nel secondo supponiamo che il chiamante ottenga dal redirect server informazioni immediate sulla posizione del chiamato senza dover ricorrere ad alcun proxy per fargli pervenire l'INVITE. In entrambi i diagrammi non viene volutamente considerato, per quanto detto in precedenza, il ruolo del location server, a cui redirect e proxy server accedono per ottenere informazioni sull'utente cercato.

Terminazione di una chiamata

Una sessione di comunicazione in corso può essere chiusa da uno qualunque dei due utenti (chiamante o chiamato) tramite un messaggio di richiesta BYE a cui segue tipicamente una risposta positiva (200 OK).

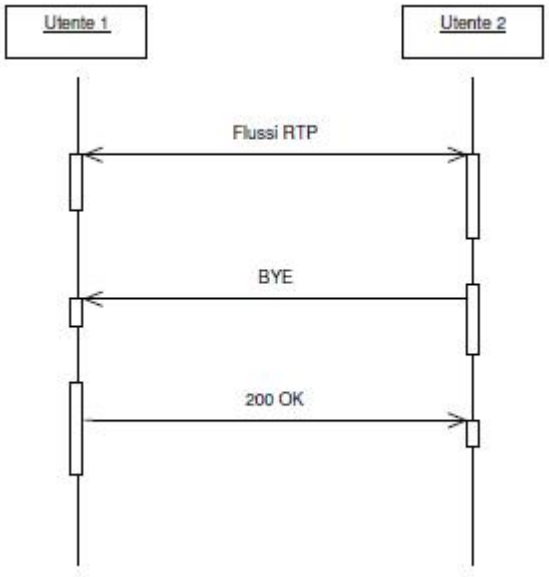


Figura 2.22: terminazione di una chiamata SIP

Capitolo 3

Il protocollo di Skype

Il nome Skype, oltre ad identificare il celebre software proprietario freeware di messaggistica istantanea e VoIP, viene comunemente utilizzato per identificare il protocollo proprietario[1] su cui si basa.

Come è stato già affermato in precedenza, questo protocollo forma un'overlay network non strutturata basata sul meccanismo dei superpeer. Proprio per questo motivo, la rete di Skype è costituita essenzialmente da due tipi di nodi:

- host ordinari (*ordinary host*)
- super nodi (*Super Node (SN)*)

Un host ordinario è un'applicazione Skype che può essere utilizzata per effettuare chiamate e inviare messaggi di testo, mentre un supernodo costituisce l'end point, cioè il nodo con cui gli host ordinari comunicano direttamente. Per quanto concerne il problema della selezione dei supernodi, è da sottolineare il fatto che qualsiasi nodo ordinario avente indirizzo IP pubblico, sufficiente memoria e banda è candidato a divenire supenodo [BS04].

Un host ordinario deve connettersi ad un supernodo e registrarsi sullo Skype login Server per poter accedere ai servizi della rete Skype. Nonostante non sia un nodo di Skype in senso stretto, il login server di Skype è un'entità importante: viene impiegato sia come repository per username e password degli utenti, sia per effettuare l'autenticazione dell'utente all'atto dell'accesso alla rete garantendo l'univocità del *login name* all'interno di tutta la rete di Skype. La figura 3 illustra la relazione tra gli host, i supernodi e il login server.

A parte quest'ultimo, non esiste alcun altro server centrale all'interno della rete Skype. Le informazioni sugli utenti sono mantenute e propagate in modo decentralizzato, così come tutte le operazioni legate alle funzioni di ricerca.
Skype utilizza il protocollo TCP per la sincronizzazione, mentre sfrutta sia TCP che UDP per il trasporto: queste due attività non avvengono sulla stessa porta.

[1] il protocollo Skype non è formalizzato in alcuno standard internazionale

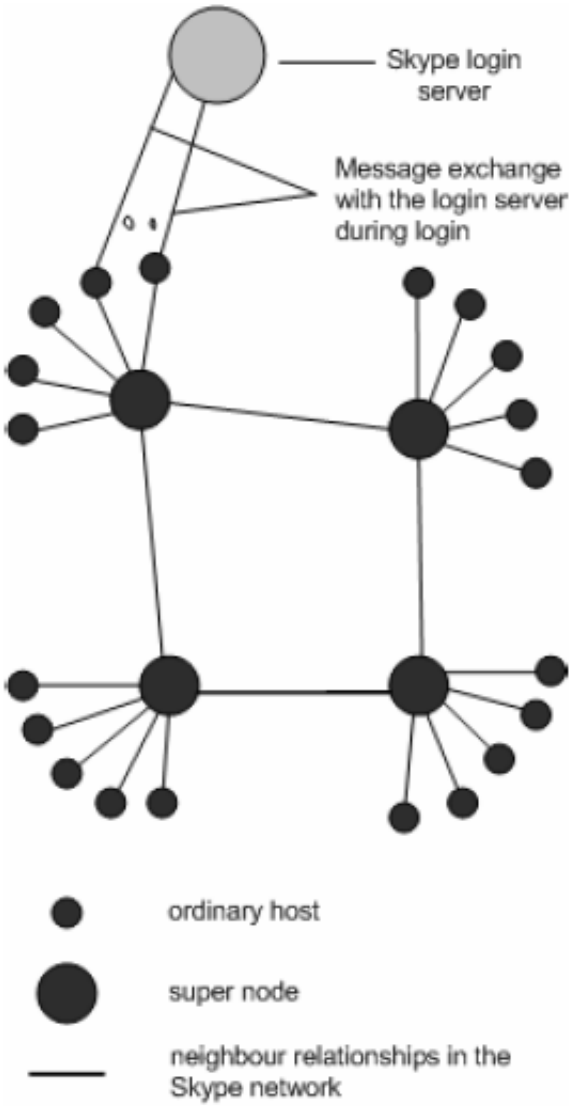


Figura 3.1: **Rete di Skype**. Grafo della rete di Skype

Dato che la rete Skype è un'overlay network, ogni client deve costruire e aggiornare continuamente una tabella contenente tutti gli altri host raggiungibili chiamata *Host Cache (HC)*, che contiene gli indirizzi IP e i numeri di porta dei supernodi. Negli studi sul protocollo di Skype effettuato in [BS04] e [BS06b] si afferma che nelle versioni del client per Microsoft Windows, ciascun nodo conserva la propria Host Cache all'interno del registro di sistema.

3.1 Componenti chiave di Skype

Uno Skype Client (SC):

- si pone in ascolto su porte particolari per le chiamate in ingresso
- mantiene una tabella degli altri nodi, chiamata HC
- utilizza codec a banda larga
- mantiene una buddy list
- cifra i messaggi end - to - end
- determina se ha di fronte un Network Address Translation (NAT) o un Firewall

Le porte

Uno SC apre una porta in ascolto TCP e UDP in base alla configurazione specificata: inizialmente tali porte vengono scelte in maniera casuale durante l'installazione. In aggiunta, esso apre anche le porte TCP 80 (porta HTTP), e 443 (porta HTTPS) in ascolto. Diversamente da molti protocolli di Internet come SIP e HTTP, non viene specificata alcuna porta TCP o UDP di default.

Host Cache

Come già affermato in precedenza, l'Host Cache è una tabella contenente coppie ⟨ indirizzo IP - numero porta ⟩ dei supernodi che ogni Skype Client costruisce e aggiorna regolarmente.
L'operazione di aggiornamento è da considerare fra le più critiche fra le operazioni svolte da Skype, perché il corretto funzionamento del client dipende soprattutto dalla affidabilità delle entries presenti nella HC. Essa, infatti, deve contenere almeno una entry valida, ovvero un elemento costituito dall'indirizzo IP e dal numero di porta di un supernodo Skype online.

Codec

Nelle versioni precedenti alla 4, si crede che Skype utilizzasse iLBC, iSAC o un terzo codec non indicato esplicitamente. Questa affermazione è dovuta al fatto che GlobalIPSound, l'azienda che ha implementato questi codec, annovera Skype fra i proprio partner. Dalla versione 4 viene utilizzato SILK (Super WideBand Audio Codec), un codec audio di proprietà di Skype rilasciato sotto licenza royalty free (di libero utilizzo gratuito). Secondo quanto comunicato da Skype, il nuovo

codec dovrebbe essere abbastanza elastico da sopportare bruschi rallentamenti di banda senza degradare troppo la qualità della chiamata, e riuscire a bilanciare il volume della voce con il rumore di fondo in modo da rendere meno fastidiosa la presenza di altri suoni durante la chiamata.

Buddy List

Skype memorizza le informazioni sugli amici nel registro di Windows(per la versione per Windows). La lista degli amici è firmata digitalmente e cifrata. La lista dei contatti è locale su una macchina, e non è memorizzata su un server centrale.

Encryption

Il sito di Skype spiega: "Skype usa Advanced Encryption Standard (AES) anche conosciuto come Rijndel , il quale è anche utilizzato dal governo americano per proteggere le informazioni sensibili." Skype adotta una cifratura a 256 bit,il quale può fornire un totale di 1.1 X 10^7 chiavi possibili, al fine di cifrare i dati di ogni chiamata e i messaggi istantanei.

NAT e Firewall

Nei lavori di Baset e Schulzrinne [BS04] e Bian [Bia06], si suppone che Skype utilizzi una variante dei protocolli Simple Traversal of UDP through Nat (STUN) e Traversal Using Relay NAT (TURN) per determinare il tipo di NAT che ha di fronte. Si suppone inoltre che queste informazioni vengano aggiornate periodicamente. In un articolo di Jürgen Schmidt [Sch], si dichiara che Skype adotta una tecnica di NAT Traversal conosciuta sotto il nome *UDP Hole Punch*: per essere in grado di scambiare i pacchetti con le controparti il più direttamente possibile, un SC utilizza questo trucco per creare dei "buchi" nei firewall (i quali non dovrebbero consentire l'ingresso di pacchetti dal mondo esterno).

Sempre in [BS04] è stato verificato sperimentalmente che il protocollo di Skype sembra preferire, ove possibile l'utilizzo del protocollo UDP per il traffico vocale, anche nelle situazioni in cui il client si trovi dietro ad un NAT che non blocca il transito di pacchetti UDP.

3.2 Le Funzioni di Skype

Le funzioni di Skype possono essere classificate in :

- avvio (Startup)
- login
- ricerca degli utenti (User search)
- call establishment e tear down
- media transfer
- messaggi di presenza
- conferenza

Per effettuare lo studio del comportamento di Skype, Baset e Schulzrinne [BS04] hanno elaborato esperimenti sotto 3 differenti configurazioni di rete, utilizzando due SC installati su macchine diverse. Nella prima configurazione, entrambi gli utenti Skype hanno un indirizzo IP pubblico. Nella seconda configurazione, uno dei due SC è posto dietro ad un NAT. Nel terzo caso, entrambi sono posti dietro ad un NAT - firewall, con restrizione sull'UDP.

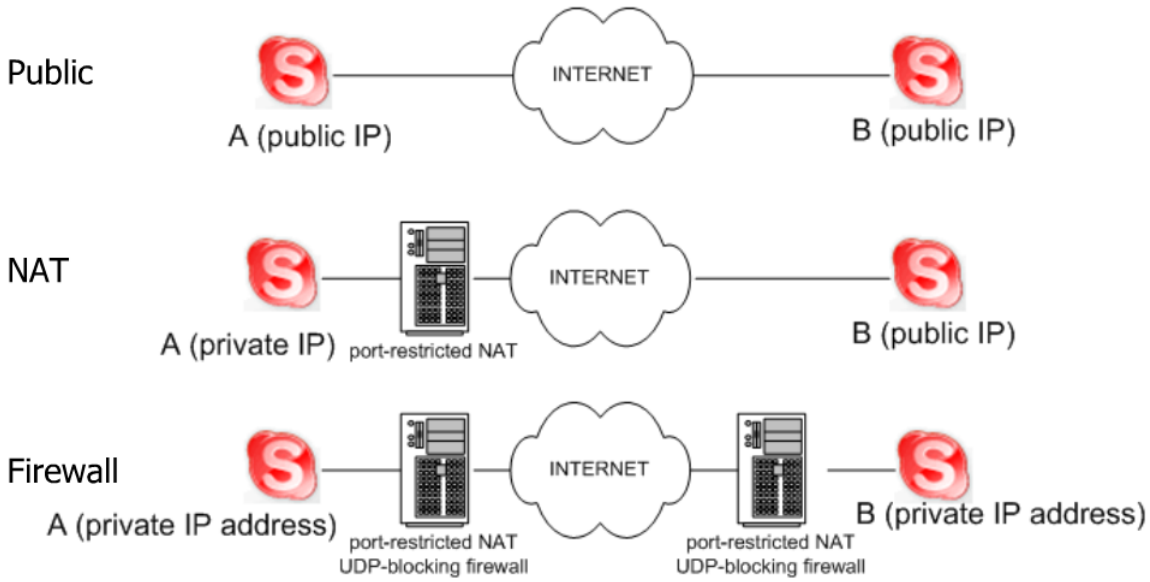


Figura 3.2: Test effettuati per lo studio del comportamento di Skype [BS06a]

Startup

Quando l'SC viene avviato per la prima volta dopo l'installazione, esso invia una richiesta "*http 1.1 GET*" al server di Skype (skype.com). La prima riga di questa richiesta contiene la parola chiave "*installed*". Durante la successiva fase di avvio, l'SC invia, per una sola volta, una richiesta "*http 1.1 GET*" allo Skype Server per verificare se è disponibile una versione aggiornata: la prima riga della richiesta contiene la parola chiave "*getlatestversion*".

Login

La funzione di login è forse la più critica fra le operazioni compiute da Skype. È durante questa fase che l'SC autentica il suo username e password sul login server (mediante il protocollo TCP), annuncia la propria presenza agli altri peers e contatti (*buddy list*), determina il tipo di NAT e firewall (se presenti) e scopre nuovi nodi Skype online dotati di indirizzo IP pubblico È stato osservato che questi nodi vengono utilizzati per mantenere connessioni con la rete di Skype nel caso i SN già conosciuti risultino non disponibili.

Il processo di Login

Come già discusso in precedenza, l'HC deve contenere almeno un entry valido per consentire ad un SC di connettersi alla rete di Skype. Se l'HC contiene una sola entry non valida, l'SC non potrà connettersi alla rete Skype: l'utente verrà quindi avvisato del fallimento del processo di login. Di seguito viene descritto dettagliatamente l'algoritmo di login.

L' SC invia un pacchetto UDP alla prima entry disponibile nella HC. Se non ottiene risposta per più di 5 secondi,esso cerca di stabilire una connessione TCP con questa entry. Se la connessione non va a buon fine, il client cerca di stabilire una connessione TCP verso l'indirizzo IP della entry sulla porta 80 (HTTP). Se la connessione fallisce ancora una volta, il client cerca di connettersi all'IP della entry sulla porta 443 (HTTPS), con un timeout di 6 secondi. L'intero processo viene ripetuto 4 volte, al termine delle quali viene riportato il fallimento del processo di login.

È stato osservato che un SC deve stabilire una connessione TCP con un SN per connettersi alla rete Skype. Se non riesce a connettersi ad un SN, segnalerà un errore di login. È da notare che la maggior parte dei firewall sono configurati per permettere il traffico TCP in uscita verso le porte 80 e 443. Un SC dietro un firewall, il quale blocca il traffico UDP e permette selettivamente un certo tipo di traffico TCP, si avvantaggia di questo fatto. Al login, infatti, stabilisce una connessione TCP con un altro nodo Skype dotato di IP pubblico sulla porta 80 o 443.

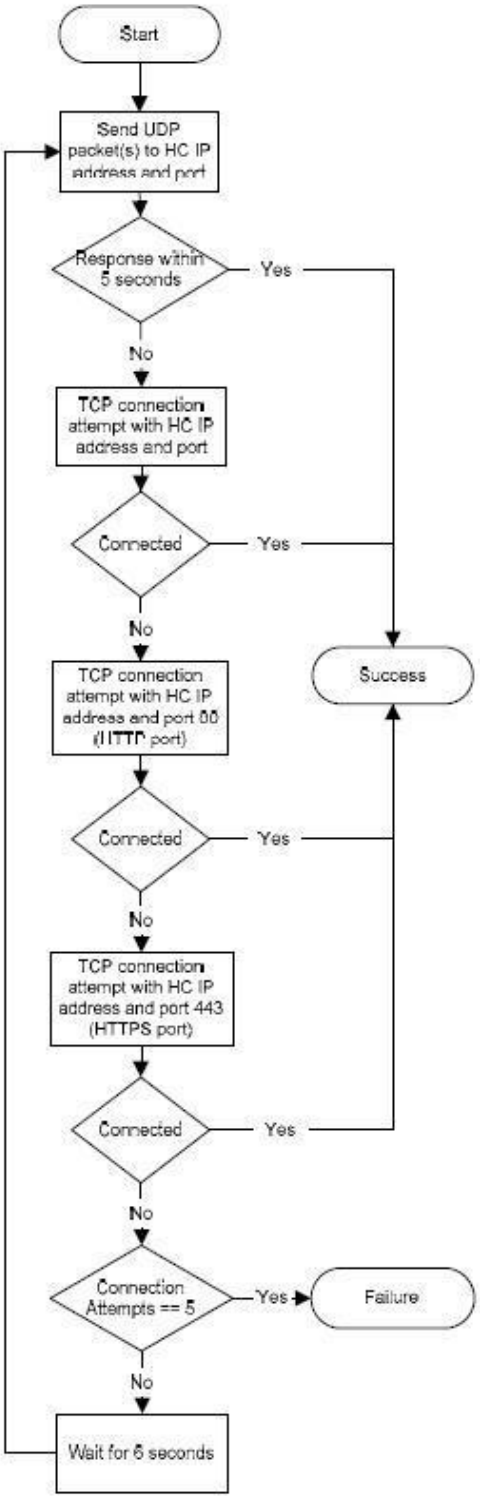


Figura 3.3: *Algoritmo di login* [BS04]. Non è mostrata l'autenticazione con il login server.

Login Server

Dopo che uno SC si è connesso ad un SN, esso deve autenticare username e password sul Skype Login Server. Il login Server è l'unico componente centralizzato sulla rete di Skype. Egli memorizza username e password degli utenti, e assicura l'univocità del login name all'interno della rete Skype. Per il successo del login complessivo, deve necessariamente andare a buon fine il login con il Server.

Durante esperimenti, si è osservato che l'SC scambia sempre dati TCP con un nodo il cui indirizzo è 80.160.91.11: si crede che quello sia il Login Server. Dai dati recuperati attraverso il reverse lookup di questo indirizzo IP, si vede che il Login Server è ospitato da un ISP situato in Danimarca.

Supernodi di Bootstrap

Dopo aver effettuato il login per la prima volta dopo l'installazione, l' HC viene inizializzata con sette coppie indirizzo IP - porta. È stato osservato che prima del primo login, l'HC era sempre inizializzata con queste sette coppie indirizzo IP - porta. è con uno di questi SN che l'SC stabilisce una connessione TCP quando un utente utilizza quell' SC per effettuare il login alla rete di Skype per la prima volta dopo l'installazione. Queste entry vengono definite supernodi di Bootstrap. Dopo l'installazione e la prima volta dello startup, è stato osservato che l'HC è vuota. Tuttavia al primo login l'SC invia pacchetti UDP ad almeno quattro nodi della lista di bootstrap.

Diverse sono le ipotesi sul posizionamento delle informazioni riguardanti i supernodi di boostrap:

- possono essere inserite nel codice dell'SC
- possono essere localizzate nel registro di sistema, cifrate e non direttamente visibili
- può essere un processo che si attiva una sola volta per contattare i supernodi di bootstrap

È stato osservato che, se l'HC viene cancellata dopo il primo login, l'SC non è in grado di connettersi alla rete di Skype. Questa osservazione suggerisce di analizzare la prima volta del processo di login e i successivi separatamente.

Prima Volta del processo di Login

L'HC è vuota al momento dell' installazione. Così, un SC deve connettersi a nodi Skype ben conosciuti al fine di riuscire ad accedere alla rete. Ciò viene effettuato inviando pacchetti UDP ad alcuni supernodi di bootstrap, e quindi attendere le loro risposte, per qualche tempo. Non è chiaro come l'SC selezioni i SN di bootstrap a cui inviare i pacchetti UDP. L'SC stabilisce quindi una connessione TCP con un supernodo di Bootstrap che risponde. Dal momento che più di un supernodo può rispondere, un SC può stabilire una connessione TCP con almeno un SN di bootstrap. L'SC, tuttavia , mantiene una connessione TCP con almeno un nodo di bootstrap, e può chiudere le connessioni TCP con gli altri nodi. Dopo aver scambiato alcuni pacchetti con un SN su TCP, egli probabilmente acquisisce l'indirizzo del Login Server. L' SC stabilisce quindi una connessione TCP con il Login Server, scambia informazioni di autenticazione con esso, ed infine chiude la connessione TCP. La connessione TCP con l'SN persiste finchè è vivo. Quando l'SN diventa indisponibile, l'SC stabilisce una connessione TCP con un altro SN.

I Login successivi

I successivi processi di login sono abbastanza simili al primo. L'SC costruisce l'HC dopo che l'utente ha avuto accesso alla rete per la prima volta dopo l'installazione. L'HC viene periodicamente aggiornata con l'indirizzo IP e il numero di porta di nuovi peer. Durante i login successivi, l'SC utilizza l'algoritmo di Login per un peer disponibile che non sia presente nell' HC. L'SC stabilisce una connessione TCP con quel nodo.

User search

Skype utilizza la sua tecnologia *Global Index* per ricercare gli utenti. Skype dichiara che la ricerca è distribuita e garantisce di trovare un utente, se esiste, e si è connesso nelle ultime 72 ore. Test approfonditi suggeriscono che Skype è sempre in grado di localizzare utenti che si sono connessi utilizzando indirizzi IP pubblici o privati nelle ultime 72 ore. Skype non è un protocollo aperto e i suoi messaggi vengono cifrati. Considerando che nella fase di login Baset e Schulzrinne sono stati in grado di formare una ragionevole opinione sulle entità coinvolte, ci si aspetterebbe di poter riproporre lo stesso nel contesto della ricerca degli utenti; purtroppo non è possibile, dal momento che non è stato possibile tracciare i messaggi dell'SC con il SN. Non è neppure possibile forzare un SC a connettersi ad un particolare SN; tuttavia è stato possibile osservare il flusso dei messaggi per le tre configurazioni di rete già presentate. Come sappiamo, SC possiede una finestra di ricerca. Dopo aver digitato lo user id e premuto il bottone "cerca", l'SC avvia la ricerca per un particolare utente. L'SC invia un pacchetto UDP al proprio SN. Il SN risponde con l'indirizzo IP e il numero di porta di quattro nodi da interrogare. A quel punto, l'SC invia pacchetti UDP a questi nodi. Se non può trovare l'utente, informa il suo SN via TCP. Il SN a questo punto risponde con l'indirizzo IP e il numero di porta di altri otto nodi, e l'SC provvederà ad inviare pacchetti UDP a quest'ultimi. Il processo prosegue finché l'SC trova l'utente o determina che non esiste. In media, durante questo processo, il client contatta otto nodi. Il tempo impiegato per effettuare la ricerca varia dai 3 ai 4 secondi. Non è chiaro su come l'SC termini la ricerca se non è in grado di trovare un utente. Un SC dietro ad un NAT scambia dati con gli SN e alcuni dei nodi che hanno riposto alla sua richiesta, effettuata via UDP. Un SC dietro ad un NAT firewall con restrizioni sul protocollo UDP invia le richieste di ricerca attraverso una connessione TCP al suo SN. Si crede che l'SN elabori la ricerca e restituisca allo SC i risultati. Diversamente dalla ricerca utenti effettuata da un SC su un indirizzo IP pubblico, in questo caso non viene contattato alcun altro nodo. Questo suggerisce che SC sappia di trovarsi dietro ad un firewall che effettui restrizioni sull'UDP.

Memorizzazione dei risultati della ricerca

Per osservare se i risultati delle ricerche vengono memorizzati su nodi intermedi, è stato elaborato il seguente esperimento. L'utente A, installato sulla macchina A, è dietro a un NAT- firewall(restrizione su UDP), ed è loggato sulla rete di Skype. L' utente B, installato sulla macchina B, è loggato utilizzando un SC con indirizzo IP pubblico. L'utente B (con IP pubblico) effettua la ricerca dell'utente A (dietro al NAT-firewall). Successivamente, l'SC sulla macchina B viene disinstallato, e lo Skype Registry cancellato, così che venga eliminata qualsiasi cache locale. L' SC viene successivamente reinstallato sulla macchina B e l'utente B ricerca l'utente A. La ricerca dura circa 3 - 4 secondi. Questo esperimento è stato ripetuto quattro volte in giorni diversi, ottenendo risultati simili. Dai risultati ottenuti si può concludere che l'SC elabora il caching delle informazioni su nodi intermedi.

Apertura e chiusura della chiamata

Apertura della chiamata

Consideriamo l'apertura di chiamata per le tre impostazioni di rete descritte precedentemente. Per ogni configurazione, consideriamo l'apertura della chiamata per utenti che sono nella lista degli amici del chiamante. Per gli utenti che non sono presenti nella lista del chiamante, la chiamata è uguale alla ricerca dell'utente più la segnalazione di chiamata. È importante notare che la segnalazione di chiamata avviene sempre su TCP. Se entrambi gli utenti operano con IP pubblico, sono online e l'uno è nella lista dei contatti dell'altro, allora alla pressione del bottone di chiamata dell'SC del chiamante stabilisce una connessione TCP con l'SC del chiamato. Lo scambio iniziale di messaggi fra chiamante e chiamato indica l'esistenza di un meccanismo di sfida - risposta. Il chiamante invia inoltre alcuni messaggi su UDP per alternare i nodi di Skype, i quali sono nodi online scoperti durante la fase di login. Per questo scenario 3KB di dati vengono scambiati. Nella seconda configurazione, dove il chiamante si trova dietro ad un firewall ed il chiamato possiede un IP pubblico, la segnalazione non fluisce direttamente da chiamante a chiamato. Il chiamante invia le informazioni di segnalazione su TCP ad nodo Skype online il quale le inoltra al destinatario della chiamata, sempre su TCP. Questo nodo online inoltra anche i pacchetti vocali al chiamato su UDP e viceversa. Nel terzo caso, nel quale entrambi gli utenti sono situati dietro a NAT - firewall, entrambi gli SC scambiano informazioni di segnalazione su TCP con un altro nodo Skype online. L'SC del chiamante invia i pacchetti voce su TCP ad un nodo online, il quale li inoltra al chiamato, sempre su TCP e viceversa.

Ci sono diversi vantaggi nell'avere un nodo che inoltri i pacchetti voce fra il chiamante e il chiamato e viceversa:

- quel nodo fornisce un meccanismo per gli utenti dietro a NAT e firewall di parlare ad ogni altro.
- Se gli utenti dietro ad un NAT o firewall vogliono partecipare ad una conferenza, e alcuni utenti con indirizzo IP pubblico vogliono collegarsi, questo nodo funge da mixer e invierà il traffico di conferenza a tutti i partecipanti.

Il lato negativo consiste nel fatto che c'è un'enorme mole di traffico che fluisce attraverso questo nodo, e questo non è molto gradito agli utenti, i quali non vorrebbero che traffico arbitrario attraversi la loro macchina.

Chiusura della chiamata

Nella fase di chiusura della chiamata, le informazioni di segnalazione sono scambiate fra chiamante e chiamato su TCP, se essi sono entrambi su IP pubblico, oppure fra chiamante e chiamato e i rispettivi SN. Per la seconda e la terza configurazione, la segnalazione di fine chiamata è inviata su TCP.

Media transfer

Se entrambi gli SC sono su IP pubblico, il traffico vocale fluisce direttamente fra loro su UDP. Il flusso vocale fluisce tramite la porta UDP configurata nella finestra di dialogo. Se chiamante o chiamato o entrambi sono posizionati dietro ad un NAT, essi inviano traffico voce ad un altro nodo Skype online su UDP. Questo nodo inoltrerà il traffico voce verso il chiamato. Se entrambi gli SC sono situati dietro ad un NAT firewall, essi inviano il traffico voce ad un altro nodo Skype online su TCP. Per il traffico vocale, un SC utilizza TCP con ritrasmissione.

Il protocollo di Skype sembra preferire l'utilizzo di UDP per la trasmissione della voce, se possibile. L'SC utilizzerà UDP per la trasmissione della voce se avrà di fronte un NAT o un firewall che consente il passaggio di pacchetti UDP.

Soppressione del silenzio

La soppressione del silenzio non è supportata da Skype. Quando né chiamante né chiamato stanno parlando, i pacchetti vocali fluiscono comunque. La trasmissione del silenzio ha 2 vantaggi:

- mantiene il binding UDP sul NAT
- questi pacchetti possono essere utilizzati per riprodurre un rumore di background.

Nei casi in cui il traffico fluisca su TCP fra chiamante e chiamato, i pacchetti si silenzio vengono comunque inviati. Lo scopo è quello di evitare il calo della dimensione della congestion windows, perché richiederebbe un po' per raggiungere il livello massimo.

Messaggi di presenza

È stato osservato che, per le tre configurazioni di rete, un SC invia al proprio SN un messaggio di refresh ogni 60 secondi.

Conferenza

In [BS04] gli autori hanno osservato le caratteristiche di una conferenza fra tre utenti Skype per le tre configurazioni di rete. Gli SC sono installati su tre macchine diverse A,B,C.(A possiede caratteristiche di memoria e potenza computazionale maggiore di B e C).

Nella prima configurazione di rete, le macchine possiedono un IP pubblico. Inizialmente fra A e B viene stabilita una chiamata. Successivamente B decide di includere C nella conferenza. È stato osservato che B e C inviano il loro traffico vocale su UDP all'SC sulla macchina A, il quale agisce da mixer. Egli miscela i suoi pacchetti con quelli di B e li invia a C utilizzando UDP.

Nella seconda configurazione, B e C sono situati dietro ad un NAT port restricted, mentre A è su indirizzo IP pubblico. Inizialmente A e B stabiliscono la chiamata. Sia A che B inviano i loro pacchetti vocali ad un altro nodo Skype online, il quale inoltra i pacchetti di A verso B e viceversa, utilizzando UDP. L'utente A pone allora B in attesa e stabilisce una chiamata con C. Inizia successivamente la conferenza fra i 3 SC. è stato osservato che B e C inviano i loro pacchetti ad A su UDP, il quale miscela i suoi pacchetti provenienti da B e C e li inoltra ad essi in modo appropriato.

Nella terza configurazione di rete, B e C sono situati dietro ad un NAT port -restricted FIREWALL UDP - restricted, mentre A possiede un indirizzo IP pubblico. L'utente A inizia la conferenza con B e C. È stato osservato che B e C inviano il loro traffico vocale verso A utilizzando TCP. Come nei casi precedenti, A miscela i suoi pacchetti con quelli provenienti da B e C e li inoltra ad essi in modo appropriato. È stato inoltre osservato che, anche se gli utenti B o C iniziano la conferenza, la macchina A, che la più performante fra le tre, viene eletta conference e mixed host.

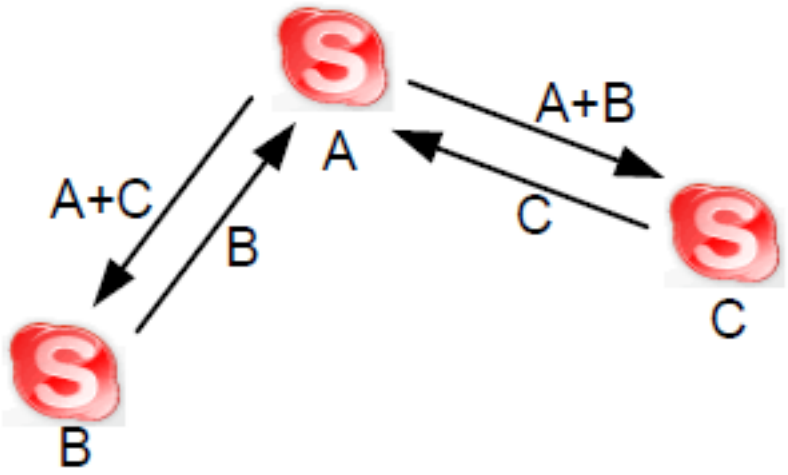


Figura 3.4: conferenza fra tre utenti skype [BS06b]

Capitolo 4

Lo Skype Client e l'interfacciamento con le API

Come spiegato nel capitolo precedente, Skype è un software proprietario freeware di messaggistica istantanea e VoIP, ed è un elemento fondamentale nell'architettura del protocollo descritto nel capitolo precedente.

In questo capitolo vengono descritte le funzionalità offerte dallo Skype Client e il modo con cui è possibile interfacciarsi ad esso per poter costruire nuovi servizi.

4.1 L'applicazione Skype

Sviluppato nel 2002 da Niklas Zennström e Janus Friis, l'applicazione di Skype unisce caratteristiche presenti nei client più comuni (chat, salvataggio delle conversazioni, trasferimento di file) ad un sistema di telefonate basato su un network Peer-to-peer.

Le principali funzionalità da Skype, descritte in [Kal] e [skl10], vengono riportate in figura 4.1.

I servizi VoIP vengono implementati da Skype attraverso la funzionalità *Call*, la quale consente di effettuare chiamate e videochiamate verso utenti Skype; questa modalità di servizio è implementata per la comunicazione diretta e multiutente.

I servizi di Instant Messaging sono implementati dalle Skype attraverso le funzionalità di *chat*, che consentono di effettuare comunicazioni con altri utenti Skype attraverso messaggi testali: anche per questa funzionalità è disponibile la modalità multiutente.

Altre interessanti funzionalità sono le *SMS*, che consentono di inviare e ricevere messaggi da qualsiasi dispositivo mobile e l' *AP2AP*, che consente lo scambio di informazioni testuali fra applicazioni che interagiscono con i client di Skype, senza che quest'ultimi siano in grado di rilevare quest'attività.

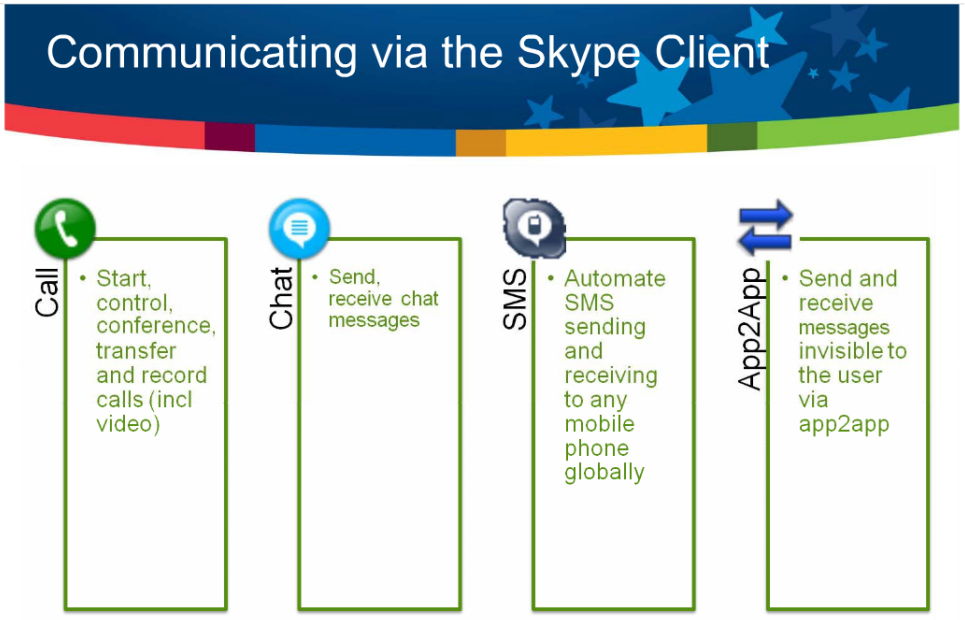


Figura 4.1: funzionalità offerte dal client di Skype

Oltre a queste, in [skl10] vengono elencate funzionalità addizionali che sono state sviluppate di recente: Skype rende possibile la condivisione dello schermo e lo scambio di file fra i suoi utenti.

I servizi descritti appartengono alla modalità di funzionamento nota come *Skype - to - Skype*, in cui entrambi i terminali della comunicazione sono utenti Skype.

Altri servizi sono offerti da Skype per consentire l'interoperabilità con la rete PSTN:

- *SkypeIN* offre la possibilità di associare ad un utente Skype un numero di telefono fisso: in questo, modo un utente PSTN è in grado di comunicare con l'utente Skype chiamando il numero ad esso collegato.
- *SkypeOUT* consente invece da un utente Skype di poter effettuare una comunicazione con un utente di telefonia PSTN: in questo caso, la comunicazione è instradata via Internet fino alla nazione del destinatario, dove viene dirottata sulla normale rete telefonica del Paese consentendo un notevole abbattimento dei costi di chiamata.

4.2 L'interfacciamento con applicazioni esterne

Skype rende possibile l'interazione con la sua applicazione attraverso un'Application Programming Interface (API). Nell'accezione più comune del termine, le

API sono librerie di codice preconfezionate che gli sviluppatori possono riutilizzare ogni volta che hanno bisogno di integrare nelle loro applicazioni le funzioni messe a disposizione da un sistema esistente. Un'API è quindi la specifica dei parametri che un'applicazione esterna deve passare a una routine e dei risultati che deve restituire. Con questo meccanismo, un programmatore può invocare all'interno del codice della propria applicazione un insieme di routine predefinite che di utilizzare le funzionalità di Skype.

Una semplice spiegazione sulle API è proposta da Dickmann [Dic07], rappresentata in figura 4.2.

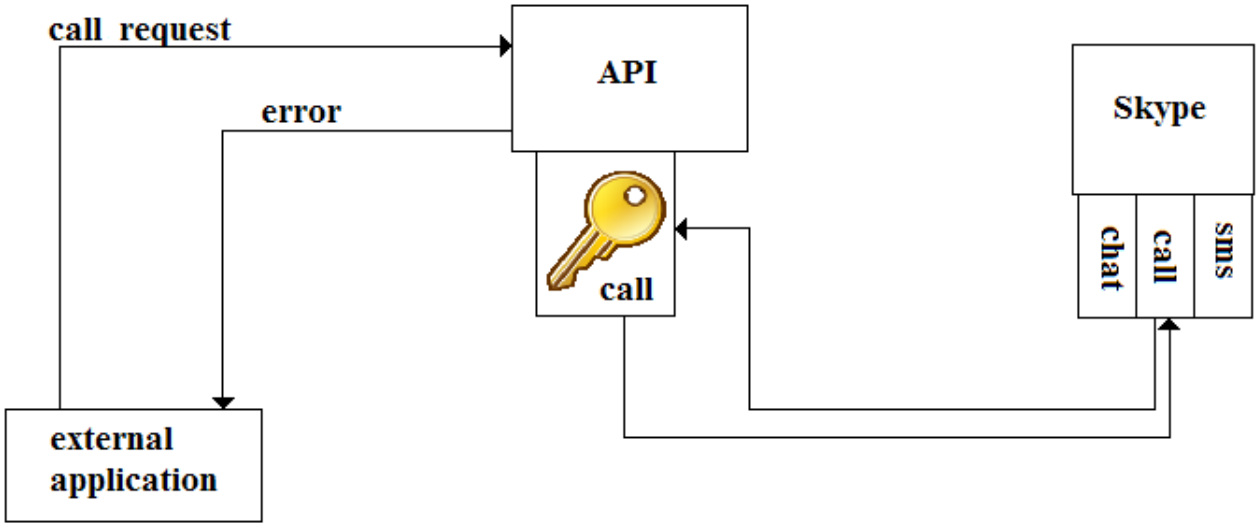


Figura 4.2: L'interfacciamento attraverso le API

Nell'esempio, i servizi offerti da Skype sono erogati da elementi di una libreria. Per potrevi accedere, l'applicazione esterna deve invocare una chiave di accesso pubblicata sulle API. Se la chiave esiste, le API inoltreranno la richiesta allo Skype Client, tradotta in un formato comprensibile a quest'ultimo: l'applicazione di Skype elaborerà quindi la funzione di libreria associata alla richiesta per poi restituire alle API il risultato dell'esecuzione. A questo punto le API elaboreranno il processo di traduzione inversa, per presentare all'applicazione esterna il risultato in formato comprensibile al programmatore.
Nel caso in cui non fosse presente la chiave nelle API, all'applicazione esterna verrà inviato un messaggio di errore.

Le API permettono quindi un ingresso controllato alle funzionalità di Skype permettendo allo stesso tempo di espandere quelle messe a disposizione. Per un'azienda di sviluppo software, mettere a disposizione un set di API di un suo software significa dare la possibilità ad altri di interagire con la sua piattaforma e, soprattutto, estendere le funzioni e le caratteristiche della struttura base della

piattaforma. In altri termini, le API sono un ottimo strumento per promuovere un programma offrendo ad altri un modo per interagirci.

Secondo [Kri06], Skype consente l'utilizzo delle API in modo vincolato:

- un'opzione è di rendere l'applicazione sviluppata *freeware* o non fare profitti tramite la vendita del prodotto sviluppato
- l'altra opzione è di condividere con il proprietario di Skype i profitti di vendita, o di pagare i diritti d'uso delle API.

Le Application Programming Interface (API) di Skype si suddividono in due grandi sezioni: le *Skype Phone API* e le *Skype Access API*.

Le *Phone API* sono driver di interfacciamento fra l'applicazione di Skype e i dispositivi hardware , come i telefoni USB o i dispositivi virtuali, come le applicazioni "softphone". Per i dispositivi hardware le Phone API sono come dei driver di periferiche dei sistemi operativi: questi driver scambiano messaggi con con l'applicazione di Skype, indicando eventi quali il suono del telefono, la pressione di un tasto per una composizione o il sollevamento della cornetta per la risposta ad una chiamata.

Le *Access API* [Gou06] [Hin05] sono di maggior interesse per gli utenti di Skype, perché quest'interfaccia supporta i comandi per inviare e ricevere messaggi, effettuare e ricevere chiamate e così via da un'applicazione esterna. Il team di sviluppo di Skype ha riconosciuto che gli utenti potrebbero essere desiderosi di estendere l'applicazione Skype in un modo che non avevano previsto. Uno sviluppatore autonomo vorrebbe essere in grado di aggiungere funzionalità che non sono prioritarie sulla lista di sviluppo dell'azienda detentrice dei diritti di Skype: per rendere possibile l'accesso a queste funzionalità da applicazione esterna, il team di sviluppo di Skype ha rilasciato una Application Programming Interface (API) apposita.

Ci sono molti esempi d'utilizzo delle Access API di Skype: fra questi, di interesse per il presente lavoro è il trasferimento di chiamata personalizzato e la gestione del comportamento dell'applicazione di Skype alla ricezione della chiamata. Di seguito si parlerà esclusivamente di quest'ultime.

4.3 Panoramica sulle Access API

Le Access API forniscono un meccanismo che consenta ad applicazioni esterne di controllare le funzioni offerte dall'interfaccia di Skype.
Queste API si presentano su due livelli distinti:

- il *Communication Layer* è un insieme di metodi che le applicazioni esterne utilizzano per stabilire una connessione con lo Skype Client e comunicare con esso
- il *Protocol Layer* è un linguaggio text - based che le apPlicazioni esterne utilizzano per "parlare" allo Skype Client un volta che è stato instaurato il canale di comunicazione per mezzo del Communication Layer

Ci sono inoltre diverse librerie wrapper che incapsulano le funzionalità delle API di Skype, formando un terzo livello.

Di seguito verranno approfonditi i livelli delle Access API

4.3.1 Il Comunication layer

Come già introdotto, il Comunication layer fornisce alle applicazioni esterne un meccanismo per comunicare con lo Skype Client.
Questo strato è dipendente dalla piattaforma: è implementato per Windows, Linux e Mac OS X. Per informazioni su come sia stata effettuata l'implementazione sui diversi sistemi operativi citati, si veda il documento di riferimento per le API di Skype [Hin05].

Una volta che la domanda è stato effettuato il collegamento allo Skype Client attraverso il livello di comunicazione, l'applicazione esterna può avviare la comunicazione utilizzando i comandi previsti nel Protocol Layer, il quale verrà presentato nel paragrafo successivo.

4.3.2 Il Protocol Layer

Lo strato di protocollo è il linguaggio per poter comunicare con lo Skype Client; la comunicazione avviene tramite scambio di messaggi testuali codificati in formato UTF - 8 [1].
I messaggi possono essere di tre tipi:

- *Comandi*: messaggi sincroni inviati dall'applicazione a Skype per richiedere un'azione, come ad esempio iniziare una chiamata o leggere il profilo di un utente;
- *Risposte*: messaggi che Skype invia all'applicazione in risposta a un comando;
- *Notifiche*: messaggi asincroni che vengono inviati da Skype quando c'è un cambiamento nello stato di un oggetto (per oggetto si intende l'oggetto nell'architettura delle API di Skype), oppure viene richiesto il valore della proprietà di un oggetto con un comando GET. Un messaggio di notifica avviene anche nel caso in cui il valore di un oggetto sia stato modificato con un comando SET. In [Hin05] viene inoltre specificato che le notifiche vengono inviate per informare su variazioni di proprietà di oggetti "rilevanti": per esempio informazioni sugli utenti della buddylist, sulle chiamate attive o sulle chat attive.

Prima di illustrare con un esempio le modalità di interazione fra applicazione esterna e lo Skype Client, vengono mostrate le sintassi dei messaggi scambiati.

La sintassi utilizzata per i comandi è $\rightarrow$ #⟨*command_id*⟩ *command*.
La sintassi della risposta ad un comando è $\leftarrow$ #⟨*command_id*⟩ *response*|*error*.

Infine, la sintassi di una notifica è $\leftarrow$ ⟨*object _id*⟩ *property value*.

Nell'esempio sottostante, nel quale viene illustrata un'interazione fra un'applicazione esterna e il client di skype, i messaggi con la freccia rivolta verso destra indicano un messaggio inviato dall'applicazione, mentre gli altri provengono dal client.

⟨ 1 ⟩ $\Longleftarrow$ CONNSTATUS ONLINE
⟨ 2 ⟩ $\Longleftarrow$ CURRENT USER HANDLE simone-peruzzo
⟨ 3 ⟩ $\Longleftarrow$ USERSTATUS ONLINE
⟨ 4 ⟩ $\Longrightarrow$ NAME SkypeAPI4Java
⟨ 5 ⟩ $\Longleftarrow$ NAME SkypeAPI4Java
⟨ 6 ⟩ $\Longrightarrow$ PROTOCOL 9999
⟨ 7 ⟩ $\Longleftarrow$ PROTOCOL 8
⟨ 8 ⟩ $\Longrightarrow$ # 0 CALL echo123

[1] Unicode Transformation Format: sistema di codifica dei caratteri Unicode che utilizza un numero variabile di byte, da 1 a 4 byte, a seconda delle necessità

⟨ 9 ⟩ ⟸ # 0 CALL 348109 STATUS UNPLACED
⟨ 10 ⟩ ⟸ CALL 348109 STATUS ROUTING
⟨ 11 ⟩ ⟸ CALL 348109 STATUS RINGING
⟨ 12 ⟩ ⟸ CALL 348109 STATUS INPROGRESS
⟨ 13 ⟩ ⟸ CALL 348109 DURATION 1
⟨ 14 ⟩ ⟸ CALL 348109 DURATION 2
⟨ 15 ⟩ ⟸ CALL 348109 DURATION 3
⟨ 16 ⟩ ⟸ CALL 348109 DURATION 4
⟨ 17 ⟩ ⟸ CALL 348109 STATUS FINISHED

Di seguito vengono descritti i messaggi scambiati nell'esempio:

- ⟨ 1 ⟩, . . . , ⟨ 3 ⟩: notifiche all'applicazione sullo stato attivo dello Skype Client, sull'utente gestito e sull'attuale stato online dell'utente.
- ⟨ 4 ⟩, ⟨ 5 ⟩: notifiche del tipo di API utilizzate.
- ⟨ 6 ⟩, ⟨ 7 ⟩: richiesta del numero di protocollo del *protocol layer* utilizzato dallo Skype Client e notifica delo stesso, informando l'applicazione che egli utilizza la versione *PROTOCOL 8*
- ⟨ 8 ⟩: l'applicazione invia un comando per la richiesta di chiamata al servizio echo123;
- ⟨ 9 ⟩: lo Skype Client risponde avvisando che la chiamata non è stata ancora inoltrata;
- ⟨ 10 ⟩ la chiamata viene inoltrata;
- ⟨ 11 ⟩: 11: notifica dell'inizio della segnalazione acustica per l'utente destinatario della richiesta di comunicazione vocale;
- ⟨ 12 ⟩: : notifica inizio chiamata;
- ⟨ 13 ⟩, . . . , ⟨ 16 ⟩: notifiche di informazioni sulla durata della chiamata;
- ⟨ 17 ⟩: notifica del termine della chiamata

Il protocollo di Skype è attualmente alla sua ottava versione. A partire con il protocollo 1 (il primo protocollo di Skype), una nuova versione è stata creata solo quando i nuovi comandi diventano incompatibili con i comandi già esistenti.

Ogni volta che viene rilasciata una nuova versione delle API viene incrementata la versione del protocollo. Quando un'applicazione esterna avvia un dialogo con lo Skype Client, deve interrogare le API per conoscere l'ultima versione del protocollo che supporta. Lo Skype Client replicherà con la versione del protocollo in grado di supportare: per ottenere quest'informazione, l'applicazione esterna deve interrogare lo Skype Client con il comando *PROTOCOL 9999*. Conoscere

la versione che lo Skype Client è in grado di supportare è molto importante. L'applicazione deve adottare la versione del protocollo usata dallo Skype Client. Per esempio, se lo Skype Client è in grado di supportare il *PROTOCOL 2* e l'applicazione esterna utilizza le API con versione *PROTOCOL 3*, quest'ultima dovrà impiegare la versione *PROTOCOL 2*.

4.3.3 Gli oggetti

Skype [ska] esporta il suo stato verso le applicazioni esterne attraverso una serie di astrazioni chiamate *oggetti* con proprietà, sui quali si possono eseguire essenzialmente due tipi di operazioni ovvero la visualizzazione e la modifica delle proprietà. Alcuni degli oggetti che appartengono allo Skype Protocol sono:

- *User* e *Profile*: rappresentano l'utente di Skype con tutte le informazioni del suo profilo (nome, cognome, sesso, ...)
- *Call* : rappresenta le sessioni di chiamata vocale;
- *Chat* : rappresenta le sessioni di chat;
- *Application*: questo oggetto viene utilizzato dalle applicazioni esterne che vogliano comunicare con il client Skype per rappresentare se stesse; esso definisce una sottoparte delle API di Skype chiamata AP2AP. AP2AP è la parte specializzata nella comunicazione end-to-end tra diverse istanze di una stessa applicazione situate su host diversi.
- *Group*: quest'oggetto consente agli utenti di raggruppare i contatti.

4.4 I Wrapper Ufficiali per le API di Skype

Tipicamente non è necessario conoscere uno ad uno questi messaggi in quanto esistono diversi wrapper delle API:

- Skype4COM per ActiveX
- Skype4Java per il linguaggio Java
- Skype4Py per il linguaggio Python

Skype4COM incapsula le API di Skype in oggetti ActiveX: questo permette di utilizzare le API da qualunque linguaggio di programmazione che possa utilizzare oggetti ActiveX, con la limitazione che queste applicazioni possono essere utilizzate solamente su piattaforma Windows. Skype4Py e Skype4Java permettono rispettivamente l'utilizzo delle API di Skype con i linguaggi a oggetti Python e Java, rendendo così le applicazioni multi - piattaforma.

In questo elaborato si parlerà più approfonditamente di *Skype4Java*, poiché questo wrapper è stato impiegato nei casi di studio delle applicazioni VoIP.

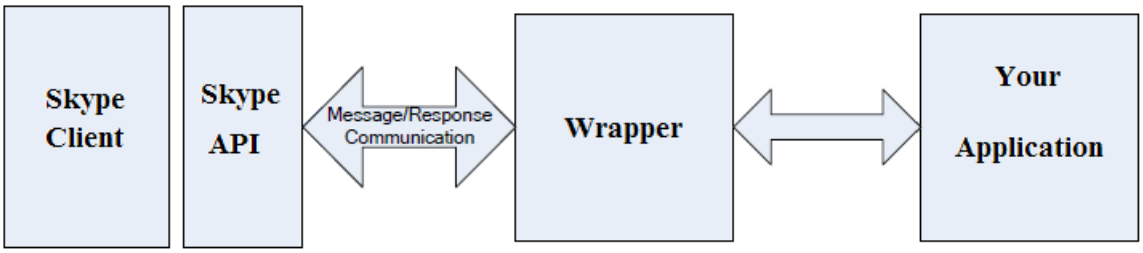


Figura 4.3: il posizionamento del wrapper

4.5 Il wrapper Skype4Java

Skype4Java [BL], è un wrapper frutto della collaborazione fra due sviluppatori: Koji Hisano (UBION INC) e Bart Lamot. Alcuni esempi di utilizzo di questo wrapper sono stati presentati da Matteo Baccan [mat] durante la manifestrazione Javaday2007.

Figura 4.4: logo di Skype4Java

Come è stato detto, le API di Skype sono composte da due livelli: il comunication layer e il protocol layer.

Il *Communication Layer* è implementato attraverso la classe astratta *Connector*, che rappresenta il meccanismo di comunicazione fra client Skype attivo e le API. Essendo questo layer platform - dependent [2], la parte del meccanismo di connessione specifico per ogni piattaforma viene implementato da un insieme di classi che estendono la classe Connector.

Le classi che estendono la classe astratta Connector sono *LinuxConnector* per Linux, *OSXConnector* per Mac OS X e due connettori specifici per Windows, WindowsConnector e Win32Connector. In Windows, la distinzione fra i due connettori sta nel fatto che Win32Connector utilizza una piccola Dynamic Linking Library (DLL) per connettersi al client di Skype mentre WindowsConnector utilizza la libreria Standard Widget Toolkit (SWT). In figura 4.5 viene fornita una rappresentazione dell'architettura del comunication layer attraverso un diagramma Unified Modeling Language (UML) delle classi.

[2]per comprendere quale sistema operativo sia presente sulla macchina dove è attivo lo skype client, viene utilizzato il metodo getProperty("os.name") della classe *System*

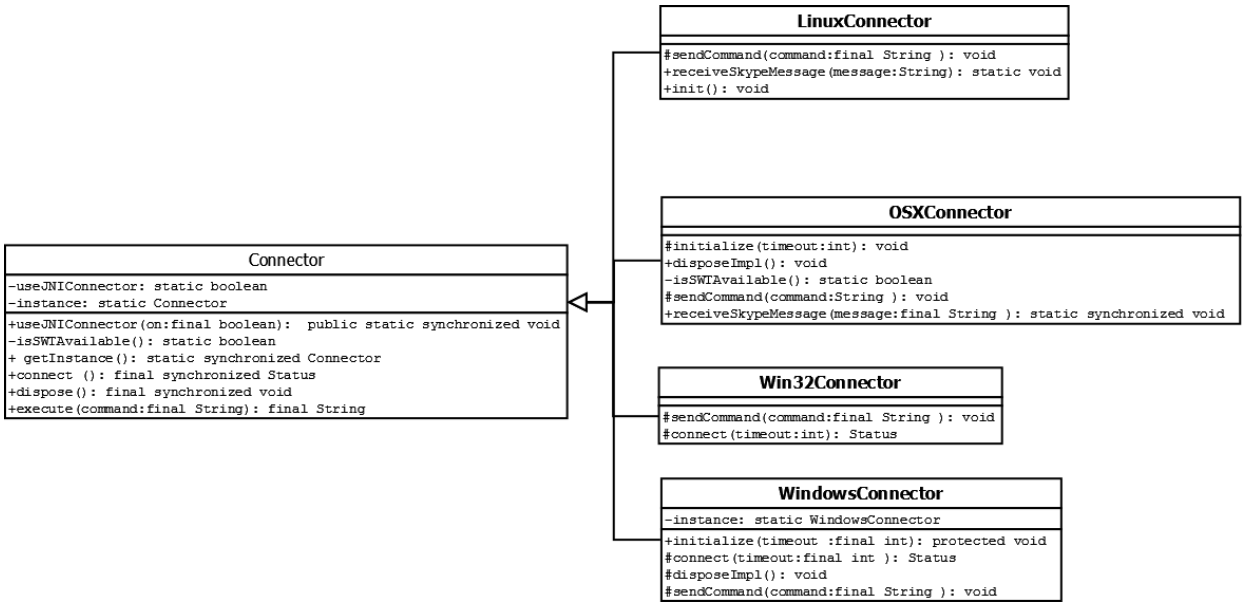


Figura 4.5: architettura del comunication layer

Il livello successivo, rappresentato dal *Protocol Layer*, viene implementato tramite una serie di classi che rappresentano gli oggetti delle Skype API.
Nella figura 4.6 viene proposto un diagramma UML per modellare le relazioni fra le principali classi di Skype4Java che definiscono questo livello.

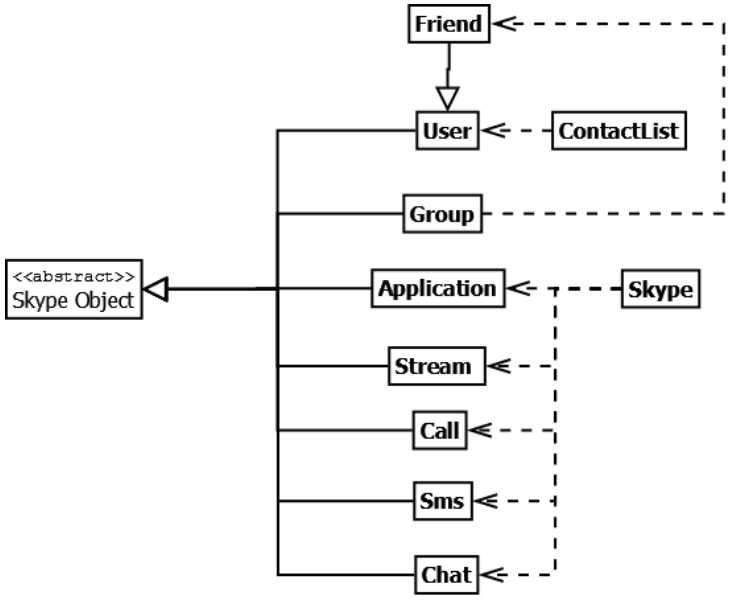


Figura 4.6: relazioni fra le principali classi del protocol layer

La classe principale è *Skype*. Essa fornisce informazioni sull'intero modello del

Protocol Layer, mentre le funzioni al suo interno vengono richiamate staticamente per costruire i vari oggetti della libreria: in questo modo l'applicazione esterna è in grado di spedire i messaggi di comando al client di Skype. Ad esempio è possibile effettuare una chiamata attraverso la funzione Skype.call(String SkypeID): tale funzione prende in input una stringa che rappresenta l'utente da chiamare e restituisce in output un oggetto di tipo Call che rappresenta il corrispondente oggetto Call delle API, e può essere utilizzato per ottenere informazioni sulla chiamata o per modificarne le proprietà (ad esempio è possibile mettere la chiamata in attesa attraverso la funzione hold()).

Per gestire i messaggi di notifica che Skype invia all'applicazione, Skype4Java utilizza gli *Event Listener* [3].

Di seguito verranno approfondite le classi impiegate nelle applicazioni proposte nei prossimi capitoli. Si parlerà della classe *Call*, la quale implementa tutte le specifiche di SKYPE CALL protocol e degli event listener che si occupano di gestire le notifiche riguardanti le chiamate.

Successivamente verranno trattare le classi *Stream* e *Application*, le quali gestiscono le API AP2AP di Skype. In particolare la classe Application rappresenta l'applicazione che intende connettersi al client di Skype.Anche in questo caso verranno illustrati tutti gli event listener associati a queste classi.

4.5.1 Gestione della chiamata e Skype4Java

L'istanza di una chiamata viene rappresentata in Skype4Java dalla classe *Call*, la quale mappa i comandi dell'oggetto Call delle API nei suoi metodi. In particolare:

- *ALTER CALL ⟨ id ⟩ END FORWARD_CALL*: questo comando, mappato nel metodo *forward()*, consente l'inoltro della chiamata verso un altro utente Skype;
- *SET CALL ⟨ id ⟩ STATUS FINISHED*: questo comando, mappato nel metodo *finish()*, impone la conclusione della comunicazione della chiamata ponendo il suo stato a *FINISHED*;

La classe *Call* consente inoltre l'estrazione dei valori delle proprietà dell'oggetto *CALL* delle API. In particolare :

- *TIMESTAMP*: questa proprietà, reperita attraverso la funzione *getStartTime()*, consente di ottenere l'orario in cui è stata ricevuta la chiamata.
- *PARTNER_HANDLE*: questa proprietà, reperita attraverso la funzione *getPartnerId()*, consente di ottenere l'identificativo dell'altro utente coinvolto nella chiamata.

[3] oggetti utilizzati in Java per reagire ad eventi asincroni

- *DURATION*: questa proprietà, reperita attraverso la funzione *getDuration()*, consente di ottenere il valore della durata della chiamata, espresso in millisecondi.

In figura 4.7 è rappresentata la gerarchia delle classi di gestione della chiamata.

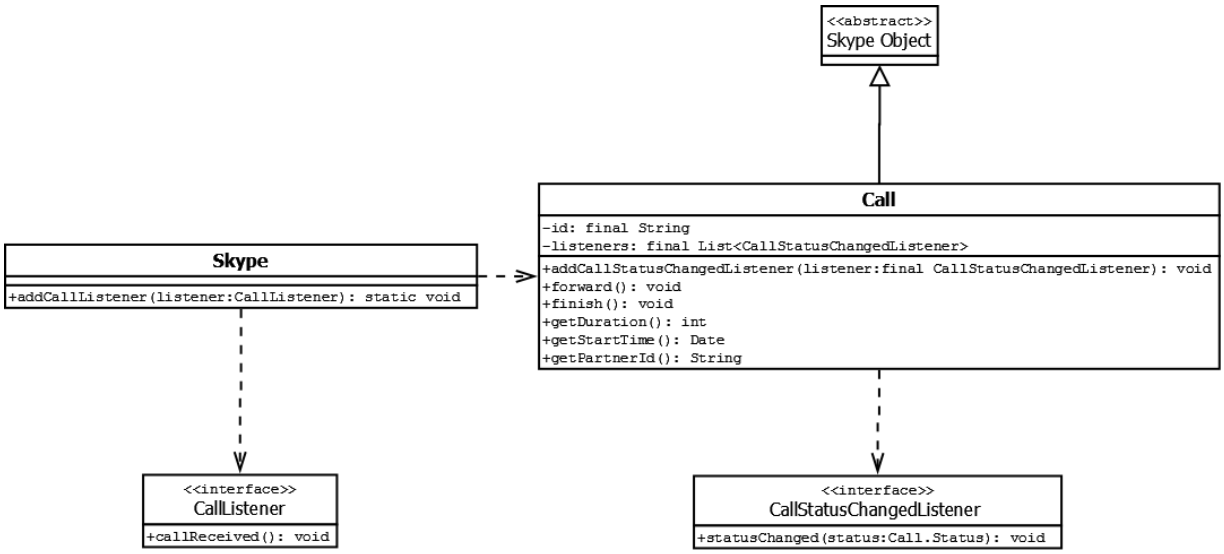


Figura 4.7: diagramma delle classi per la gestione della chiamata

Per intercettare le chiamate provenienti dalle API, alla classe *Skype* viene associato un oggetto di tipo *CallListener* utilizzando il metodo *addCallListener(CallListener listener)*. Il metodo*callReceived(Call receivedCall)* di *CallListener*, permette di associare alla chiamata ricevuta un oggetto di tipo *Call*. Inoltre, la libreria di Skype4Java consente la gestione dei cambiamenti di stato della chiamata attraverso il metodo *statusChanged(Call.Status status)* della classe *CallStatusChangedListener*.

4.5.2 AP2AP e Skype4Java

Come è stato detto, l'istanza di un'applicazione che vuole effettuare una comunicazione AP2AP con un'altra istanza della stessa ospitata su un host diverso, è rappresentata dalla classe *Application*. In questa classe vengono mappati i seguenti comandi di AP2AP:

- *Create*: questo comando, mappato della funzione *addApplication(String name)*, stabilisce un collegamento tra l'applicazione (identificata con il campo *name*) e il client di Skype;
- *Connect* questo comando, mappato nella funzione *connect(Friend f)*, stabilisce un canale di comunicazione (stream) tra due istanze di una stessa

applicazione situate su due host diversi. f è l'ID che identifica il client di Skype remoto a cui ci si connette;

- *Disconnect*: questo comando, mappato nella funzione *disconnect(Friend f)*, permette la chiusura dello stream di comunicazione con il client identificato da f ;
- *Delete*: questo comando, mappato nella funzione *finish()*, disconnette l'applicazione dal client Skype chiudendo tutti gli stream aperti.

La classe *Stream* rappresenta invece un canale di comunicazione AP2AP. Attraverso questo canale, due istanze della stessa applicazione collegati a client Skype situati su host diversi possono comunicare. Nella classe Stream, i metodi rilevanti sono:

- *Write*: questo comando, mappato sulla funzione *write(String s)*, permette di inviare la stringa *s* nello stream precedentemente creato con la funzione *connect()*;
- *Read*: questo comando, mappato sulla funzione *textReceived()* della classe StreamListener, permette di leggere i dati in arrivo sullo stream;

In figura 4.8 è rappresentata la gerarchia delle classi di AP2AP. Le classi *Application* e *Stream* estendono la classe *SkypeObject*, che è la classe madre dell'architettura di Skype4Java. Il metodo *addApplication()* della classe *Skype* permette di creare un oggetto di tipo Application. Tramite la funzione connect() della classe Application viene stabilita la comunicazione AP2AP e viene istanziato un oggetto *Stream* che rappresenta il canale di comunicazione.

Con la funzione *addStreamListener()* della classe *Stream* si può associare al canale un oggetto *StreamListener*, il quale definisce le azioni da eseguire alla ricezione di dati sullo stream. Allo stesso modo con la funzione *addApplicationListener()* della classe *Application*, viene associato all'applicazione un oggetto *ApplicationListener* che si occupa di gestire le connessioni in entrata da parte di istanze remote dell'applicazione.

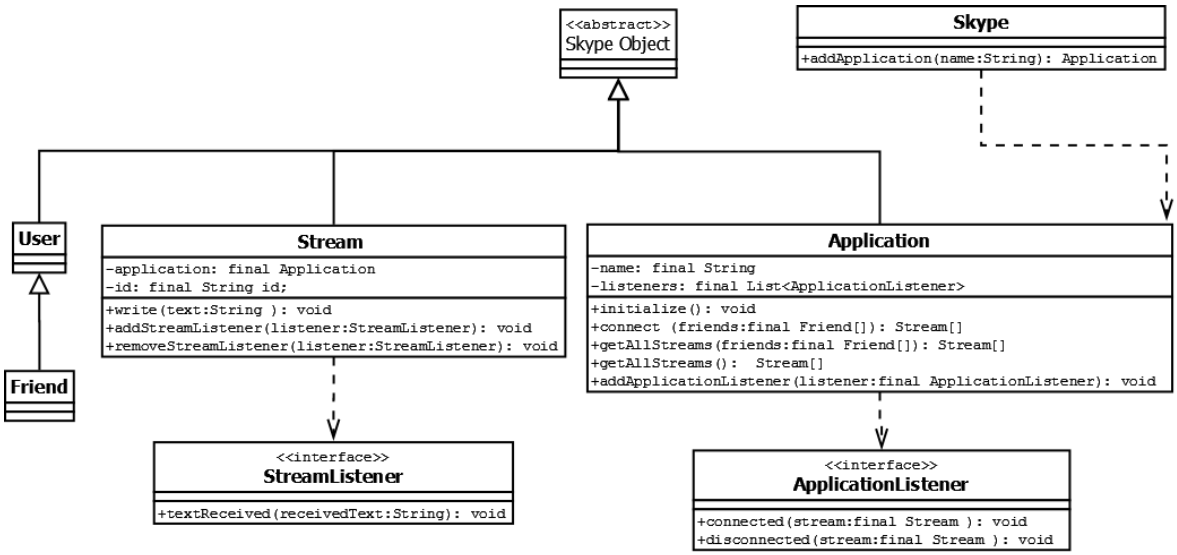


Figura 4.8: diagramma delle classi Application e Stream

Capitolo 5

Il sistema VoIP progettato in Zucchetti

Con l'avvento del Web 2.0 Internet è diventato lo spazio nel quale condividere, collaborare, comunicare, socializzare, partecipare e ciò ha reso possibile il modello dell'impresa estesa, ossia di un'azienda aperta verso l'esterno che interagisce in modo facile e veloce con i propri clienti.

Lo scenario concreto proposto dal paradigma Web 2.0 è quello di un utente non più semplice fruitore dei contenuti presentati nel web: egli è in grado di creare contenuti, e di interagire con gli altri utenti.

Il caso di studio qui proposto parte da questo concetto: sfruttare Internet come mezzo per facilitare la comunicazione con i clienti dell'azienda e poter far fronte alle loro esigenze in modo rapido ed efficiente. Nel caso specifico, un cliente, mentre utilizza un prodotto sviluppato dall'azienda, potrebbe avere la necessità di chiamare il servizio assistenza: nasce quindi l'esigenza, da parte dell'azienda, di sviluppare un' applicazione VoIP che faciliti quest'interazione.

Già in passato Zucchetti aveva sviluppato alcuni prototipi di software VoIP per la gestione delle assistenze che sostituisse il vecchio sistema basato su PBX.

Un primo prototipo di software che svolgesse le funzioni di Automatic Call Distribution è stato progettato nel 2006: tale sistema è stato implementato utilizzando Netmeeting[1], un client VoIP che utilizza il protocollo di segnalazione H.323. Tuttavia gli scarsi risultati tecnici hanno contribuito all'abbandono di quel progetto.

Più di recente è stato sviluppato un progetto, basato su Skype [Aha]. Questo progetto prevede che tutti gli Skype Client siano attivati sulle postazioni degli operatori con lo stesso identificativo: alla ricezione di una chiamata, tutti gli Skype Client iniziano a squillare e il primo operatore che risponde inizia la comunicazione con il cliente, mentre tutti le altre applicazioni Skype segnalano la fine della chiamata. La forte limitazione di questo sistema è il vincolo sul posizionamento degli utenti, i quali devono essere situati sulla stessa LAN.

[1] il celebre software VoIP di Microsoft

Il sistema che verrà descritto in questo capitolo ha lo scopo di eliminare questo limite, utilizzando un modulo intermedio che svolga la funzione di Automatic Call Distribution. La descrizione viene suddivisa in tre parti.

Nella prima parte viene presentato uno studio delle funzionalità che il sistema deve offrire e della *fattibilità* del progetto. I requisiti vengono definiti utilizzando una descrizione narrativa e un insieme di casi d'uso, mentre la valutazione della fattibilità verrà effettuata basandosi su un'analisi dei costi - benefici, sulla durata temporale presunta del sistema e sull'applicabilità di questa soluzione nell'azienda.

Nella seconda parte viene presentata l'architettura del progetto, corredata da degli screenshot catturati durante la fase di test del prototipo iniziale sviluppato in azienda.

La fase conclusiva viene impiegata per offrire spunti di sviluppo del progetto.

5.1 Requisiti e casi d'uso

L'applicazione VoIP che verrà descritta deve essere in grado di gestire tutte le chiamate provvenienti dai clienti dell'azienda. Gli attori del sistema sono i clienti, gli operatori e un centralino VoIP con funzioni di Automatic Call Distribution (ACD).
Il sistema deve essere in grado di ricevere le chiamate e di inoltrarle verso gli operatori disponibili all'assistenza; nel caso di mancata disponibilità di operatori, il centralino dovrà notificare al cliente l'impossibilità all'erogazione del servizio attraverso un messaggio di chat. Il sistema dovrà inoltre essere in grado di riconoscere eventuali utenti non clienti dell'azienda: le loro chiamate dovranno essere rifiutate, notificando al mittente la motivazione.
Il sistema deve rendere possibile all'operatore la visualizzazione dei dati del cliente e lo storico delle assistenze precedenti alla ricezione di una chiamata.
Dai colloqui tenuti con il responsabile Zucchetti per la sede di Padova, è emersa la necessità di fare in modo che la soluzione software fosse progettata in modo da ridurre l'invasività nei confronti del cliente: sostanzialmente i clienti devono aver installato nelle proprie macchine solamente un software VoIP con funzionalità standard (chiamata, chat, ...) e un browser web: questo per evitare l'addestramento dei dipendenti delle aziende clienti.

Modellazione delle funzionalità del sistema con use case diagram

Per identificare i requisiti funzionali del sistema, [Fow04] indica lo use case diagram come tecnica ideale per lo scopo: si basa sulla descrizione, in forma quasi narrativa, delle interazioni tipiche tra gli utenti e il sistema.
Lo use case diagram del caso di studio viene proposto in figura 5.1.

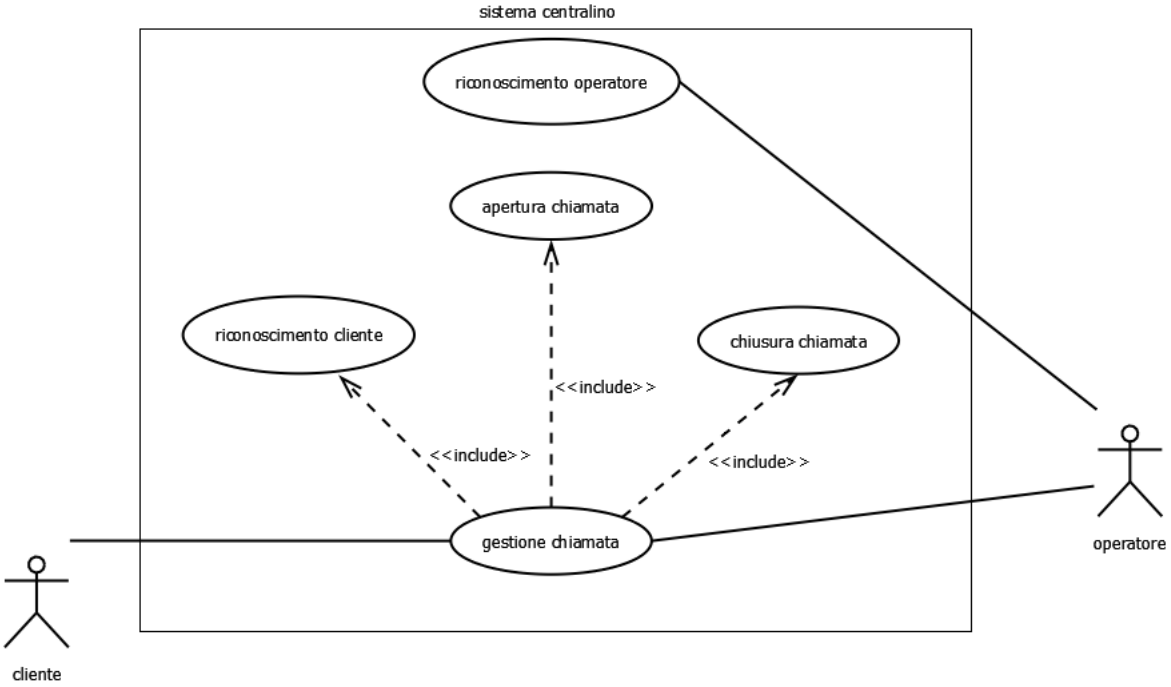


Figura 5.1: use case diagram del centralino VoIP

Nel seguito vengono specificati contenuti dei casi d'uso rappresentati nello use case diagram.

La fattibilità del Sistema

La fattibilità del sistema è data dalla stima delle risorse necessarie per portare a compimento lo sviluppo del software: questa operazione è stata effettuata in collaborazione con il responsabile Zucchetti della sede di Padova.

Le risorse software

Il client VoIP su cui è ricaduta la scelta come software di base per lo sviluppo del sistema è Skype, un'applicazione di facile installazione, di utilizzo intuitivo e molto diffuso fra i clienti dell'azienda. Per il cliente l'accesso al servizio è quindi subordinato alla registrazione dell'identificativo di Skype concordato nella fase di stesura del contratto.
Per il cliente è inoltre prevista la possibilità di accesso all'assistenza chiamando un numero PSTN: per questo scopo Skype fornisce l'interoperabilità con la rete PSTN attraverso il servizio SkypeIN, il quale fornisce un numero di telefonia fissa associato all'utente Skype del sistema.
Per la realizzazione delle componenti del sistema, Skype fonisce un meccanismo

Caso d'uso: **Riconoscimento Cliente**
Scenario principale 1 il cliente effettua una chiamata al sistema 2 il sistema riconosce il cliente 3 il sistema inoltra la chiamata verso un operatore disponibile
Estensioni: 2 a il sistema non riconosce il cliente. .1 il sistema chiude la chiamata .2 il sistema notifica al mittente che non può accedere al servizio di assistenza 3 a il sistema non rileva alcun operatore disponibile all'assistenza .1 il sistema chiude la chiamata .2 il sistema notifica al cliente la mancanza di operatori disponibili all'erogazione del servizio

Figura 5.2: forma testuale del caso d'uso *Riconoscimento Cliente*

Caso d'uso: **Riconoscimento Operatore**
Scenario principale 1 l'operatore invia un messaggio di autentificazione al sistema 2 il sistema effettua il riconoscimento dell'operatore 3 il sistema aggiunge alla propria lista l'operatore
Estensioni: 2 a il sistema non riconosce l'operatore .1 il sistema notifica al mittente che non può erogare alcun servizio

Figura 5.3: forma testuale del caso d'uso *Riconoscimento Operatore*

(API) di estensione delle funzionalità di base, per poter creare servizi specifici dell'applicazione che si vuole sviluppare.

Caso d'uso:**Chiusura Chiamata**
Scenario principale
1 l'operatore o il cliente conclude la chiamata
2 l'operatore notifica la propria disponibilità ad una nuova assistenza
3 il sistema aggiunge alla propria lista l'operatore

Figura 5.4: forma testuale del caso d'uso *Chiusura Chiamata*

Caso d'uso: **Apertura Chiamata**
Scenario principale
1 l'operatore notifica al sistema l'indisponibilità ad erogare nuove assistenze
2 l'operatore visualizza lo storico del cliente
3 l'operatore risponde al cliente
4 l'operatore inserisce la motivazione dell'assistenza

Figura 5.5: forma testuale del caso d'uso *Apertura Chiamata*

Caso d'uso: **Gestione Chiamata**
Scenario principale
1 il sistema effettua il riconoscimento dell'utente
2 l'operatore effettua l'apertura della chiamata
3 l'operatore eroga il servizio di assistenza al cliente
4 l'operatore o il cliente effettua la chiusura della chiamata

Figura 5.6: forma testuale del caso d'uso *Gestione Chiamata*

Per l'implementazione del sistema informativo di archiviazione dei dati dei clienti, degli operatori e delle assistenze viene impiegato *PostgreSQL*, un database relazionale open source.

Le risorse umane

Nessuna risorsa umana è stata impiegata oltre al sottoscritto: nelle fasi di progettazione e sviluppo del progetto si è verificato solo quale sporadico colloquio col il responsabile aziendale e con un programmatore della sede principale situata a Lodi.

Le risorse ambientali

In questa sezione vengono specificati i componenti hardware e software su cui si appoggia il sistema. Sia per l'hardware che per il software non è stato necessario l'acquisto di risorse specifiche. I due vincoli imposti dal sistema progettato sono:

- il supporto dello stack protocollare TCP/IP da parte dell' architettura hardware per rendere possibile lo scambio di pacchetti UDP e TCP fra le applicazione di Skype.
- i sistemi operativi supportino l'applicazione di Skype; poichè tutti i PC dell'azienda erano dotati di sistemi operativi Windows, per i quali è implementata una versione di Skype, non è stata necessaria alcuna installazione aggiuntiva.

La stima dei costi - benefici

Nella seguente lista vengono specificate le voci di costo economico per l'hardware e il software impiegato:

- costo per hardware e sistemi operativi: nessun costo è stato posto in carico all'azienda per l'acquisto di harware e sistemi operativi specifici per il progetto in esame.
- costi per l'utilizzo di Skype: il software VoIP di Skype è freeware, quindi di libero utilizzo così come le API pubbliche e il wrapper Skype4Java. L'unico costo riguarda l'acquisto del servizio di SkypeIN per l'acquisizione di un numero di telefonia PSTN.
- costi per tool di sviluppo: l'ambiente (IDE) *Eclipse*, utilizzato per sviluppare le applicazioni è a costo zero essendo opensource
- costi per l'utilizzo del DMBS PostgresSQL: ricadendo nell'ambito dell'opensource il suo contributo alla voce costi è nulla.

L'investimento per la realizzazione del sistema prevede quindi il costo del solo servizio SkypeIN come costo di installazione, al quale devono essere aggiunti i costi di manutenzione del software.

I ritorni previsti sono espressi in termini di benefici per i clienti e economici per l'azienda. Per i clienti si prevede un aumento della soddisfazione in quanto il servizio di assistenza diverrà più accessibile e più efficiente rispetto al vecchio servizio.
Per l'azienda si prevede una riduzione dei tempi di assistenza per ogni cliente, da cui consegue un aumento del rendimento degli operatori e quindi un ritorno economico.

La stima della durata temporale del progetto

Non è stata effettuata una stima della vita del progetto. La considerazione del responsabile Zucchetti è che le operazioni effettuate da questo sistema sono le classiche operazioni di un centralino telefonico. Ciò che è cambiato è la tecnologia impiegata: si suppone quindi che, finché non apparirà una nuova tecnologia in grado di fornire un servizio innovativo rispetto al VoIP, questo sistema continuerà ad essere operante con le opportuna evoluzioni.

5.2 L'architettura del Sistema

La disposizione dei componenti del sistema

Le applicazioni impiegate per rispondere ai requisiti funzionali specificati nella sezione precedente sono:

- i client di Skype
- le applicazioni esterne *ServerSkype* e *ClientSkype*
- il DBMS PostPostgreSQL

Una nota importante riguarda la modalità di accesso al servizio: per il cliente sarà possibile contattare il centralino utilizzando la funzione *Skype Me!* [skm]. Questa funzionalità consente di inserire all'interno delle pagine dell'applicazione web dei prodotti aziendali un pulsante che consente di contattare il client controllato da *ServerSkype*: l'unico vincolo imposto al cliente per l'accesso a questa funzionalità è ovviamente quello di avere Skype installato sulla propria macchina.

Per documentare la distribuzione fisica delle parti del software in esecuzione sulle macchine fisiche, [Fow04] consiglia l'impiego del *diagramma di deployment*: nella figura sottostante viene impiegato questo diagramma per illustrare la disposizione dei moduli che compongono il sistema.

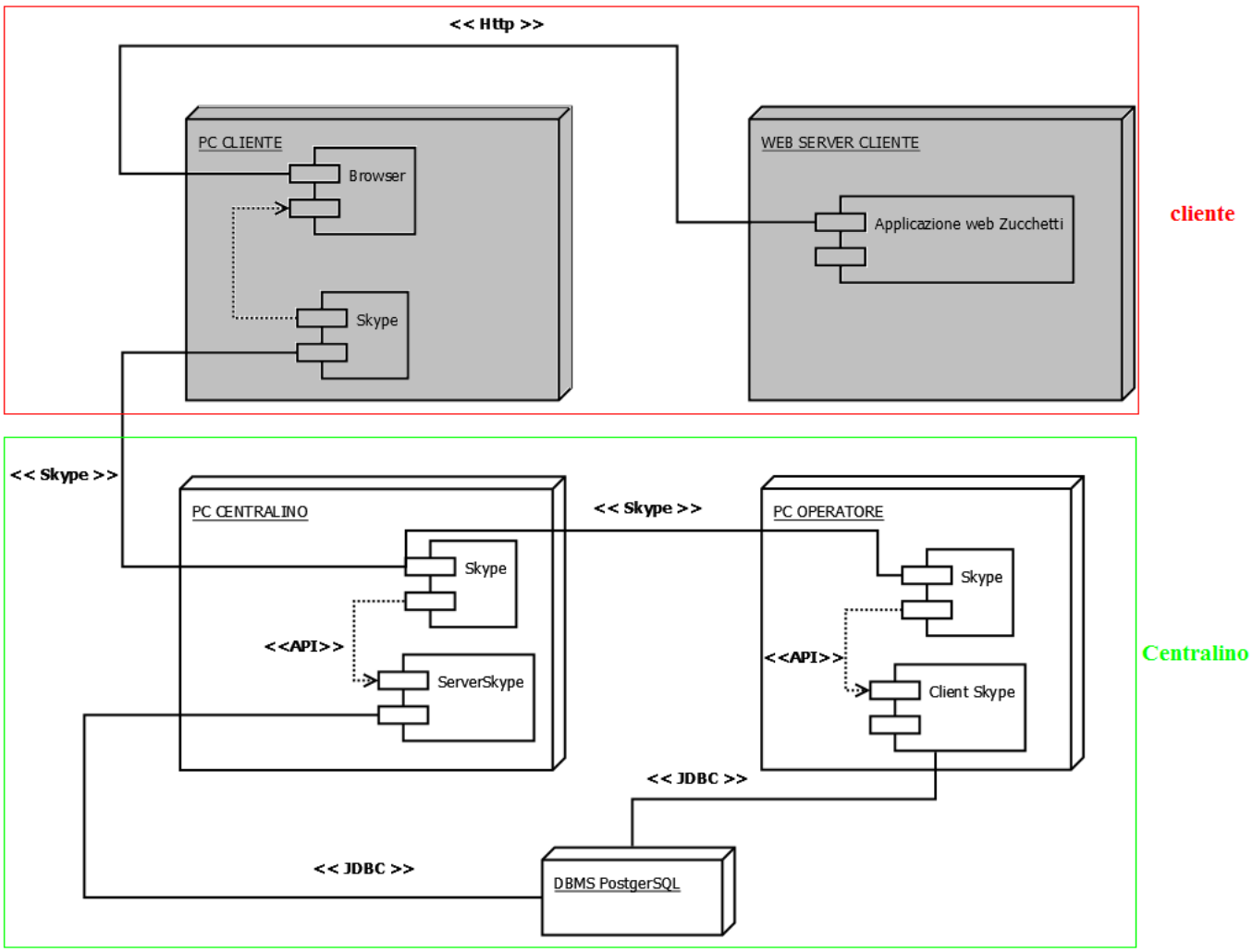


Figura 5.7: Diagramma di Deployment del sistema

Nella prossima sezione verrà presentata l'architettura a livelli del sistema sviluppato per rispondere alle funzionalità individuate durante la fase di analisi dei requisiti.

I livelli dell'applicazione

L'applicazione progettata si presenta con un'architettura logica a tre livelli:

- il livello di logica dei *Dati*
- il livello di logica dell'*Applicazione*
- il livello di logica di *Presentazione*

5.2.1 Lo strato di logica dei dati

In questa sezione viene presentata la modellazione dei dati riguardanti il sistema informativo del sistema. Il primo passo per compiere la modellazione è stata la costruzione di un modello Entity Relationship (ER) per la rappresentazione concettuale dei dati ad un alto livello di astrazione: attraverso questa modellazione dei dati è stato possibile individuare le entità della realtà in esame e le relazioni che intercorrono fra di esse. Le entità descrivono le informazioni sui clienti(intesi come clienti aziendali), sui loro contatti (Skype o PSTN) e sugli operatori del gruppo di assistenza Zucchetti. Le relazioni fra le entità del dominio informativo vengono descritte da un insieme di associazioni: nel caso in esame, le associazioni prendono il nome di *possesso* e *assistenza*. L'associazione *possesso* descrive tutti i contatti con cui un cliente può accedere al servizio assistenziale, mentre *assistenza* descrive tutte le assistenze effettuate mettendo in relazione il cliente con l'operatore.

Attraverso questo modello è possibile ricavare informazioni utili per il sistema quali:

- l'ora e la data delle assistenze effettuate ad un cliente, con l'operatori che ha risposto alle richieste;
- gli operatori di un gruppo di assistenza;
- il cliente chiamante e l'azienda di cui è dipendente;

In figura 5.8 viene mostrato il modello ER del sistema informativo

Attraverso un processo di traduzione del modello ER, si è giunti al modello relazionale, un modello logico di rappresentazione dei dati facilmente implementabile su sistemi di gestione di basi di dati (DBMS, Data Base Management System)

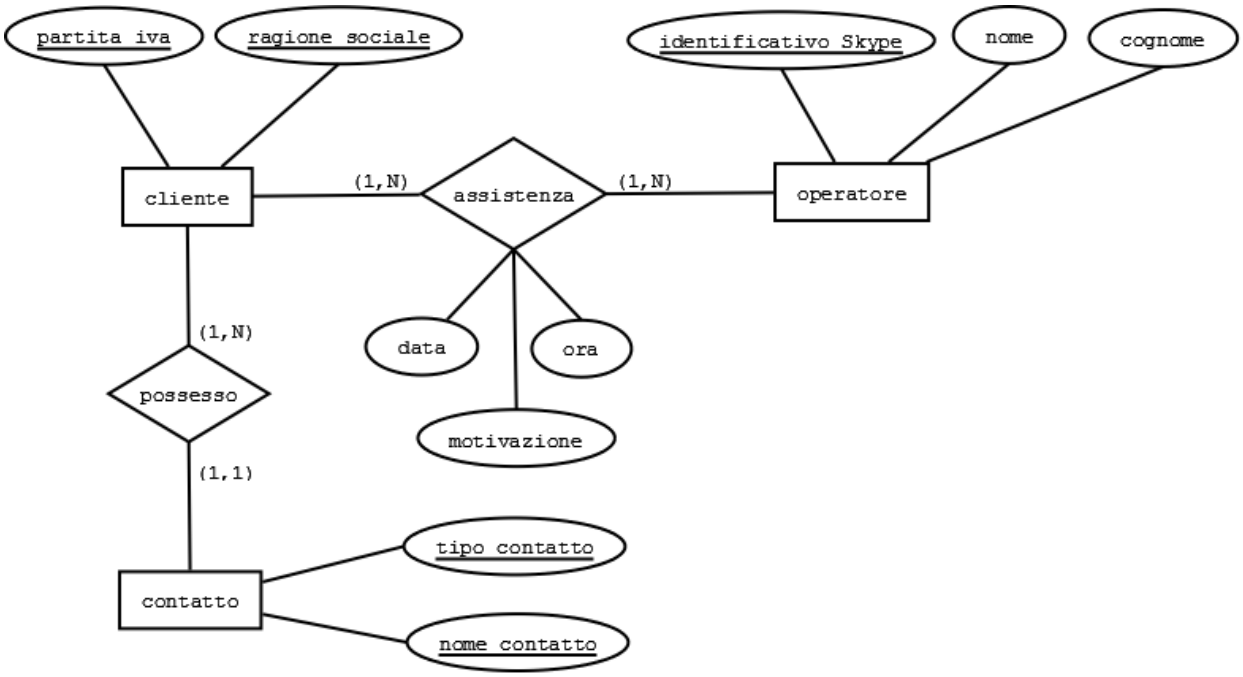


Figura 5.8: modello ER

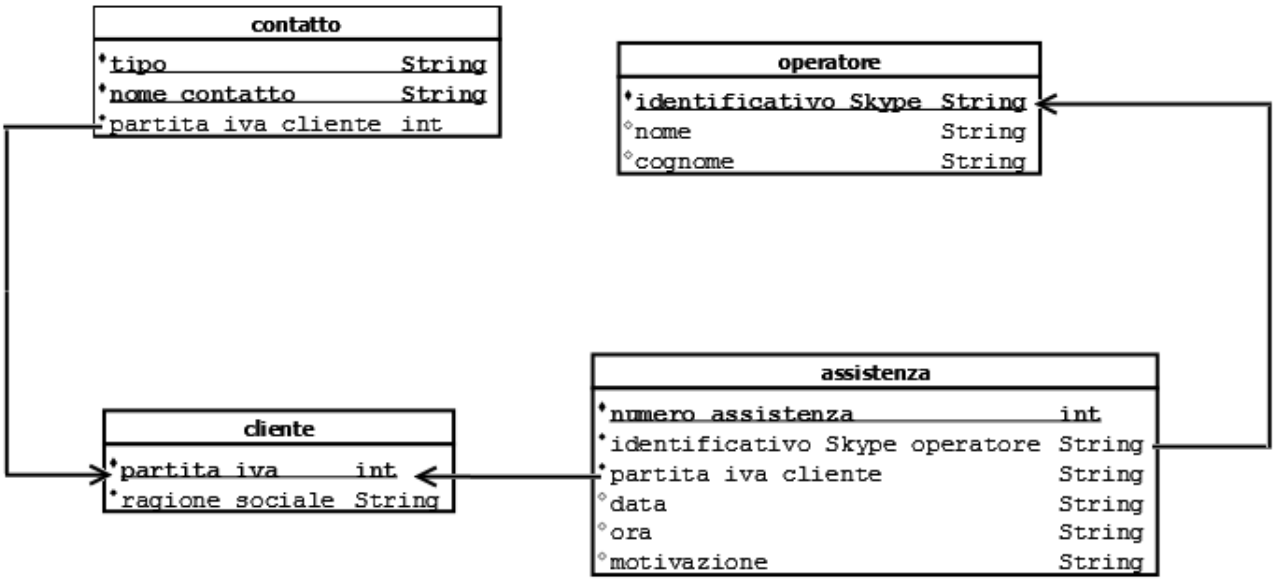


Figura 5.9: modello relazionale

5.2.2 Lo strato di logica dell'applicazione

L'applicazione SkypeServer

Questa applicazione è stata progettata con lo scopo di ricevere le chiamate dai clienti, di verificare la loro identità e di inoltrare le chiamate verso operatori disponibili; ha inoltre l'onere di verificare le identità degli operatori e, attraverso una lista, tenere traccia di quelli disponibili per l'erogazione del servizio di assistenza.
Per rappresentare i tipi di oggetti che fanno parte dell'applicazione SkypeServer e le varie tipologie di relazioni statiche fra di essi, viene impiegato il diagramma UML delle classi [Fow04] di figura 5.10.

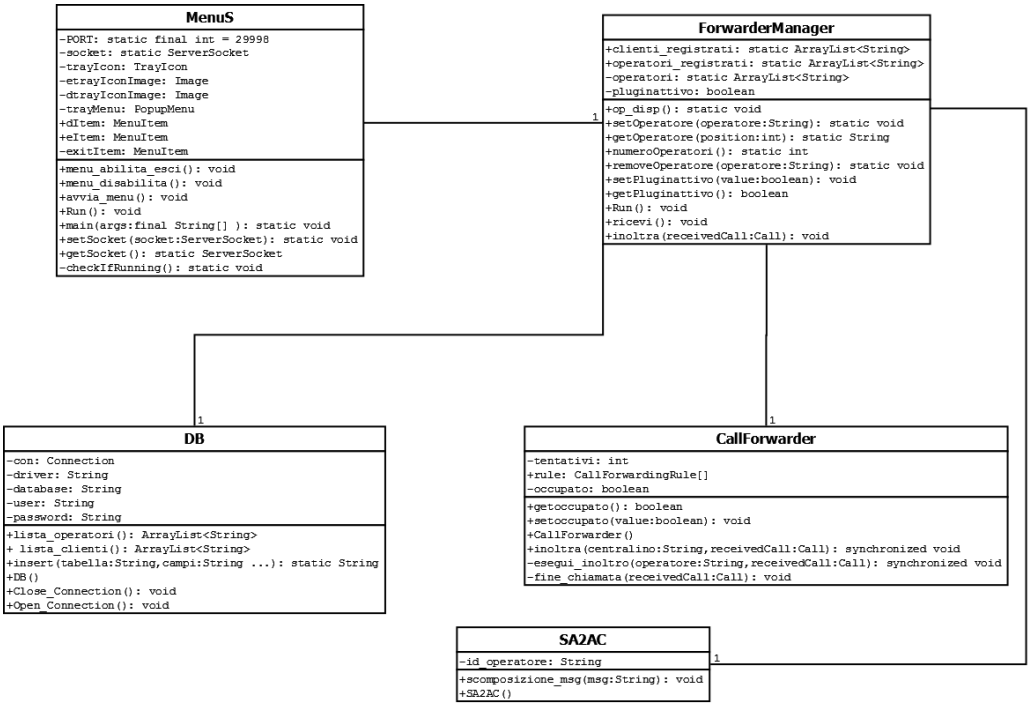


Figura 5.10: Class diagram di SkypeServer

Nel diagramma, la classe *MenuS* è impiegata per gestire l'aspetto del livello di presentazione, descritto nella sezione precedente: tramite trayIcon viene mostrato lo stato dell'applicazione (attiva/non attiva) e la possibilità, utilizzando il trayMenu, di poterlo modificare.

A MenuS viene associata la classe *ForwarederManager*, la quale coordina tutte le operazioni che l'applicazione deve svolgere e gestisce le liste degli operatori disponibili all'erogazione del servizio di assistenza.
Per verificare l'identità dei clienti e degli operatori, ForwarderManager utilizza la classe *DB*, la quale si occupa di effettuare le interrogazioni in linguaggio SQL al Data Base Management System (DBMS).

Per effettuare l'inoltro della chiamata, viene impiegeta la classe *CallForwarder*: l'istanza di questa classe viene richiamata in mutua esclusione. Il motivo dell'accesso in mutua esclusione è di consentire allo Skype Client di inoltrare la chiamata senza che altre chiamate in attesa possano essere indirizzate verso la stessa destinazione.

Infine, per comunicare con gli operatori tramite AP2AP, viene impiegata la classe *SA2AC*: quest'ultima provvede inoltre ad aggiornare la lista degli operatori disponibili, gestita dalla classe ForwarderManager.

L'applicazione SkypeClient

Quest'applicazione, illustrata attraverso il diagramma UML delle classi di figura 5.11, è stata progettata con lo scopo di gestire la visualizzazione delle assistenze precedenti del cliente chiamante e di inserimento della motivazione della nuova assistenza.
Come già avvenuto per SkyperServer, anche in questo caso i tipi di oggetti che fanno parte dell'applicazione SkypeClient e le varie tipologie di relazioni statiche fra di essi, viene impiegato il diagramma UML delle classi.

Similmente allo schema proposto in SkypeServer, in cui MenuS era stata realizzata allo scopo di gestire lo stato dell'applicazione sulla postazione in cui era attiva, in *SkypeClient* viene utilizzato *MenuC*. A MenuC viene associata la classe *SkypeOperatore*, la quale gestisce il comportamento dell'applicazione. SkypeOperatore, per inviare lo stato dell'applicazione (disponibile / non disponibile), effettua una comunicazione AP2AP utilizzando la classe *CA2AS*; CA2AS spedisce lo stato di SkypeClient che è memorizzato nella variabile booleana *disponibile*, la quale subisce variazioni in seguito ad eventi catturati dall'applicazione.
Vengono inviati messaggi all'applicazione SkypeServer contenenti lo stato dell'applicazione in seguito ai seguenti eventi:

- il thread *VIVO* (classe privata interna a CA2AS) invia un messaggio periodico di informazione dello stato;
- l'operatore risponde alla chiamata
- la chiamata si conclude
- l'operatore attiva l'applicazione SkypeClient
- l'operatore disattiva l'applicazione SkypeClient

La scelta di utilizzare un thread che periodicamente invia un messagggio a SkypeServer è giustificata dal fatto che la classe Stream, che si occupa della gestione del canale di comunicazione AP2AP, dealloca lo stream dopo un minuto dall'invio dell'ultimo messaggio: essendo l'allocazione di un nuovo stream piuttosto onerosa, si è deciso di perseguire la strada del mantenimeno dello stream.

Infine, la gestione dei dati del cliente, la visualizzazione delle assistenze e l'inserimento del motivo della nuova assistenza viene realizzato nelle classi *FrameV*, *MyFrame*, *GestoreEventiFramev* e *GestoreEventiFrame*.
La classe *FrameV*, che estende *Frame* del pacchetto standard Abstract Window Toolkit (AWT)[2], gestisce la finestra di visualizzazione dei dati del cliente; tutti gli eventi riguardanti la finestra sono gestiti dalla classe *GestoreEventiFramev*.

Dopo aver presentato le applicazioni che compongono il sistema, viene fornita una rappresentazione del comportamento del sistema per rispondere ai requisiti nel caso d'uso *Riconoscimento cliente*, individuato nella fase di analisi.

[2]package java.awt

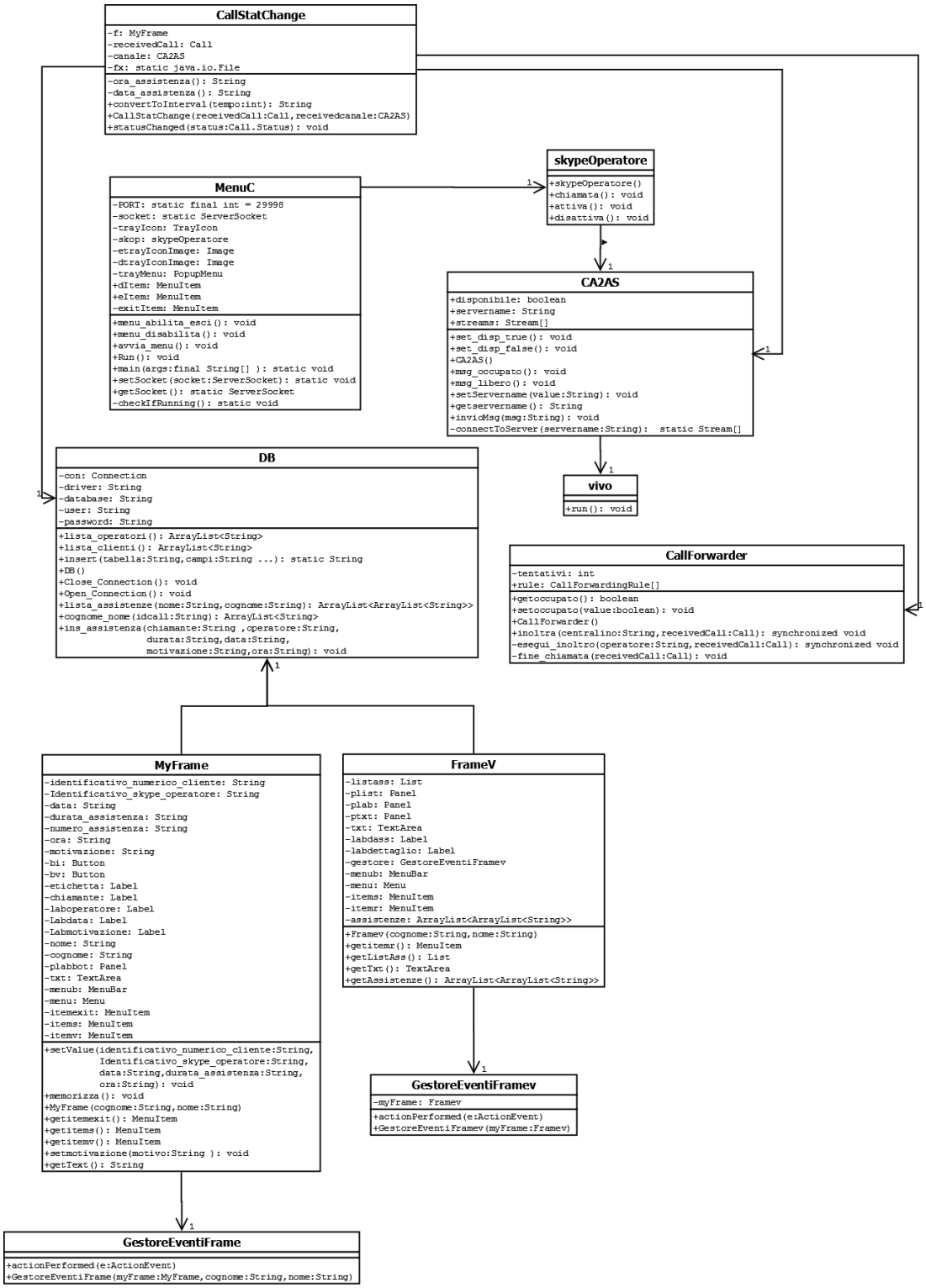


Figura 5.11: Class diagram di SkypeClient

Riconoscimento del cliente di inoltro della chiamata

Nel diagramma UML di sequenza sottostante viene mostrata la sequenza dei messaggi che vengono scambiati fra le applicazioni coinvolte nel progetto, per implementate la funzionalità di inoltro della chiamata.

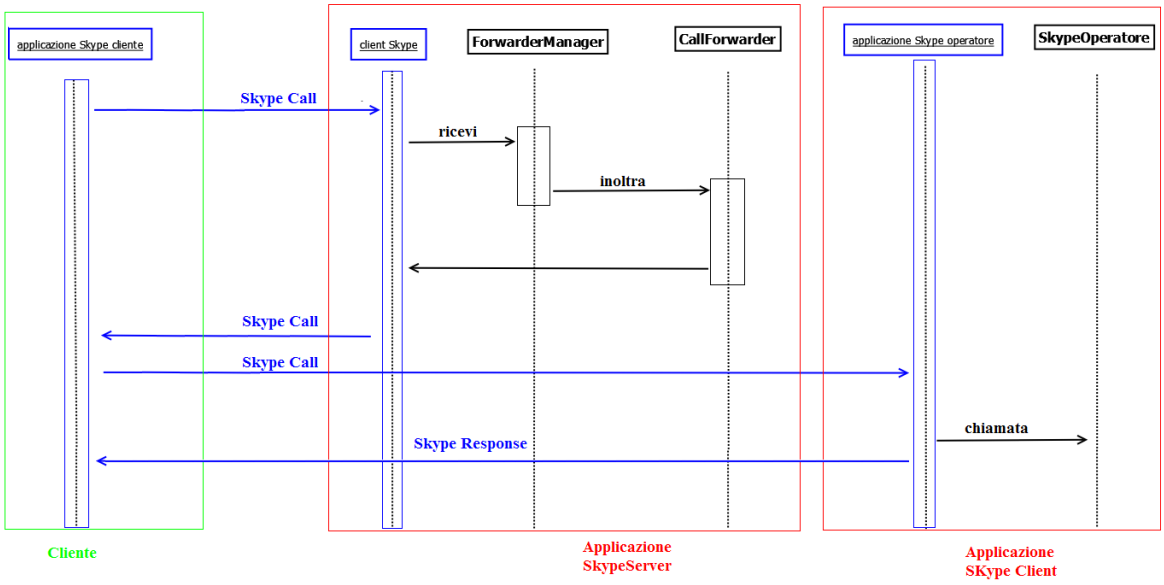


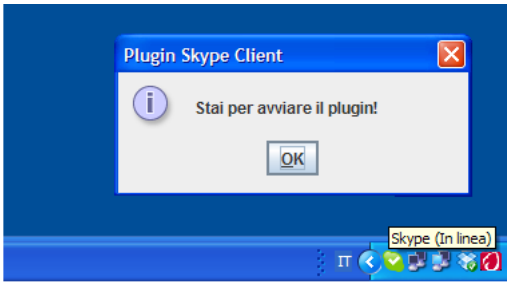
Figura 5.12: Class diagram di sequenza del caso d'uso *Riconoscimento cliente*

Lo scenario si apre con la chiamata allo Skype Client controllato dall'applicazione esterna *SkypeServer*. Quest'ultima riceve una notifica della ricezione della chiamata sul metodo *ricevi* della classe *ForwarderManager*. Questo metodo istanzia un oggetto di tipo *Call* associato alla chiamata di Skype, e lo passa alla funzione *inoltra* della classe *CallForwader*: quest'ultima comunicherà al'applicazione di Skype l'identificativo della destinazione dell'inoltro. A questo punto il client di Skype controllato dall'applicazione *SkypeServer* notificherà il nuovo identificativo di destinazione della chiamata, il quale provvederà ad effettuare la nuova chiamata automaticamente. Quando la chiamata raggiungerà il client di Skype controllato dall'applicazione SkypeClient, una notifica della ricezione di una chiamata verrà effettuata alla classe *SkypeOperatore*, la quale provvederà a visualizzare la finestra mostrata nella sezione di presentazione. A quel punto l'operatore risponde alla chiamata ed inzia l'erogazione del servizio assistenziale.

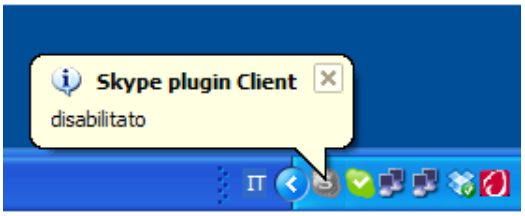
5.2.3 Lo strato di logica di presentazione

Il livello di presentazione dell'applicazione è molto semplice: viene qui presentata solamente l'interfaccia con l'utente di *ClientSkype* in quanto del tutto similare a quella di *ServerSkype*.

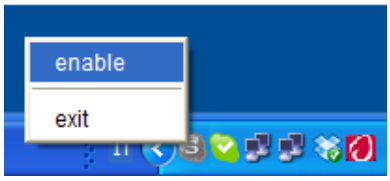
L'interfaccia con l'utente implementata in *ClientSkype* è un'icona nella Windows Tray che consente di abilitare l'applicazione di controllo dello Skype Client. Nelle figure 5.13(a), . . . , 5.13(h) vengono mostrati alcuni screenshot catturati durante l'utilizzo dell'applicazione: solamente quando l'icona visualizzata dall'applicazione è quella di figura 5.13(d) l'operatore è in grado di ricevere chiamate dal centralino.



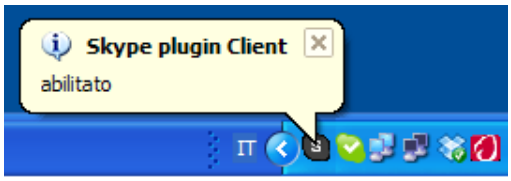
(a) avvio del plugin



(b) plugin nello stato "disattivo"

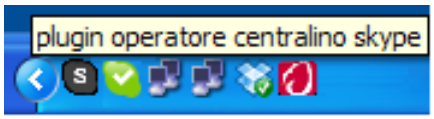


(c) attivazione del plugin

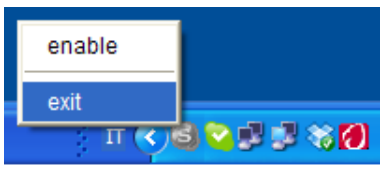


(d) plugin nello stato "attivo"

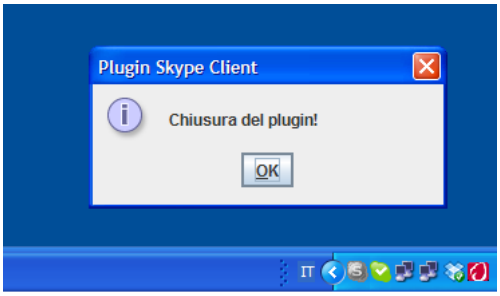
(e) attivazione del plugin



(f) nota di descrizione del plugin



(g) chiusura del plugin



(h) notifica chiusura del plugin

L'elemento addizionale di *ClientSkype* è l'interfaccia grafica per l'operatore. Quando l'operatore attiva la sua applicazione, è disponibile all'erogazione del servizio di assistenza: un esempio di assistenza è mostrato in figura 5.13.

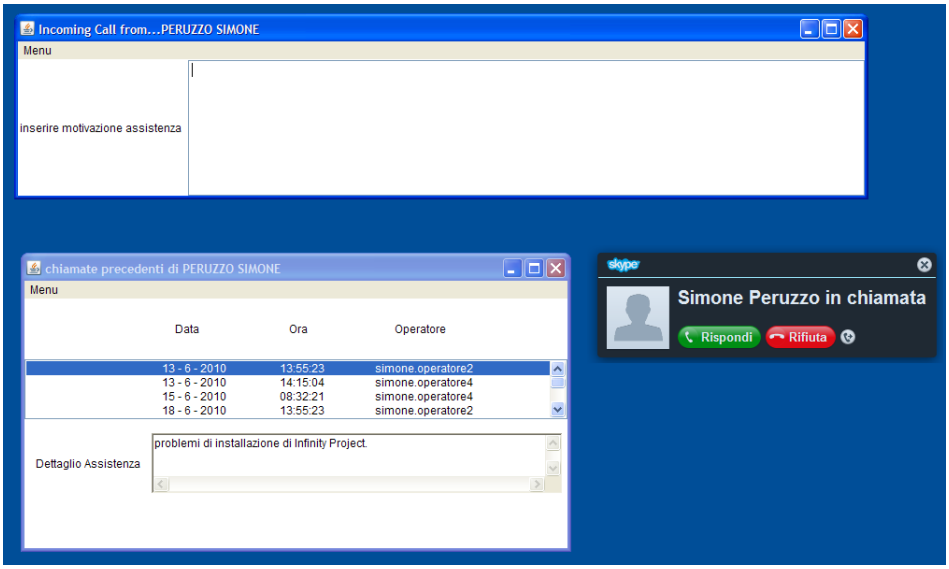


Figura 5.13: esempio di assistenza con la visualizzazione dello storico del cliente

Alla ricezione di una chiamata, si apre la finestra per l'inserimento della motivazione dell'assistenza: l'operatore può inoltre aprire una seconda finestra in cui viene visualizzato lo storico delle assistenze al cliente.

Per quanto concerne il cliente, nelle pagine web del prodotto aziendale viene presentato un pulsante "assistenza telefonica", che gli consente di contattare il centro assistenza attraverso la funzione *Skype Me!*.
Nella figura sottostante viene mostrato un esempio di chiamata da parte del cliente.

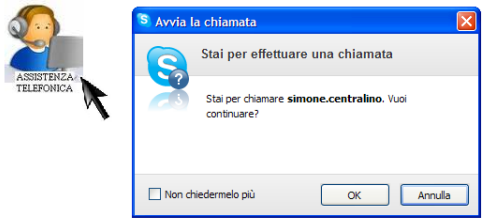


Figura 5.14: pulsante di chiamata all'assistenza

5.3 Sviluppi Futuri

Il casi di studio qui proposto è ancora in fase embrionale: diverse sono le proposte che vengono qui effettuate, sintetizzate in alcuni punti che verrano di seguito presentati. Gli sviluppi che verranno proposti sono classificati in:

- sviluppi dell'architettura del sistema, ed integrazione con funzionalità di condivisione dello schermo
- sviluppi del progetto per la gestione di più gruppi di assistenza specializzati e evoluzione verso un sistema di CRM(Customer Relationship Management)

Sviluppo dell'architettura

In questa senzione vengono proposte delle migliorie da apportare ai livelli dell'architettura sviluppata. Per quanto concerne lo strato di presentazione, la proposta di modifica riguarda le intefacce grafiche mostrare all'utente. Per il livello delle applicazioni e dei dati si propone l'aumento del grado di *fault - tolerance* che grava sulle architetture centralizzate, introducendo il concetto di *ridondanza dei componenti*.

Un ulteriore spunto di riflessione riguarda la possibilità di estendere il servizio di assistenza integrando le funzionalità VoIP a altre di condivisione di schermo remoto, in modo da poter effettuare un intervento sulla postazione del cliente senza dover ricorrere l'assistenza domiciliare.

Miglioramento del livello di presentazione per l'operatore

L'idea che si vuole proporre è quella di sostituire l'interfaccia grafica in cui vengono presentati i dati dei clienti una pagina web visualizzata su browser. Quando un operatore riceve una chiamata via Skype, automaticamente si apre una finestra del browser, il quale visualizza i dati del cliente su pagina web.

L'idea è quella di utilizzare un *servlet* in grado di costruire dinamicamente una pagina web contenente il risultato di una query ad un database accessibile utilizzando JDBC. I servlet sono moduli che estendono le funzionalità dei web server (es: Tomcat), secondo il meccanismo descritto in figura 5.16.

Nel caso di studio, l'applicazione *SkypeClient* dovrà inviare come parametro l'identificativo del cliente chiamante ed ottenere in risposta la pagina web contenente tutte le informazioni riguardanti lo storico delle assistenze al cliente stesso.

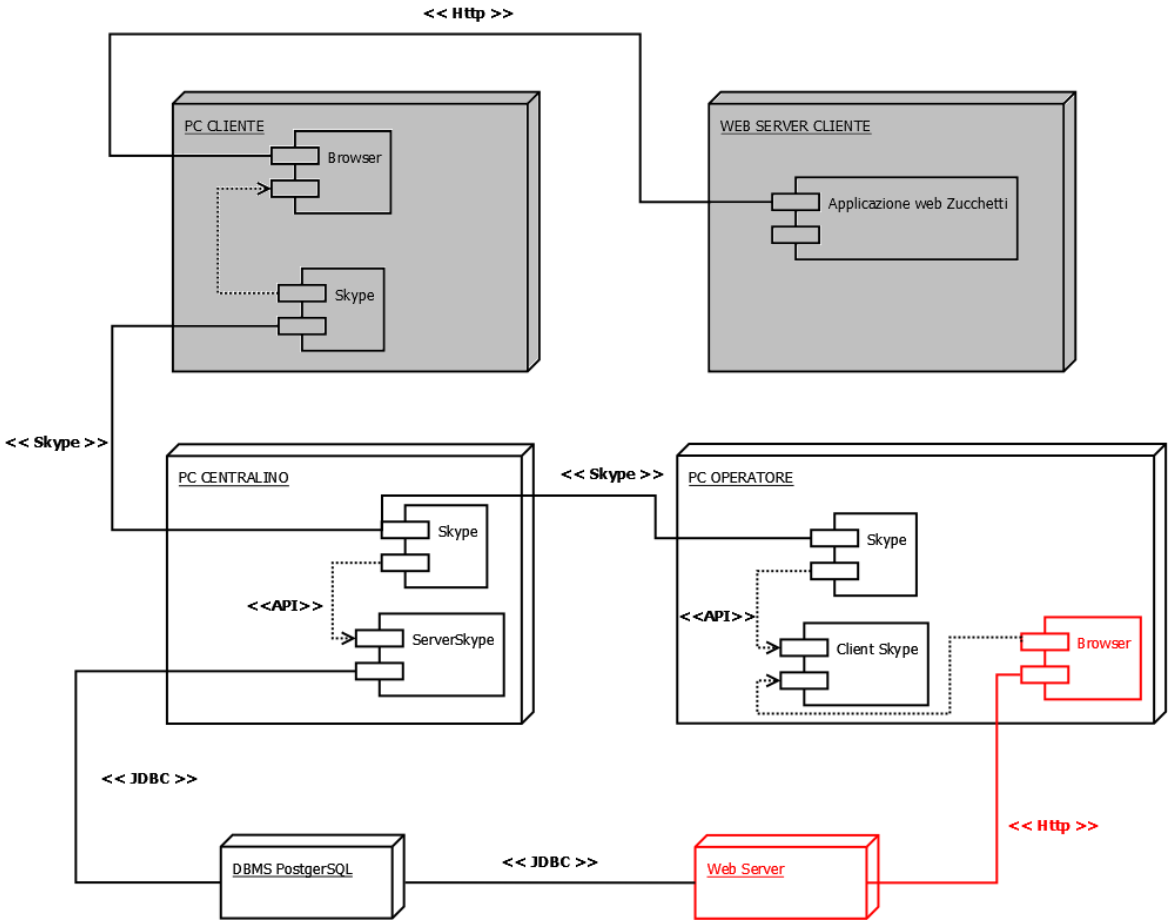


Figura 5.15: modello del sistema con web server

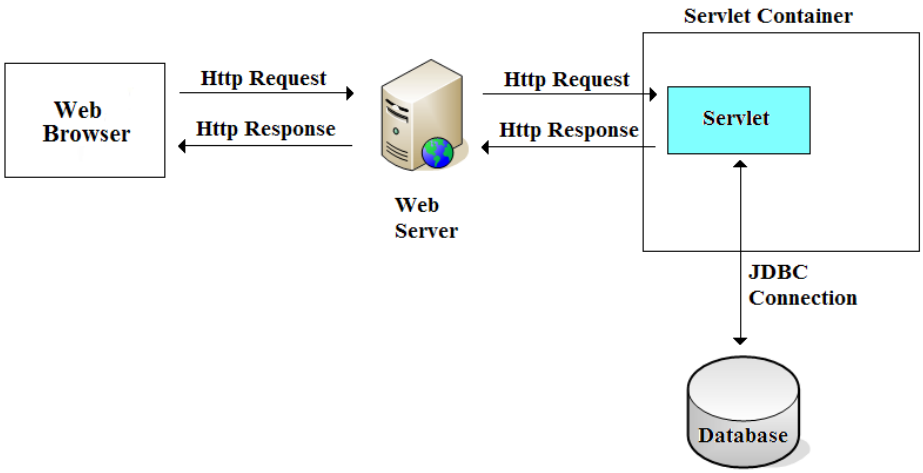


Figura 5.16: l'architettura di una servlet [HMD05]

Aumento del grado di *fault - tolerance*

Uno spunto di riflessione riguarda l'irrobustimento dell'architettura del sistema, che chiaramente è di tipo centralizzato: questo modello è aflitto dal problema noto come *single point of failure*. La tolleranza ai guasti è un grosso limite dei sistema client server: è sufficiente che il solo componente server subisca un guasto per compromettere il funzionamento dell'intero sistema. L'evoluzione in questo caso deve prevedere l'introduzione di un certo grado di *ridondanza* dell'applicazione server, per poter irrobustire il sistema e metterlo al riparo da singoli malfunzionamenti.
La soluzione sarebbe quella di istanziare su macchine diverse Skype Client con lo stesso identificativo previsto per il centralino del particolare gruppo. Secondo [MH06] è possibile inserire questa ridondanza: si necessita tuttavia di un approfondimento sul comportamento dei client alla ricezione di una chiamata. Dalle ricerche svolte in [MH06] si conclude che tutti i client vengono raggiunti dalla richiesta di chiamata ed iniziano squillare: a questo punto si rende necessario effettuare l'elezione del client che dovrebbe effettuare l'inoltro della chiamata. A tale proposto una soluzione potrebbero prevedere che tutti gli identificativi delle chiamate vengano ricercati su un archivio contenente gli ID delle chiamate inoltrate: se tale identificativo è presente, l'applicazione SkypeServer attiva su quel client conclude la chiamata, altrimenti memorizza l'identificativo nel database e inoltra la chiamata. In questo si potrebbe impedire l'indesiderato inoltro della stessa chiamata verso operatori diversi.

Anche il database può essere replicato per aumentare il grado di fault tolerance: a tale proposito PostgresSQL offre funzionalità per introdurre ridondanza. La più interessante, descritta in [dup], è denominata *Synchronous multimaster replication*: la replica sincrona permette di parallelizzare i server e farli funzionare come se fossero uno. Ovviamente, poiché le operazioni di lettura e scrittura sono eseguite su ogni server, sarà necessario più tempo per eseguirle. Il vantaggio è che il cluster appare come un solo computer.

Integrazione di funzionalità VoIP e condivisione desktop remoto

Una seconda considerazione di sviluppo riguarda invece la possibilità di integrare al servizio di chiamata, la condivisione dello schermo remoto in modo che l'operatore possa intervenire sulla postazione del cliente nel momento in cui avviene l'assistenza telefonica:si giungerebbe ad un'evoluzione del servizio nel quale l'operatore può svolgere la riparazione del problema dal proprio computer, evitando l'assistenza domiciliare.

Qui vengono citati diversi software che implementano il servizio di condivisione di desktop remoto:

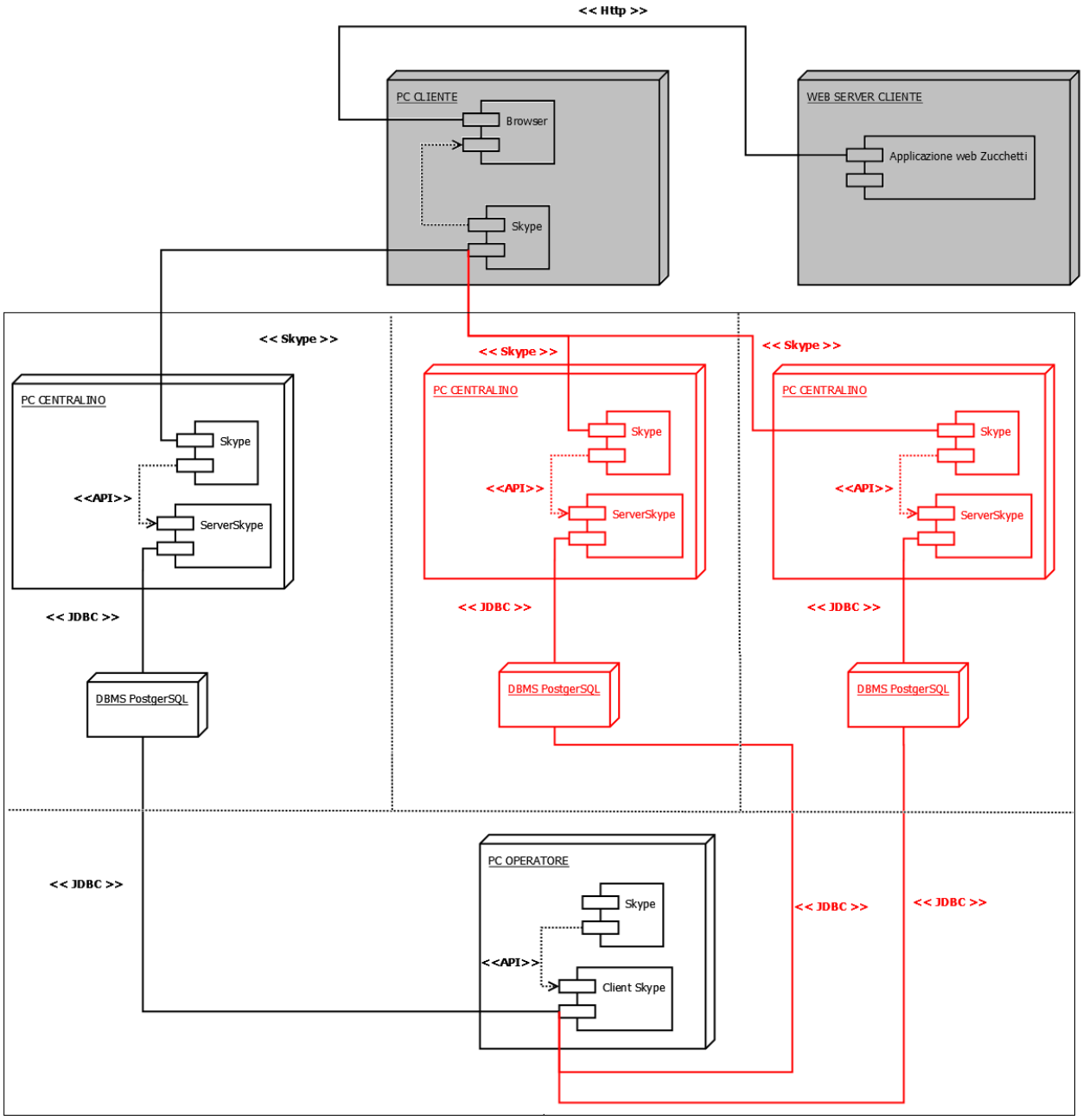


Figura 5.17: modello fisico esteso

- *Firnass* [fir], un'estensione che funziona su Internet di Java Remote Desktop (JRDesktop)
- *RealVNC* [rea]
- *TightVNC* [Tig]

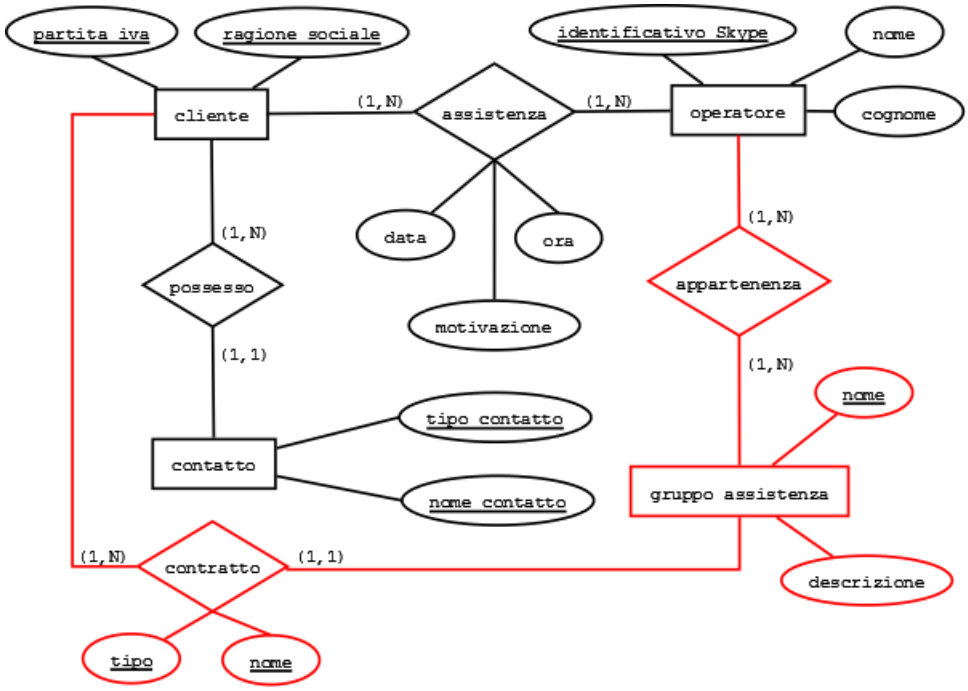


Figura 5.18: Estensione del modello ER di figura 5.8

Gestione dell'organizzazione di più centralini specializzati e evoluzione verso un sistema di CRM

Si parlerà della suddivisione del servizio di assistenza telefonica in gruppi indipendenti, ognuno dei quali specializzato per uno specifico prodotto aziendale. Si può prevedere che questo sistema possa evolvere inoltre da semplice *call - center* a *contact center* per poi inglobare il sistema in una strategia di Customer Relationship Management (CRM).

Gestione assistenza con gruppi dedicati

Questo spunto di sviluppo, pensato per ottimizzare l'organizzazione del reparto assistenziale, prevede la gestione indipendente dei diversi settori di assistenza dell'azienda: per ogni linea di prodotti sarebbe interessante poter associare un gruppo di assistenza dedicato. Questa evoluzione del sistema prevede una suddivisione degli operatori per gruppo, e ad ogni raggruppamento verrebbe associato un centralino VoIP apposito.

Il modello ER che viene proposto in figura 5.18 è un'evoluzione del modello proposto in figura 5.8.

Inclusione del sistema nel concetto di Customer Relationship Management

L'ultimo spunto di sviluppo riguarda l'evoluzione del sistema da semplice gestore delle chiamate a software di CRM. Il concetto di Customer Relationship Management descrive un insieme di azioni di marketing volte al mantenimento della clientela già esistente in un insieme di azioni volte al mantenimento della clientela esistente in un'ottica di aumento della soddisfazione della stessa che porta direttamente ad un aumento della fedeltà verso l'azienda. Un'impresa che adotta tale strategia non considera unicamente il cliente come unico elemento del mercato, ma anche l'ambiente che lo circonda, cercando di inserirsi in esso per creare rapporti più duraturi. Il CRM si occupa in particolar quindi di trovare nuovi clienti, aumentare i rapporti con i clienti già acquisiti e la fidelizzazione di quelli più promettenti, detti anche "clienti di primo piano".

Sostanzialmente, le funzionalità di CRM possono essere sintetizzate nei seguenti tre punti:

- l'acquisizione diretta di nuovi clienti;
- l'aumento delle relazioni con i clienti più importanti;
- la trasformazioni degli attuali clienti in procuratori, ossia consumatori che lodano l'azienda incoraggiando altre persone a rivolgersi alla stessa per i loro acquisti;

Molti sono gli strumenti utilizzabili da un sistema di CRM al fine di rafforzare il rapporto con azienda - cliente:

- call center
- forum di discussione;
- una banca dati contenente le Frequently Asked Questions (FAQ);
- un indirizzo e - mail a cui rivolgersi;
- servizi informativi forniti anche su altri strumenti (come SMS da inviare al cellulare del cliente);
- storia dei pagamenti effettuati dal cliente;

Capitolo 6

Conclusioni

In questa trattazione si è parlato del Voice over IP come un importante elemento della strategia Web 2.0. In particolare, questa tecnologia fornisce un importante supporto alla collaborazione fra gli utenti: grazie ad esso il grado di interattività della rete aumenta.

In ambito aziendale il VoIP si concretizza nell'idea di un centralino per l'assistenza alla clientela: questo strumento può essere seriamente preso in considerazione alla scopo di migliore il rapporto di collaborazione cliente - azienda. Il caso di studio presentato ne è un esempio sintomatico dell'interesse rivolto verso questa nuova opportunità: Zucchetti, un'importante un'azienda specializzata nello sviluppo di sistemi informatici, sta puntando la propria attenzione su questa tecnologia per migliorare le proprie relazioni con i clienti.

Il centralino progettato e prototipizzato ha fatto emergere nuove opportunità, che sono state ricondotte a due ambiti di evoluzione del sistema. La primo area di sviluppo è riguardante la necessità di aumento della tolleranza ai guasti e dell'interfaccia grafica del sistema. La seconda area dell'evoluzione del sistema nell'ottica dell'aumento dell'efficienza della collaborazione con il cliente. Questa può essere ottenuta attraverso la suddivisione del gruppo degli operatori in sottogruppi con competenze specifiche, ed pensando all' applicazione VoIP come elemento di un software di Customer Relationship Management.

Il concetto di di collaborazione sul Web trova applicabilità anche in ambito scientifico: nei portali di divulgazione scientifica può essere interessante cogliere l'opportunità offerta da un'applicazione VoIP tra gli utenti iscritti e gli autori delle pubblicazioni scientifiche. Lo scenario in cui il sistema VoIP potrebbe operare è semplice: se per ogni autore di pubblicazioni fosse disponibile un profilo in cui sia presente una descrizione del suo ambito di ricerca, queste informazioni potrebbero essere impiegate allo scopo di consentire all'utente l'individuazione dell'esperto dell'argomento che gli interessa, ed avere quindi la possibilità di un colloquio.

L'ambito di utilizzo del servizio in questione, denominato *Ask the Expert*, viene descritto brevemente di seguito.

L'utente registrato nel portale effettua una richiesta di chiamata premendo un pulsante presente nella sua pagina web; a quel punto si apre una finestra pop - up in cui inserire una descrizione dell'argomento di interesse. Questa descrizione viene quindi inviata assieme alla richiesta di comunicazione vocale. La descrizione viene confrontata con le informazioni dei profili degli autori delle pubblicazioni, utilizzando un algoritmo di *matching semantico*.
L'output di tale elaborazione è una lista di identificativi degli autori delle pubblicazioni ordinata per grado di rilevanza rispetto alla tematica di interesse dell'utente. A questo punto il sistema VoIP provvede all'inoltro della chiamata verso il primo operatore della lista che si sia disponibile.

Appendice A

Il protocollo IP

Un buon punto di partenza nella progettazione di un'Internetwork è la definizione del suo *modello di servizio*, cioè del servizio fra host che si vuole fornire. Il problema principale nella definizione di un modello di servizio per una internetwork è che è possibile fornire un certo servizio fra host soltanto se tale servizio può essere fornito da tutte le reti sottostanti.
La filosofia utilizzata nella definizione del modello di servizio di IP è stata quella di renderlo poco esigente da fare in modo che qualsiasi tecnologia di rete che possa comparire all'interno di una Internetwork sia in grado di fornire il servizio necessario.
Il modello di servizio di IP può essere immaginato come composto da due parti: uno schema di indirizzamento che fornisce il modo di identificare tutti gli host nella rete interconnessa, ed un modello datagram(cioè privo di connessione) per la consegna dei dati. Questo modello di servizio viene a volte chiamato *best effort*, in quanto IP non fornisce alcuna garanzia sulla consegna dei datagrammi, nonstante compia tutti gli sforzi possibili.

Consegna di datagrammi Il datagramma IP, descritto nell' RFC 791 [rfc81a], è un elemento fondamentale del protocollo IP. ogni datagramma porta con sé sufficienti informazioni perché la rete possa inoltrare il pacchetto fino alla sua destinazione corretta, senza bisogno di alcun meccanismo preventivo che segnali alla rete cosa fare quando arriva il pacchetto:semplicemente, lo inviate, e la rete fa del suo meglio per portarlo alla destinazione desiderata. L'espressione fa del suo meglio *(best effort)* significa che se qualcosa non funziona bene ed il pacchetto viene perduto, corrotto, mal indirizzato o per qualsiasi motivo non riesce a raggiungere la propria destinazione, la rete non fa nulla: ha fatto del suo meglio, e ciò è tutto quanto le era richiesto. Non compie alcun tentativo per recuperare la situazione: si tratta di un servizio che a volte viene chiamato *inaffidabile*.
La consegna best-effort non significa soltanto che i pacchetti possano essere smarriti. A volte vengono consegnati nell'ordine sbagliato e altre volte un pacchetto può essere consegnato più volte: i protocollli di livello superiore o le applicazioni

che operano al di sopra di IP devono essere consapevoli di tute queste possibilità di malfunzionamento.

Formato del pacchetto Un ruolo fondamentale nel modello di servizio IP è assunto dal tipo di pacchetti che possono essere trasportati. Il datagramma IP, come la maggior parte dei pacchetti, consiste di un'intestazione seguita da un certo numero di byte di dati.

Il formato dell'intestazione è mostrato in figura A.1.
Osservando ciascun campo dell'intestazione IP, si nota che il "semplice" modello di consegna best - effort e datagram possiede comunque caratteristiche complesse.
Il campo Version specifica la versione di IP, che attualmente è la versione 4 ed è nota con il nome *IPv4*[1] .
Il campo successivo, Hlen , indica la lunghezza dell'intestazione, misurata in parole di 32 bit. Quando non ci sono opzioni, l'intestazione è lunga 5 parole (20 byte).
Il campo Type Of Service (TOS) di 8 bit, consente il trattamento differenziato dei pacchetti in base alle necessità delle applicazioni. i 16 bit successivi dell' intestazione contengono la lunghezza (Length) del datagramma, compresa l'intestazione. Diversamente dal campo Hlen , il campo Length conta i Byte invece delle

[1]IPv4 è la versione di IP attualmente in uso, ed è così chiamata per distinguera da IPv6, che è la versione dell'Internet Protocol designata come successore di IPv4. Esso introduce alcuni nuovi servizi e semplifica molto la configurazione e la gestione delle reti IP

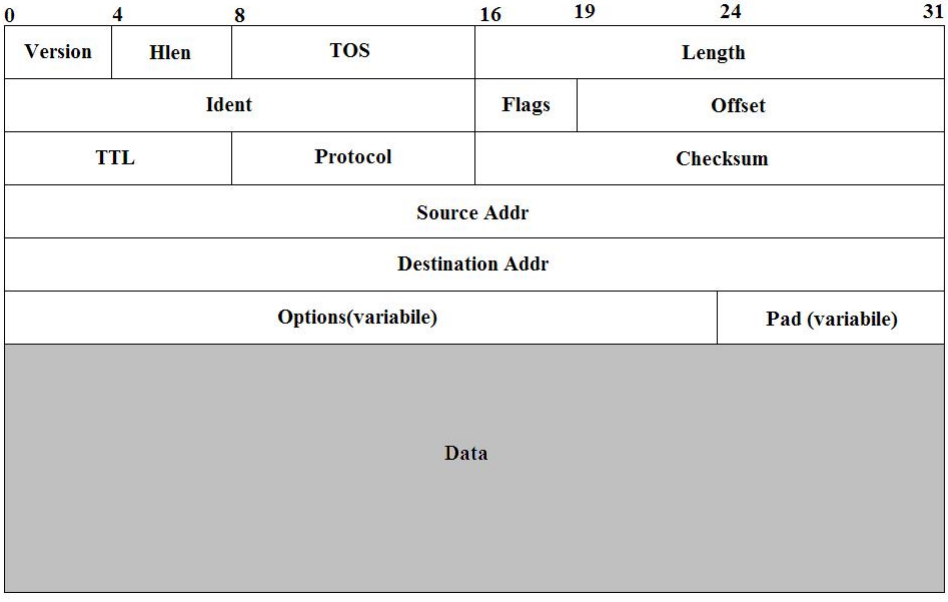


Figura A.1: Intestazione del pacchetto IPv4

parole, per cui la dimensione massima di un datagramma IP è 65535 byte. Puó verificarsi che la rete fisica sulla quale Internet Protocol (IP) opera non sia in grado di trasportare pacchetti così grandi : per tale ragione, IP fornisce supporto per un procedimento di frammentazione e ricostruzione dei pacchetti, in modo da rendere possibile il loro trasporto anche se reti dotatti di valori di Maximum Transmission Unit (MTU) bassi.
Spostandoci sulla terza parola dell'intestazione, il byte successivo é il campo Time To Live (TTL), il quale identifica il numero di hop consentiti al pacchetto sulla rete: l'obiettivo di questo campo è di catturare quei pacchetti che continuano a viaggiare in percorsi circolari ed eliminarli, per non consentire un consumo indefinito di risorse.
Il campo Protocol che si presenta successivamente è una chiave per il demultiplexing che identifica il protocollo di livello superiore a cui va consegnato il pacchetto IP. Esistono valori definiti per TCP(6) e UDP(17) e per molti altri protocolli che si possono trovare al di sopra di IP nel grafo dei protocolli.
Il successivo campo Checksum viene calcolato considerando l'intera intestazione IP come una sequenza di parole di 16 bit, sommandole utilizzando l'aritmetica in complemento a uno e prendendo il complemento a uno del risultato; se uno dei bit dell'intestazione viene corrotto durante il trasferimento, la somma di controllo non conterrà il valore corretto al momento della ricezione del pacchetto. Dato che un'intestazione corrotta può avere anche un errore nell'indirizzo di destinazione (di conseguenza il pacchetto potrebbe essere stato consegnato ad uno destinatario sbagliato), la cosa migliore da fare è ignorare qualsiasi pacchetto la cui somma di controllo risulti errata.
Gli ultimi due campi obbligatori dell'intestazione sono l'indirizzo del mittente (SourceAddr) e del destinarario (DestinationAddr) del pacchetto.
Infine, al termine dell'intestazione, si trovano un certo numero di opzioni, la cui presenza o assenza vengono dedotti dal campo che contiene la lunghezza dell'intestazione (Hlen).

Appendice B

Il protocollo UDP

Descritto nell' RCF 768 [Pos80], UDP è il più semplice protocollo di trasporto che si possa imamginare estende il servizio di consegna fra host svolto dalla rete sottostante in un servizio di comunicazione fra processi. Essendo molto probabile che vi siano molti processi in esecuzione in un host qualsiasi, il protocollo deve aggiungere un livello di demultiplexing, consentendo così la condivisione della rete fra più processi applicativi presenti in ciascun host. Oltre a questo requisito, questo protocollo di livello transport non fornisce alcuna altra funzionalità al servizio best - effort fornito dalla rete sottostante.
L'unica cosa interessante di questo protocollo è la forma di indirizzamento usato per identificare il processo che svolge attività in rete. L'approccio utilizzato da UDP prevede che i processi si identifichino l'un l'altro *indirettamente*,utilizzando un identificatore astratto, spesso chiamato *porta*: un processo sorgente invia un messaggio ad una porta e il processo destinatario riceve il messaggio da una porta. Nel prossimo paragrafo verranno descritti i campi che formano l'intestazione del pacchetto UDP.

L' intestazione del pacchetto UDP L'intestazione per un protocollo di trasporto (*end-to-end*) che realizzi questa funzione di demultiplexing contiene tipicamente un identificativo (porta) sia per il mittente (sorgente) sia per il destinatario (ricevente) del messaggio.

La figura B.1 mostra l'intestazione UDP. Si noti che il campo che ospita la porta UDP è lungo solamente 16 bit: ciò significa che esistono 64K possibili porte, ovviamente non sufficienti ad identificare tutti i processi di tutti gli host di Internet. Fortunatamente, le porte assumono significato solamente sul singolo host, non sull'intera Internet: in sostanza un processo viene effettivamente identificato da una porta su un certo host, cioè da una coppia $<$ porta , host $>$. Questa coppia costituisce la chiave di demultiplexing per il protocollo UDP.
È comunque improprio affermare che UDP svolga solamente il semplice demultiplexing dei messaggi destinati ai processi applicativi: assicura inoltre la corret-

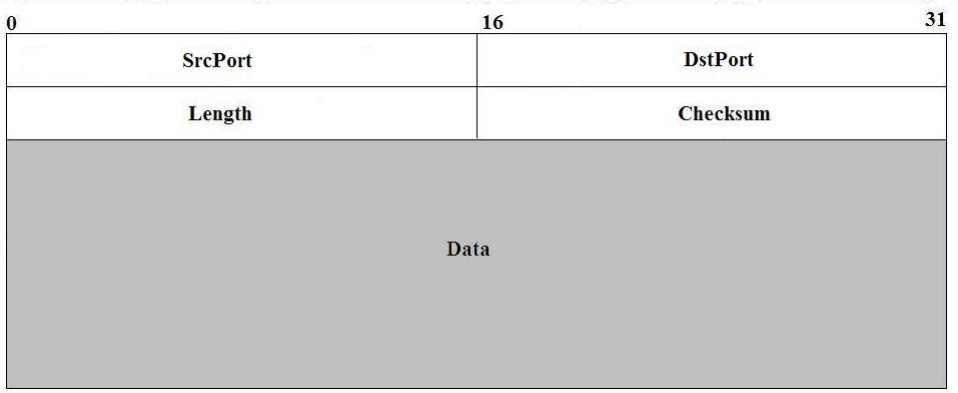


Figura B.1: Formato dell'intestazione UDP

tezza del messaggio per mezzo di una somma di controllo. Il protocollo UDP calcola la propria somma di controllo sull'intera intestazione UDP, sul contenuto del messaggio e su una parte chiamata *pseudointestazione (pseudoheader)* composta da tre campi dell'intestazione IP: il numero di protocollo, l'indirizzo IP del mittente e l'indirizzo IP del destinatario, e da un campo che contiene la lunghezza del pacchetto UDP. La motivazione che giustifica la presenza della pseudointestazione consiste nell'esigenza di verificare che il messaggio sia stato trasmesso tra i due estremi finali corretti.
Ad esempio, se l'indirizzo IP del destinatario fosse stato modificato durante il trasporto del pacchetto, provocando l'errata consegna del pacchetto stesso, questo fatto verrebbe scoperto dalle somme di controllo di UDP.

Appendice C

Il protocollo TCP

Diversamente da un semplice protocollo di demultiplexing come UDP, un protocollo di trasporto più sofisticato offre un servizio di flusso di byte affidabile e orientato alla connessione. Tale servizio di è dimostrato utile in un'ampia varietà di applicazioni, poiché libera l'applicazione dal compito di gestire i dati fuori ordine o mancanti.
TCP, descritto nell' RFC 793 [rfc81b], è probabilmente il protocollo più utilizzato di questo tipo: garantisce la consegna affidabile e in sequenza di un flusso di byte. È un protocollo *full - duplex*, per cui ciascuna connessione TCP fornisce una coppia di flussi di byte, consentendo al ricevitore di limitare la quantità di dati che il mittente può inviare in un certo istante.
Infine, come UDP, il protocollo TCP fornisce il supporto ad un meccanismo di demultiplexing che consente a più programmi applicativi in un host di sostenere simultaneamente una conversazione con le loro pari entità. Oltre alle caratteristiche appena elencate, il protocollo TCP realizza anche un meccanismo di controllo della congestione messo a punto con grande cura: l'idea su cui si basa questo meccanismo è quello do controllare la velocità con cui TCP invia i dati, non per evitare che il mittente sovraccarichi il ricevente, ma per impedire al mittente di sovraccaricare la rete.
Vengono ora illustrati la modalità di gestione del flusso di byte e il formato dell'intestazione del pacchetto TCP.

Gestione del flusso di byte Il protocollo TCP è *orientato ai byte*: ciò significa che il mittente invia byte in una connessione TCP e il destinatario riceve byte dalla connessione TCP. Anche se il termine *flusso di byte* descrive il servizio offerto da TCP ai processi applicativi, il protocollo TCP non trasmette singoli byte su Internet, ma memorizza un numero sufficiente di byte ricevuti dal processo applicativo sull'host mittente finché non ha riempito un pacchetto di dimensioni ragionevoli, dopodiché il processo ricevente leggerà da questo buffer quando vorrà. I pacchetti scambiati fra pari entità del protocollo TCP sono detti *segmenti*, perché ognuno di essi trasporta un segmento del flusso di byte.

Intestazione TCP

Ciascun segmento contiene l'intestazione schematicamente raffigurata in figura C.2, che verrà brevemente illustrata.

I campi SrcPort e DstPort identificano, rispettivamente, le porte della sorgente e della destinazione, similmente a quanto accade in UDP. Questi due campi si combinano con gli indirizzi di sorgente e destinazione per identificare univocamente ciascuna connessione TCP, cioè la chiave di demultiplexing di TCP è data dalla quaterna < SrcPort,SrcIPAddr,DstPort,DstIPAddr >.
Dal momento che le connessioni TCP nascono e muoiono, è possibile che fra due porte una connessione venga instaurata, venga usata per inviare e ricevere dati e poi venga chiusa, per poi creare una seconda connessione fra le medesime porte. In questa situazione si parla, a volte, di due diverse *incarnazioni* della stessa connessione.
I campi Acknowledgement, SequenceNum e AdvertisedWindow sono coinvolti nell'algoritmo *sliding window*[1] di TCP. Dato che TCP è un protocollo orientato al byte, ogni byte di dati ha un numero di sequenza: il campo SequenceNum contiene il numero di sequenza del primo byte di dati trasportato da quel segmento. I campi Acknowledgement e AdvertisedWindow contengono informazioni in merito al flusso di dati che scorre nell'altra direzione. Per semplificare la trattazione, si ignori il fatto che i dati possano fluire in entrambe le direzioni, focalizzando

[1] è un metodo di controllo del flusso di dati nelle reti di calcolatori, in particolare usato dal protocollo TCP

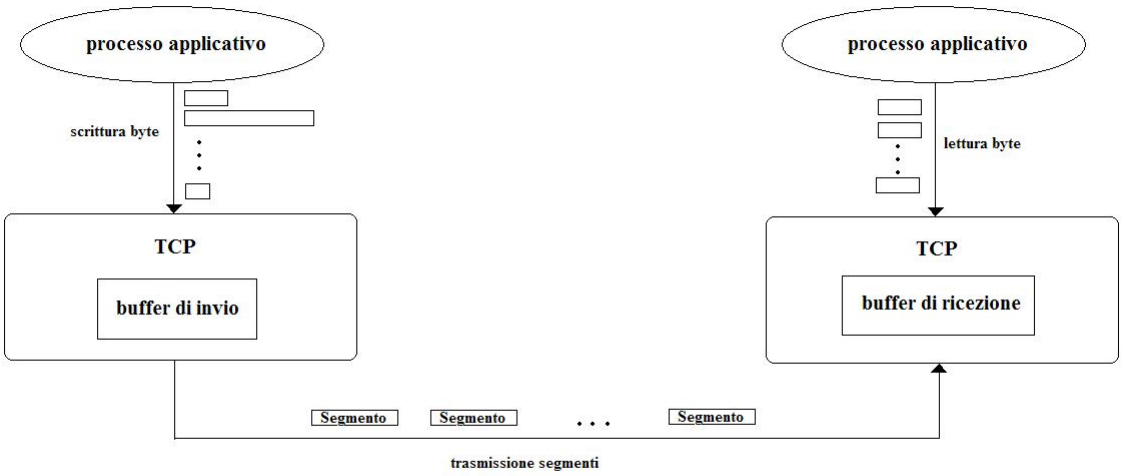


Figura C.1: gestione del flusso di byte del protocollo TCP

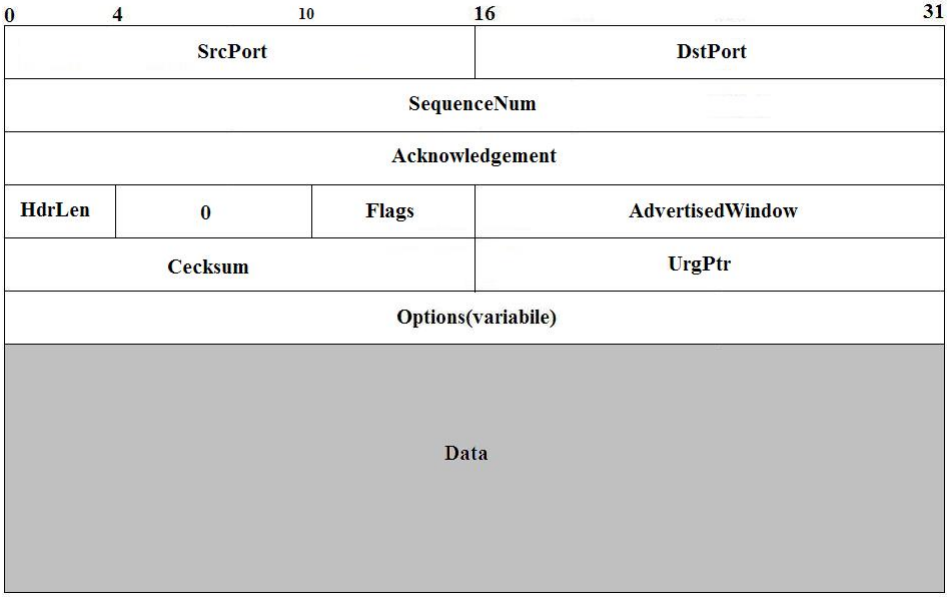


Figura C.2: formato dell'intestazione TCP

l'attenzione sui dati che fluiscono in una direzione con un certo valore di SequenceNum e sui valori Acknowledgement e AdvertisedWindow che vanno nella direzione opposta, come illustrato in C.3

Il campo di 6 bit, flags viene usato per trasportare informazioni di controllo fra pari entità in una connessione TCP. I segnali possibili sono SYN, FIN, RESET, PUSH, URG e ACK. I segnali SYN e FIN sono usati quando si instaura e si termina [2], rispettivamente, una connessione TCP. Il segnale ACK assume il valore 1 ogni volta che il campo Acknowledgement è valido, e, conseguentemente, il ricevitore vi deve porre attenzione. Il segnale URG al valore 1 significa che il segmento

[2]l'algoritmo usato dal protocollo TCP per instaurare e terminare una connessione è conosciuto con il nome *three-way handshake*

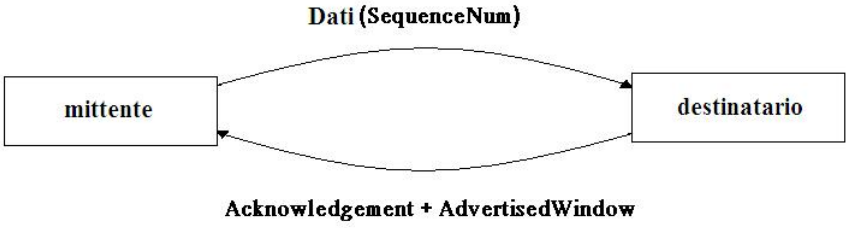


Figura C.3: diagramma semplificato del processo TCP, con i dati che fluiscono in una direzione e le conferme (ACK) nell'altra

contiene dati urgenti: in questo caso il campo UrgPtr indica dove iniziano i dati non urgenti contenuti nel segmento, mentre i dati urgenti sono contenuti all'inizio del corpo di ogni segmento, fino al valore di UrgPtr byte all'interno del segmento stesso. Il segnale PUSH indica che il mittente ha invocato l'operazioni "push", chiedendo all'entitá TCP ricevente di segnalare questo fatto al processo ricevente. Infine, il segnale RESET al valore 1 significa che il ricevente è stato confuso (ad esempio perché ha ricevuto un segmento che non si aspettava di ricevere), per cui vuole terminare bruscamente la connessione.
Per concludere, il campo Checksum viene usato esattamente come descritto per UDP: viene calcolato sull'intestazione TCP, i dati TCP e la pseudointestazione, che è composta dall'indirizzo del mittente, del destinatario e i campi che esprimono la lunghezza dell'intestazione IP. La somma di controllo è obbligatoria per il protocollo TCP, sia in IPv4 che IPv6 [3]. Inoltre, essendo l'intestazione TCP di lunghezza variabile (a causa delle intestazioni che possono essere allegate dopo i campi obbligatori), è presente un campo HdrLen che indica la lunghezza dell'intestazione in parole di 32 bit. Tale campo è anche noto con il nome di Offset, perché misura l'offset dall'inizio del pacchetto all'inizio dei dati.

[3] nuova versione del protocollo IP

Appendice D

RTP

Il protocollo RTP è stato definito dal Working Group Audio/Video Transport dell'IETF nell'RFC[1] 1889(gennaio 1996) e successivamente in RFC 3550(luglio 2003). Nonostante questo protocollo sembri, dal nome, essere un protocollo di trasporto, viene trattato come protocollo dello strato applicativo poiché contiene una notevole quantità di funzioni che sono specifiche per le applicazione multimediali.

RTP offre un servizio di instaurazione di comunicazioni real time appoggiandosi su UDP. Dovrà quindi sopperire ad alcune sue mancanze, in particolar modo:

1. *fornire informazioni sul payload type*, ossia informazioni sul formato dei dati inviati, ovvero la codifica effettuata sull'informazione annidata nel pacchetto
2. fornire il *sequence numbering*, ossia l'identificatore di sequenza per poter ristabilire l'ordine dei pacchetti ed eventualmente valutare quanti pacchetti sono andati perduti o scartati.
3. fornire *timestamping*, ossia la posizione temporale dei dati contenuti nel pacchetto
4. identificare le sorgenti per sincronizzare diversi flussi

Si noti che RTP non fornisce alcun meccanismo per garantire la consegna tempestiva dei pacchetti o fornire altre garanzie di qualità del servizio, ma, come già è stato detto, si affida ai servizi di livello sottostante per poter poter offrire il suo servizio. Non fornisce alcuna garanzia né alcun meccanismo di prevenzione

[1] è un documento che riporta informazioni o specifiche riguardanti nuove ricerche, innovazioni e metodologie dell'ambito informatico o, più nello specifico, di Internet. Attraverso l'Internet Society gli ingegneri o gli esperti informatici possono pubblicare dei memorandum, sottoforma di RFC, per esporre nuove idee o semplicemente delle informazioni che una volta vagliati dall'IETF possono diventare degli standard Internet.

contro la consegna fuori ordine dei pacchetti, né assume che la rete sottostante sia affidabile nell'offrire pacchetti in sequenza. RTP inserisce per l'appunto nei pacchetti il numero di sequenza, permettendo al ricevitore di ricostruire la sequenza dei pacchetti inviati dal mittente; il numero di sequenza trova una sua utilità anche nel fatto che può essere utilizzato per identificare la posizione corretta di un pacchetto, per esempio nel video decoding, senza necessariamente decodificare i pacchetti in sequenza.

IETF ha suddiviso lo standard RTP in due parti strettamente legate:

1. RTP , per trasportare dati con proprietà temporali
2. Real Time Control Protocol (RTCP) , per monitorare la qualità del servizio e fornire informazioni sui partecipanti di una sessione in atto.

Il progetto del protocollo RTP si basa un principio architetturale noto come Application Level Framing (ALF),che venne ideato da Clark Tennenhouse nel 1990 come nuova modalità per progettare protocolli per le applicazioni multimediali emergenti, dopo aver capito che tali nuove applicazioni sarebbero difficilmente state servite da protocolli esistenti, come TCP, e che quindi non sarebbero nemmeno state sevite adeguatamente da un qualsiasi tipo di protocollo che andasse bene per tutti. Il nucleo di questo principio consiste nel ritenere che un'applicazione sia la migliore candidata a comprendere le proprie necessità. RTP quindi si prefigge lo scopo di essere essere maleabile, per fornire alla specifica applicazione le particolari informazioni di cui essa ha bisogno.
Viene ora fornito una descrizione dell'intestazione dei protocolli RTP e RTCP

Formato del pacchetto RTP

La figura D.1 mostra il formato dell'intestazione usata dal protocollo RTP. I primi 12 byte sono sempre presenti, mentre gli identificativi delle sorgenti che forniscono dati (*contributing source identifiers*) vengono usati solo in alcune circostanze. Dopo questa intestazione ci possono essere estensioni dell'intestazione opzionali, come descritto in seguito. Infine, l'intestazione è seguita dal carico utile di RTP, il cui formato è determinato dall'applicazione. Lo spirito di questa è che debba contenere soltanto quei campi che saranno utilizzati con buona probabilità da molte applicazioni, perché qualsiasi cosa che sia molto specifica di una singola applicazione sarà veicolata in modo piú efficiente dal carico utile di RTP relativo a quell'applicazione. L'intestazione di RTP prevede inizialmente un campo **V** composto di due bit, il quale identifica la versione: attualmente la versione in uso è la 2. Il bit successivo **(P)** è il bit di padding, che viene impostato al valore 1 quando al carico utile di RTP vengono aggiunti, per qualche motivo, byte di *padding*, cosa che può avvenire per riempire un blocco di una determinata dimensione.In tal caso, all'intestazione del protocollo di livello inferiore(di solito UDP) viene comunicata la lughezza totale dell'intestazione RTP, dei relativi dati e dei byte di padding, e l'ultimo byte d padding contiene il conteggio di quanti siano i byte da ignorare.Il bit di estensione **X** viene utilizzato per indicare la presenza di un'intestazione di estensione, , che viene definita da una specifica applicazione e segue l'intestazione principale. Il bit X è seguito da un campo a 4 bit **CC** che conta il numero di sorgenti di dati (*contributing sources*), se ve ne sono all'interno dell'intestazione. Il bit **M** (*Marker bit*) viene utilizzato per identificare un frame, e assume significati diversi a seconda dell'applicazione(es "raffica di parole", "inquadratura video", ...). Di seguito troviamo il campo per il tipo di carico utile, (*payload type* (**PT**)), a 7 bit, che indica quale tipo di dati multimediali siano contenuti nel pacchetto. Il numero di sequenza (*sequence number*) viene utilizzato per consentire al ricevitore di un flusso RTP di rilevare pacchetti mancanti o fuori ordine. È da notare che il protocollo RTP non intraprende alcuna azione nel momento in cui rileva la perdita di un pacchetto : viene demandato all'applicazione il compito di decidere cosa fare, perchè è molto probabile che tale decisione sia strettamente correlata all'applicazione stessa. La funzione del campo *timestamp* è quella di consentire al ricevitore di riprodurre i campioni nel momento opportuno e di consentire la sincronizzazione

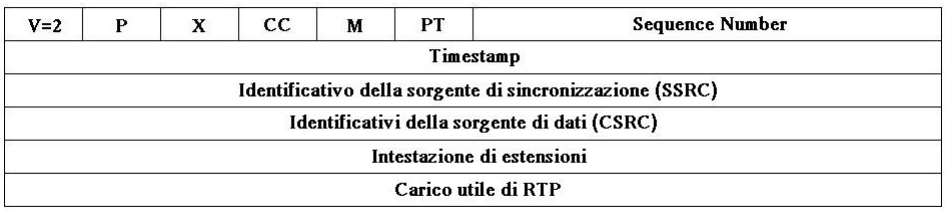


Figura D.1: Formato dell'intestazione di RTP

di diversi flussi multimediali. L'identificativo della sorgente di sincronizzazione (SSRC, *synchronized source*) è un numero a 32 bit che identifica univocamente una sorgente di un flusso RTP.In una conferenza multimediale, ogni fonte di flussi sceglia un valore di SSRC casuale, con il vincolo di dover risolve eventuali conflitti nell'improbabile caso che due fonti scelgano il medesimo valore. Rendendo l'identificativo della sorgente un'entità diversa dall'indirizzo di rete o di trasporto della sorgente stessa, il protocollo RTP si garantisce l'indipendenza dal protocollo di livello inferiore, oltre che consentiread un singolo nodo avente diversi sorgenti di flusso di distingurele tra di esse. Il campo per le sorgenti di dati (**CSRC**, *contributing source*) viene usato soltanto quando un certo numero di flussi RTP attraversano un mixer (*miscelatore*), che viene utilizzato per ridurre i requisiti di banda di una conferenza ricevendo dati da più sorgenti e trasmettendoli come un singolo flusso.

Il protocollo RTCP

L'RTP control protocol è basato sulla trasmissione periodica dei pacchetti di controllo a tutti i partecipanti ad una sessione. Il protocollo sottostante deve provvedere alla separazione dei pacchetti di dati e di controllo, usando porte con numeri diversi.
L'RTCP esegue quattro funzioni:

1. La funzione principale è provvedere al controllo della qualità della trasmissione dati. Questa funzione è permessa dai reports RTCP. È anche possibile, per una entità non direttamente coinvolta nella sessione, ricevere informazioni di feedback e di agire quindi da monitor di rete diagnosticando eventuali problemi di congestione.

2. RTCP trasporta un identifier di sorgente RTP chiamato canonical name (CNAME). Poiché l'SSRC identifier può cambiare in caso di problemi di conflitto o di restart di programmi, i riceventi richiedono il CNAME per ricondursi al partecipante. Essi richiedono il CNAME anche per riorganizzare sessioni multiple trasmesse da uno stesso mittente (per es.: video e audio).

3. Le prime due funzioni richiedono che tutti i partecipanti spediscano pacchetti RTCP. Per questo la frequenza alla quale vengono spediti i pacchetti, deve essere controllata in funzione del numero di partecipanti. Spedire pacchetti RTCP tutti a tutti, permette di risalire al numero di partecipanti, quindi di permettere il calcolo di tale frequenza (vedi RTCP Transmission Interval).

4. Una quarta funzione opzionale regola un controllo minimo sulle sessioni in atto, dove i partecipanti entrano ed escono senza parametri di negoziazione. Ciò serve a realizzare una conferenza con libero accesso. Le funzioni 1 - 3 sono obbligatorie, mentre la 4 può non essere implementata.

Il formato del pacchetto RTCP

RTCP si basa sulla trasmissione di cinque tipi di pacchetto :

- SR (sender report): per portare statistiche di ricezione e di trasmissione effettuate dai partecipanti trasmettono dati RTP.
- RR (receiver report): per le statistiche di ricezione di un partecipante che riceve solo dati RTP.
- SDES (source descriptor): per gli elementi di descrizione , incluso CNAME .
- BYE : indica la fine di una partecipazione.
- APP : pacchetto per funzioni specifiche di una applicazione.

Ogni pacchetto è costituito da una prima parte fissa, e da una seconda parte variabile. Per una descrizione dettagliata dei RTCP si prenda come riferimento l'RFC 3550. Più pacchetti RTCP possono essere inviati in uno stesso pacchetto UDP. Ogni pacchetto deve essere però processato dalla applicazione in modo indipendente l'uno dall'altro.

I seguenti vincoli sono però imposti :

- I pacchetti SR e RR devono essere spediti tutte le volte che i vincoli di banda lo consentono , cioè devono sempre esserci in un pacchetto composto.
- I nuovi ricevitori devono ottenere il CNAME delle sorgenti il più presto possibile per associare i diversi stream RTP.
- Il numero di pacchetti in un pacchetto composto, inizialmente deve essere limitato per aumentare la probabilità di un successo nella validazione dei pacchetti stessi.

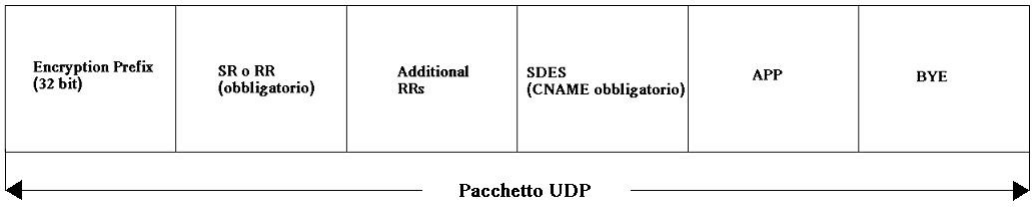


Figura D.2: Pacchetto RTCP composto

Il pacchetto composto, di almeno due pacchetti RTCP, avrà il seguente formato D.2:

- prefisso di criptazione: solo se il pacchetto è criptato, esso è preceduto da una quantità random di 32 bit ricalcolata per ogni pacchetto[2].
- SR o RR: il primo pacchetto deve essere uno fra questi due per facilitare la validazione.
- RRs addizionali: pacchetti che vanno aggiunti, se il numero di sorgenti di cui si sono fatte le statistiche supera 31.
- SDES: un pacchetto SDES contenente CNAME deve essere presente in ogni pacchetto composto. Altri SDES opzionali possono essere inclusi.
- BYE o APP: in particolare il pacchetto BYE deve essere l'ultimo.

[2]vedi RFC 1321 e RFC 2437 per chiarimenti sulla criptazione

Bibliografia

[Aha] Michel Glele Ahanhanzo. Siteskypeplug prototipo 1.0.0.1. 27 Aprile 2010.

[AST07] Maarten Van Steen Andrew S. Tanenbaum. *Sistemi distribuiti: prinicipi e paradigmi*. Pearson Education, 2007.

[Bia06] Jiang Bian. An analysis of the skype peer-to-peer voip system. *College of Computing Georgia Institute of Technology*, 2006.

[BL] Scott Lewis Bart Lamot, hisano. Skype4java (skype api for java). http://en.sourceforge.jp/projects/skype/releases/.

[BS04] Salman A. Baset and Henning Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. *Columbia University Technical Report CUCS-039-04*, September 2004.

[BS06a] Salman Baset and Henning Schulzrinne. An analysis of the skype peer to peer internet telephony protocol. http://www.powershow.com, April 2006.

[BS06b] Salman A. Baset and Henning G. Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. *Department of Computer Science Columbia University, New York NY 10027*, 2006.

[Cas01] Manuel Castells. *Internet Galaxy*. Oxford University Press, 2001.

[Com03] Douglas Comer. *Internet e reti di calcolatori*. Addison - Wesley, 2003.

[Dic07] Matt Dickman. what is an api? http://www.marketingprofs.com, 2007.

[DP01] J Davidson and J Peters. *Fondamenti di Voice Over IP*. Cisco Press, 2001.

[dup] Pgcluster round table. http://pgfoundry.org/docman.

[fir] Welcome to firnass: The java remote desktop control/sharing. http://www.firnass.com/.

[Fow04] Martin Fowler. *UML DISTILLED*. Paerson, 2004.

[Gou06] Michael Gough. *Skype me!: from single user to small enterprise and beyond*. Syngress, 2006.

[Har03] William C. Hardy. *VoIP Service Quality*. McGraw-Hill, 2003.

[Hin05] Taavet Hinrikus. Skype api. Technical report, Skype Techonolgies S.A., 2005.

[HMD05] Paul J. Deitel Harvey M. Deitel. *Java. Tecniche avanzate di programmazione*. APOGEO, 2005.

[HS03] R. Frederick V. Jacobson H. Schulzrinne, S. Casner. Rtp: A transport protocol for real-time applications. RFC 3550, July 2003.

[Jan03] Jan Janak. Sip introduction. Technical report, FhG FOKUS, 2003.

[JE03] Joy Rahman Juania Ellis, Charles Pursell. *The convergence of Voice, Video and Network Data*. Elsevier Science, 2003.

[K+01] Anshul Kundaje et al. Voice over ip. Master's thesis, Electrical Engineering Department V J Technological Institute University of Bombay, 2001.

[Kal] Peter Kalmstrom. Skype development techniques and tools. an overview. eBay Developers Program 2007. pubblicato in www.slideshare.net.

[Kri06] Gobala Krishnan. *Internet Telephony Secrets*. Liberty Straits Solutions, 2006.

[LF02] Onelio Bertazioli Lorenzo Favalli. *GSM-GPRS. Tecniche, architetture, procedure*. Hoepli Informatica, 2002.

[Lon05] Alessandro Longo. *Come si fa a telefonare gratis, o quasi, con Internet*. Tecniche Nuove, 2005.

[mat] Skype 4 java, ovvero come poter gestire skype all'interno delle nostre applicazioni e vivere felici ... http://roma.javaday.it/javaday2007.

[MH06] Tomás Plch Ondrej Serý Petr Tuma Martin Hamrle, Tomás Klacko. P2 skype demo: How to interact with skype. Technical report, Distributed Systems Research Group, Department of Software Engineering Faculty of Mathematics and Physics, Charles University, October 2006.

[Par06] Maurizio Parrino. *Voice Over IP.Guida Completa*. Apogeo, 2006.

[PD08] Larry L. Peterson and Bruce S. Davie. *Reti di Calcolatori*. Apogeo, 2008.

[Pos80] J. Postel. User datagram protocol. RFC 768, August 1980.

[rea] Realvnc - vnc remote control software. http://www.realvnc.com/.

[rfc81a] Internet protocol, darpa internet program, protocol specification. RFC 791, September 1981.

[rfc81b] Transmission control protocol , darpa internet program, protocol specification. RFC: 793, September 1981.

[Sch] Jürgen Schmidt. The hole trick, how skype & co. get round firewalls. December 2006.

[ska] Public api reference. http://developer.skype.com/accessories.

[skl10] It administratorsguide. Technical report, Skype Limited, 2010.

[skm] Crea un pulsante skype. http://www.skype.com/intl/it/tell-a-friend/get-a-skype-button/.

[Tan04] Andrew S. Tanenbaum. *Reti di Calcolatori*. Paerson, 2004.

[Tig] Tightvnc software free, lightweight, fast and reliable remote control software. http://www.tightvnc.com/.

[zuc] Zucchetti le soluzioni che creano successo. http://www.zucchetti.it.