Università degli studi di Bologna

FACOLTA' DI INGEGNERIA

Corso di Laurea in Ingegneria Informatica
Reti di Calcolatori L-A

Infrastruttura per la gestione di mobilità dei dispositivi basata su SIP

Tesi di Laurea di:
Elisabetta Visciotti

Relatore:
Chiar.mo Prof. Ing. Antonio Corradi
Correlatore:
Dott. Ing. Luca Foschini

ANNO ACCADEMICO 2004/2005

SESSIONE II

Sommario

PAROLE CHIAVE

Wireless Internet

Mobilità Terminale

Servizi Multimediali

Handoff

Protocollo SIP

Introduzione

Gli ultimi anni sono stati caratterizzati da un enorme sviluppo della rete Internet e dei servizi che essa mette a disposizione. La continua evoluzione nel campo delle reti di calcolatori ha portato alla nascita di nuove tecnologie che garantiscono conduttività e accesso ai servizi da qualunque posto ed in qualsiasi momento: le reti senza fili (wireless).

La rete senza fili è una rete in cui due o più terminali possono comunicare tra loro e con la rete Internet senza il bisogno di cavi di connessione. Al contrario, i terminali mobili comunicano via radio utilizzando un'antenna collegata alla rete fissa come punto di accesso (Access Point, AP) ad Internet.

L'introduzione delle reti senza fili ha quindi favorito la mobilità dei terminali, ossia la possibilità per un utilizzatore di potersi spostare con il proprio terminale da un punto di accesso ad un altro; tale processo è solitamente definito handoff (o handover). Tale scenario pone diverse sfide sia nella realizzazione dei servizi che nella progettazione dell'infrastruttura di supporto per l'erogazione dei servizi stessi. In particolare, l'infrastruttura deve garantire la continuità di servizio in modo che l'utente si possa muovere senza che la propria sessione di lavoro subisca interruzioni.

La continuità di sessione diventa poi fondamentale in un particolare ambito applicativo: l'erogazione di servizi multimediali. Tali servizi sono, infatti, molto sensibili a perdite e ritardi nell'arrivo dei dati, che possono compromettere la fruizione del servizio stesso fino a portare alla terminazione della sessione. Il processo di handoff, ossia l'intervallo tra la disconnessione del terminale dall'AP di origine e quindi dalla rete e la riconnessione presso un nuovo AP e la rete, deve

essere il più veloce possibile; bisogna inoltre garantire che la sessione creata non termini e possibilmente evitare che i dati in arrivo dal server, durante la migrazione, vadano persi.

In questa tesi ci occuperemo della gestione della sessione durante il processo di handoff, al fine di garantire continuità nella comunicazione cercando di evitare la perdita dei dati in arrivo dal server durante lo spostamento del terminale da una locazione ad un'altra. In particolar modo, ci proponiamo di realizzare un protocollo per la gestione della sessione e implementarlo come estensione dello standard Session Initiation Protocol (SIP).

Il protocollo SIP sta acquisendo un'importanza notevole nel mondo wireless poiché ha diversi strumenti per la gestione della mobilità di sessione e della mobilità terminale. Inoltre, essendo un protocollo di livello applicativo offre la flessibilità necessaria allo sviluppo di soluzioni portabili a prescindere dalle diverse tecnologie di comunicazione.

La tesi è stata così organizzata: nel primo capitolo è stato mostrato lo scenario applicativo, sono state descritte le principali caratteristiche dei servizi multimediali e le problematiche legate all'erogazione di servizi multimediali in reti senza fili ed è stato introdotto l'obiettivo della tesi. Il concetto di rete wireless, una possibile classificazione di tali reti e un'introduzione al problema dell'handoff in reti 802.11 costituiscono il secondo capitolo. Nel terzo capitolo è stato descritto il protocollo di sessione SIP; sono state approfondite le parti poi utilizzate nella nostra soluzione: la struttura dei messaggi SIP, le funzionalità degli user agent e dei server, l'event framework, e il protocollo SDP. Nel quarto sono state evidenziate le diverse tipologie di mobilità supportate da SIP, con maggiore riferimento alla terminal mobility, di cui ci siamo poi occupati effettivamente, e le soluzioni esistenti per la gestione dell'handoff, discutendo i vantaggi introdotti e le carenze che esse hanno. Il quinto capitolo è dedicato all'analisi e alla progettazione della

soluzione sviluppata relativa al micro handoff, il sesto alla fase di implementazione e il settimo alla fase di testing.

CAPITOLO 1

Erogazione di servizi multimediali in ambiente wireless.

In questo capitolo sarà introdotto lo scenario in cui la tesi opera, saranno evidenziati i problemi derivanti dalla mobilità del terminale nell'utilizzo di servizi multimediali.

1.1 Servizi multimediali.

Gli anni recenti sono stati testimoni di una grandissima crescita nello sviluppo e nell'operatività di applicazioni funzionanti in rete che trasmettono e ricevono contenuti multimediali. Il termine multimediale indica la combinazione di due o più media quali audio e video, che devono essere riprodotti in sincrono e con continuità durante un certo intervallo di tempo. Esempi tipici di servizi multimediali sono [1]:

- **Video on Demand (VoD)**: l'utente richiede di visualizzare un filmato che è stato preregistrato e archiviato sul server. Il flusso dei dati è monodirezionale e va dalla macchina che possiede il filmato all'utente che lo richiede.
- **Streaming Live**: anche in questo caso il flusso di informazioni è monodirezionale ma invece di richiedere un filmato o un file audio in archivio, l'utente chiede di poter ricevere le immagini di un evento in diretta come una conferenza oppure un concerto.

- **Video conferenza o Voice Over IP**: permette agli utenti di usare audio/video per comunicare tra loro in tempo reale. All'audio interattivo in tempo reale ci si riferisce spesso come alla telefonia Internet, poiché dalla prospettiva dell'utente, è simile al tradizionale servizio di telefonia a commutazione di circuito. La telefonia Internet facilita la realizzazione di nuovi servizi che non sono facilmente supportati dalle reti a commutazione di circuito tradizionali, tra cui l'integrazione Web-telefono, la comunicazione di gruppo in tempo reale, il filtraggio del chiamante, …. Con il video interattivo in tempo reale, chiamato anche videoconferenza, gli individui comunicano visivamente oltre che verbalmente. Chiaramente in questo tipo di applicazioni il flusso delle informazioni è bidirezionale, cioè ogni utente trasmette e riceve informazioni potenzialmente verso e da tutti i partecipanti.

Le applicazioni multimediali sono altamente sensibili al ritardo da terminale a terminale (end-to-end delay) e alle variazioni del ritardo, infatti i pacchetti che subiscono un ritardo da mittente a destinatario superiore a poche centinaia di millisecondi sono sostanzialmente inutili.

Il protocollo dello strato di rete per l'Internet attuale fornisce un servizio best-effort a tutti i pacchetti che trasporta. Internet fa del suo meglio per muovere i dati dal sender al receiver il più velocemente possibile, ma il servizio best-effort non fa alcuna promessa circa il ritardo end-to-end per qualsiasi pacchetto. Né fa alcuna promessa circa le variazioni del ritardo di un pacchetto all'interno di un flusso di pacchetti. Poiché TCP e UDP funzionano su IP, nessuno di questi protocolli può fornire alcuna garanzia sul ritardo alle applicazioni che ad essi si rivolgono. Per questo motivo la qualità del servizio (QoS), caratterizzata da un insieme di parametri specifici riferiti al servizio di interesse, deve essere gestita sopra il livello di rete.

1.2 Servizi multimediali mobili.

La rete wireless, di cui si parlerà in maniera più approfondita nel successivo capitolo, è una rete senza fili, una rete in cui diversi dispositivi (dai PC ai cellulari) sono collegati tra loro e alla rete senza fili esterni. La comunicazione tra le diverse macchine e la rete avviene attraverso gli Access Point (punti di accesso alla rete), antenne capaci di ricevere e trasmettere segnali da e per l'utente mobile.

Attraverso una rete wireless è possibile offrire gli stessi servizi offerti da una rete fissa. Questo però non significa che la resa di tali servizi sia identica ed anzi alcuni di essi possono portare alla luce le differenze intrinseche tra le due tecnologie (capitolo 2).

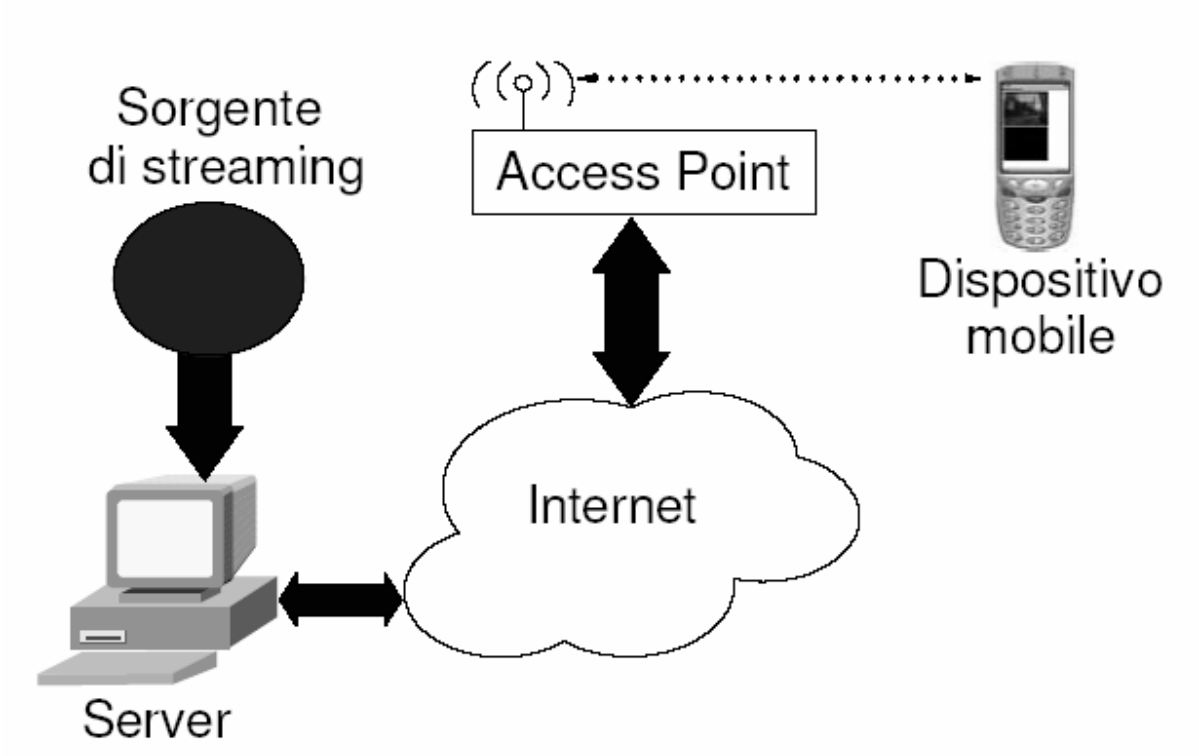

Figura 1.1 Erogazione di servizi multimediali verso dispositivi mobili.

Per i contenuti multimediali è necessario dunque utilizzare un tipo di servizio continuo, cioè un servizio che trasmetta un flusso di dati con una frequenza costante e adotti strategie volte a evitare interruzioni casuali di flusso sia per le reti fisse sia per le mobili. Con la tecnologia wireless ci sono diversi problemi da affrontare nell'erogazione e ricezione di contenuti multimediali, problemi che hanno come conseguenza ritardi nella ricezione, quindi non si riesce a garantire sempre servizi real-time, perdita di flusso e quindi rendono obsoleto il servizio che si sta utilizzando. Alcune di queste problematiche sono descritte nel seguito.

In base alla tecnologia wireless utilizzata abbiamo diverse velocità di trasmissione, Bluetooth ha una velocità teorica pari a 723 Kbps invece IEEE 802.11b 11Mbps. La limitazione di banda cui i dispositivi mobili sono solitamente soggetti è un fattore che condiziona molto le scelte progettuali [1]. Per ridurre in ogni caso la mole di dati per contenuti multimediali è necessario utilizzare codec opportuni per la compressione. Algoritmi di compressione molto complessi richiedono più computazione per la decodifica e ciò non sempre è accettabile per dispositivi mobili che hanno prestazioni limitate. Potrebbe essere richiesto un utilizzo eccessivo del processore, che potrebbe portare ad un uso eccessivo della batteria e problemi di surriscaldamento. Possiamo quindi affermare che gli strumenti wireless tendono ad essere più lenti di quelli connessi.

La limitazione di banda non è poi l'unico problema in cui si può incorrere nelle reti wireless, infatti esse sono soggette a continue variazioni di prestazioni, esistono molti elementi che possono incidere sulla qualità di trasmissione:

- Le condizioni atmosferiche.
- Le variazioni di potenza del segnale causate dallo spostamento in luoghi dove le trasmissioni radio arrivano con difficoltà.
- Lo spostamento da una zona coperta da un access point ad un'altra con diverso access point.

- La batteria, che determina più o meno potenza in ricezione e trasmissione.
- Ostacoli presenti nel percorso tra AP e utente mobile, cammini multipli.

Con la rete wireless, come sarà descritto nel capitolo seguente, si ha la mobilità dei dispositivi, ossia l'utente può muoversi con il proprio terminale da un punto di accesso (Access Point) alla rete ad un altro punto di accesso, noto come processo di handoff o handover. Quando un terminale mobile si allontana troppo dalla stazione base servente (la stazione base individua la cella elementare, garantisce la copertura radio in una certa porzione del territorio, un esempio è l'access point), avvicinandosi contemporaneamente ad una delle basi adiacenti, la qualità della comunicazione con la base servente tende a peggiorare, mentre una o più stazioni base adiacenti sono in grado di offrire una qualità di comunicazione migliore.

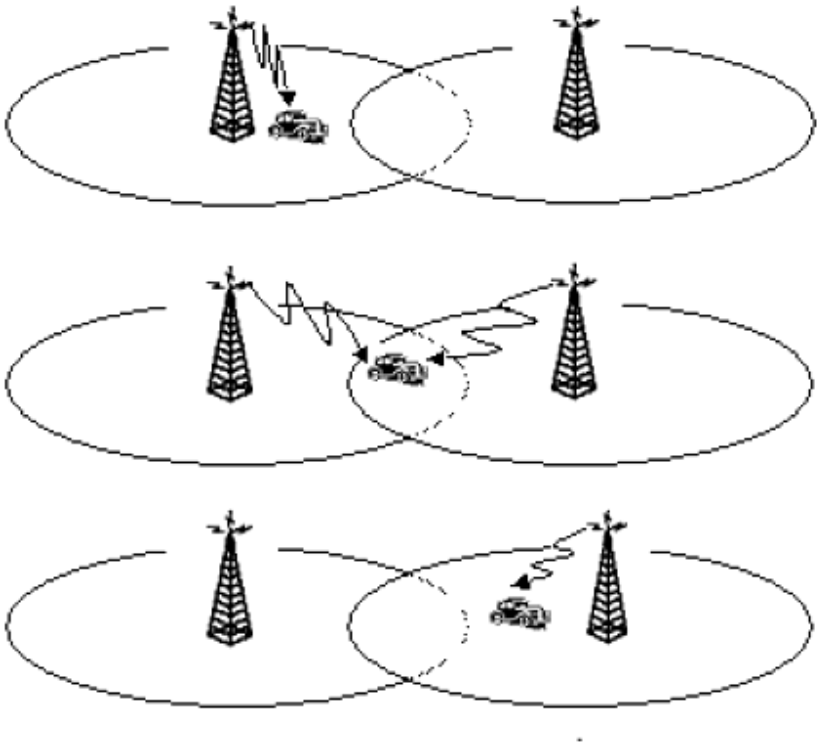

Figura 1.2 Handoff (handover).

L’handoff è, dunque, il passaggio della comunicazione tra mobile e rete dalla stazione base servente alla nuova base (Figura 1.2).

Quando si parla di mobilità utente e del conseguente processo di handoff emergono problemi riguardanti la ricezione di dati multimediali. Nel momento in cui l’utente mobile si disconnette dall’AP di origine di una certa località per riconnettersi ad un altro AP di un’altra località di destinazione, il flusso dei dati si interrompe, in quanto per un certo intervallo di tempo il terminale cliente non è connesso alla rete che sta fornendo il servizio. Si ha, dunque, un’interruzione nella visualizzazione/ascolto dei dati, la durata dell’interruzione è proporzionale al tempo di handoff.

Oltre al problema dell’interruzione nella comunicazione c’è anche il problema della perdita dei dati durante tale spostamento, infatti, il server, prima di essere avvisato della nuova locazione del cliente, continua ad inviare i dati nella postazione di origine, dati naturalmente che non saranno mai ricevuti dal terminale cliente.

Nel momento in cui il cliente si riconnette dovrà avvisare il server della sua nuova postazione e del flusso che si aspetta di ricevere, tale ripristino può avere tempi più o meno lunghi, i dati potrebbero, dunque, diventare obsoleti, soprattutto in quei servizi real-time, in cui è importante che tutto sia inviato e ricevuto in determinati istanti.

Sono state sviluppate molte soluzioni per cercare di ridurre tali problemi, alcune saranno esposte brevemente nel capitolo 4, soluzioni riguardanti i livelli più bassi dello stack OSI, altre invece i livelli applicativi.

1.3 Gestione della sessione.

Con il termine sessione viene indicato l'intervallo di tempo tra l'invio della prima richiesta da parte del client verso il server e la chiusura della comunicazione con quest'ultimo. L'invio del flusso multimediale verso il terminale client viene gestito dal livello di sessione, facendo riferimento allo stack OSI parliamo di livello 5. Ogni trasmissione di contenuti multimediali deve essere preceduta da una fase preliminare di comunicazione. In questa fase il client deve contattare il server e chiedere la sua disponibilità nel fornire il servizio richiesto, inoltre vengono decise le caratteristiche del servizio, ossia la qualità. Il client per poter usufruire del flusso richiesto deve identificarsi e "contrattare" con il server. Al termine di questa fase iniziale c'è l'invio del flusso multimediale richiesto.

Come già discusso, i flussi multimediali hanno la necessità di essere inviati con continuità e senza interruzione al fine di avere un servizio real-time e senza perdita di dati. Con l'utilizzo della rete wireless e con la conseguente mobilità dei terminali si ha il problema dell'handoff. Durante il processo di handoff il cliente si disconnette dalla rete per un certo intervallo di tempo, la disconnessione provoca un'interruzione del flusso e la perdita dei dati che in tale intervallo continuano ad essere inviati dal server prima di essere avvisato dello spostamento. E' necessario dunque che la sessione, nonostante l'interruzione, sia mantenuta, in maniera tale da essere ripristinata nel momento in cui il client si connette ad un altro AP, così da non dover iniziare di nuovo una sessione con il server, quindi si ha un handoff di sessione, infatti deve spostarsi da un AP ad un altro. E' necessario anche poter bufferizzare i dati in arrivo dal server durante la migrazione, in maniera tale da non dover ritrasmettere il flusso già inviato al client quando si riconnetterà. Per poter bufferizzare tali pacchetti è indispensabile introdurre tra il server e il client un'altra entità: il proxy.

Il proxy funge da intermediario tra il server e il client e ha come funzione quella di interfacciarsi con il server per ricevere il flusso e con il client per inviare tale flusso ricevuto, inoltre deve contenere un buffer per poter memorizzare i dati destinati al client durante l'handover per poi inviarli a quest'ultimo nel momento in cui il client si riconnetterà, in maniera tale da non appesantire il server ed evitare la perdita dei dati transienti.

1.4 Obiettivo della tesi.

L'obiettivo della tesi è quello di velocizzare il processo di handoff (o handover) così da non rendere obsoleti i dati in arrivo al client, evitare interruzioni nella comunicazione e la perdita di dati durante la migrazione.

Dal nostro punto di vista, per alcuni domini applicativi (soft real-time, multimediale), l'handoff dovrebbe essere gestito a livello applicativo al fine di rendere la soluzione indipendente dalla tecnologia utilizzata dai livelli sottostanti e soprattutto realizzare soluzioni specifiche per il servizio che si sta fornendo o utilizzando.

La soluzione che proponiamo riguarda il livello applicativo e nello specifico il livello di sessione. Attraverso il livello di sessione due host possono instaurare un dialogo per raggiungere un "*accordo*", ad esempio decidere la qualità e le caratteristiche dei servizi da fornire/ricevere. Solo dopo lo scambio di informazioni iniziali è possibile inviare e ricevere dati multimediali. Nella nostra soluzione abbiamo utilizzato il protocollo SIP e le sue estensioni sia per controllare lo scambio di messaggi iniziali tra i due host, al fine di instaurare una sessione, sia per gestire il processo di handoff e la conseguente perdita di dati (capitoli 5 e 6). Inoltre abbiamo presupposto che il proxy tra client e server abbia un buffer per la ricezione dei dati inviati dal server al client durante l'handover. Come si vedrà nel capitolo di

analisi e progettazione, è stato introdotto anche un buffer sul client così da fornire dati all'utilizzatore finale durante tale disconnessione, in modo da non provocare o almeno ridurre le interruzioni nella visualizzazione.

E' importante che l'utilizzatore finale non si accorga del cambio di AP e quindi continui a usufruire del servizio richiesto, la migrazione deve essere del tutto trasparente e veloce.

CAPITOLO 2

Ambiente Wi-Fi.

In questo capitolo saranno introdotti il concetto di rete wireless e gli standard 802.11X relativi alle reti WLAN e all'handoff.

2.1 Vantaggi della rete wireless.

Wireless letteralmente significa "*senza fili*" e si riferisce ad un sistema di comunicazione in cui i segnali viaggiano nell'aria e non su fili o cavi di trasmissione [2]. In un sistema wireless la trasmissione e la ricezione dei dati avvengono principalmente via radiofrequenza (RF) o attraverso raggi infrarossi (Irda). Il mezzo trasmissivo è dunque l'etere. Tra i possibili vantaggi offerti da una rete wireless possono essere citati:

- **Eliminazione del cablaggio:** si ha così la possibilità di realizzare una rete di comunicazione anche in quei luoghi in cui è difficile o addirittura impossibile installare dei cavi in maniera semplice e rapida.
- **Installazione facile, veloce e flessibile:** questo rende le reti wireless particolarmente adatte per installazioni "temporanee" (mostre, fiere, congressi, situazioni di emergenza, gruppi di lavoro).
- **Costi di gestione ridotti:** l'investimento iniziale richiesto per l'hardware delle reti wireless può essere più alto del costo dell'hardware delle reti cablate, ma le spese di gestione e manutenzione complessive sono in genere più basse.

- **Connettività per l’utenza in movimento:** un utente può collegarsi con il suo terminale e accedere a diverse informazioni o a qualsiasi tipo di applicazione, ovunque si trovi un punto di accesso nell’area di copertura della rete.
- **Roaming:** continuità di comunicazione anche spostandosi da una località ad un’altra.

2.2 Svantaggi della rete wireless.

La rete wireless ha introdotto diverse caratteristiche e migliorie che la rete fissa non offriva, ma anche svantaggi rilevanti che nel seguito saranno evidenziati:

- Il mezzo di trasmissione scelto comporta implicitamente una banda minore rispetto a quella ottenibile in una rete fissa per le proprietà stesse della trasmissione attraverso onde elettromagnetiche.
- Gli AP sono dispositivi spesso collegati ad una rete fissa capace di una certa velocità di trasferimento che l’AP utilizza per servire gli utenti ad esso collegati. Quindi più utenti sono collegati ad un AP maggiormente questo dovrà suddividere la banda disponibile equamente tra loro, per questo si accetta solo un numero di utenti tale da poter garantire una certa qualità di servizio.
- Reti di questo tipo sono più inaffidabili rispetto alle fisse e la perdita di informazione può essere sensibilmente maggiore poiché il mezzo fisico (etere) è più soggetto ad interferenze ed errori. Inoltre la comunicazione diventa così sensibile ad eventuali ostacoli presenti sul cammino tra AP e utente come muri.
- Altro problema da considerare è quello della sicurezza, tale problema è legato alla natura stessa del mezzo di comunicazione che, a differenza di

un cavo, non porta l'informazione unicamente al destinatario, ma la spedisce a chiunque sia in grado di riceverla all'interno dell'area.

2.3 Classificazione reti wireless.

Le reti wireless possono essere classificate, anche se non esiste una tassonomia universalmente accettata, secondo le esigenze da affrontare e del raggio di comunicazione richiesto dalle stesse, in particolare, si possono distinguere quattro categorie principali [3]:

- **WBAN (Wireless Body Area Network):** utilizzato per connettere dispositivi indossabili; i componenti di un computer indossabili sono distribuiti sul corpo umano (auricolari, computer da polso) e una BAN fornisce il supporto di rete e di auto-configurazione (l'inserimento e la rimozione di un dispositivo da una BAN dovrebbero essere trasparenti per l'utente). Il raggio di comunicazione corrisponde circa ad uno o due metri.
- **WPAN (Wireless Personal Area Network):** sono quelle che vengono definite reti wireless individuali ossia, gruppi di dispositivi differenti tra loro come telefonini, palmari, stampanti, notebook, collegati per soddisfare le esigenze delle singole persone. Le WPAN hanno un raggio di copertura pari a circa 10 metri, dunque dimensioni molto ridotte. Bluetooth è un tipico esempio di tecnologia applicata alle reti PAN; la banda disponibile è di un centinaio di Kbps.
- **WLAN (Wireless Local Area Network):** sistema di comunicazione relativo ad una determinata area locale in cui i vari dispositivi (PDA, palmtop, computer,…) si scambiano i dati senza ricorrere all'ausilio di cavi, introducendo in questo modo un maggior grado di mobilità e

flessibilità. Qui il raggio di copertura è molto più ampio, infatti possono coprire circa centinaia di metri. Consentono una banda dell'ordine delle decine di Mbps e lavorano alla frequenza di 2.4 GHz.

- **WWAN (Wireless Wide Area Network):** reti di questo genere nascono per permettere la comunicazione fra dispositivi situati a notevoli distanze, nell'ordine delle centinaia e migliaia di Km (sono chiamate anche reti geografiche). Esempi di tali reti utilizzano tecnologie UMTS (Universal Mobile Telecommunications System) e GPRS (General Racket Radio Service).

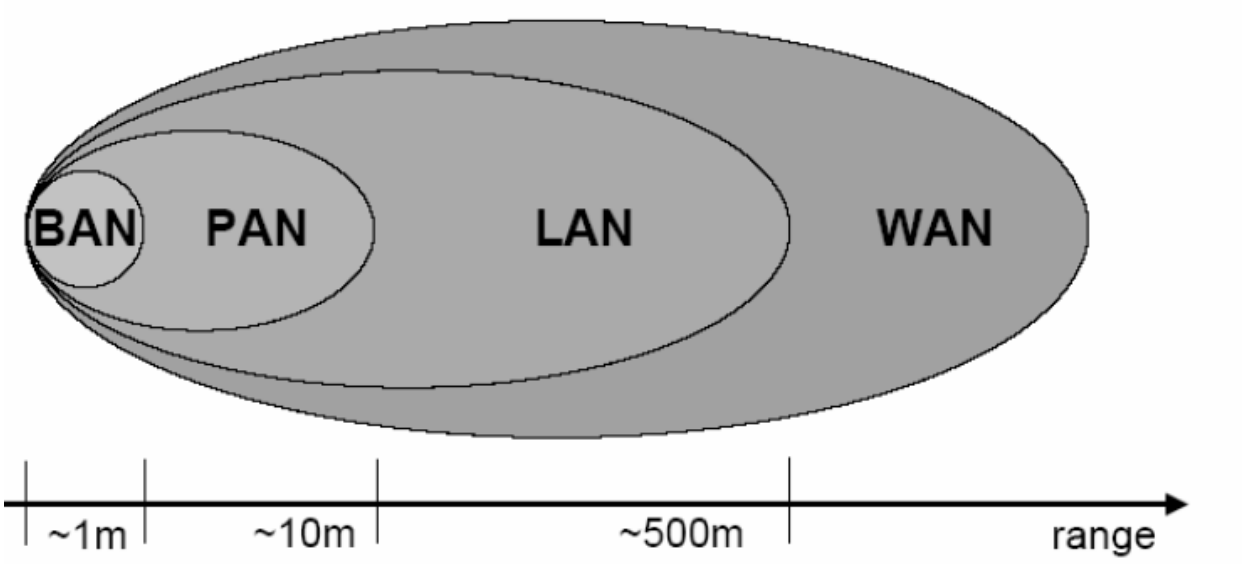

Figura 2.1 Tassonomia delle reti wireless.

2.4 Tipologie di handoff.

Come è stato accennato nel capitolo 1, per poter accedere alla rete wireless è necessario instaurare una comunicazione con un punto di accesso (AP). Ogni AP garantisce una certa copertura di rete, che può essere più o meno grande in base alla tecnologia utilizzata, quindi l'utente che si sposta con il proprio terminale può uscire dal raggio di copertura dell'antenna ed entrare nell'area coperta da un altro AP e quindi stabilisce la connessione con la nuova antenna. Il processo di disconnessione da un AP e di riconnessione ad un altro AP è noto come processo di handoff. Tale processo può essere diverso in base ai terminali wireless e della tecnologia wireless utilizzata. Sono possibili due fondamentali tipi di handoff:

- **Hard handoff**, il dispositivo mobile non è capace di interagire con due base station contemporaneamente, interrompe completamente la connessione con la precedente base station prima di stabilirne una nuova con un'altra base station, minimizzando così l'overhead di segnalazione ma incrementando la latenza e la perdita dei pacchetti.
- **Soft handoff**, utilizzato per descrivere un handover in cui la seconda connessione viene stabilita prima di interrompere completamente la prima. Durante questo periodo il dispositivo può comunicare simultaneamente con le due base station senza dover interrompere la comunicazione.

Un'altra classificazione dell'handoff è stata eseguita in base alle tecnologie della nuova e precedente station:

- **Handoff orizzontale**, transizione di una connessione da una base station ad un'altra all'interno dello stesso tipo di rete.

- **Handoff verticale**, processo di handover tra differenti tipi di rete (es. tra Wi-Fi e Bluetooth).

Uno dei problemi principali da affrontare nel supportare la mobilità delle elaborazioni è quello di assicurare una migrazione trasparente di un dispositivo da un punto di accesso ad un altro. In tale contesto l'handoff viene definito come il processo che si trova "behind the scene", che esegue, cioè, i corrispondenti cambiamenti di stato nella parte non mobile del sistema. Se un handoff garantisce una perdita minima o nulla di dati, è detto *smooth handoff*. Se garantisce un ritardo minimo o nullo è detto *fast handoff*. Un handover smooth e fast è detto *seamless handoff* (handoff continuo).

Le specifiche dello standard Wi-Fi (IEEE 802.11, di cui si parlerà in seguito) non impongono uno specifico tipo di handover, ma principalmente le apparecchiature Wi-Fi implementano l'hard handoff, complicando di conseguenza lo sviluppo di servizi continui e perdendo pacchetti durante la migrazione.

2.5 Standard 802.11.

Agli inizi degli anni novanta fu approvato lo standard IEEE **802.11** [4] che dettava le specifiche a livello fisico e datalink per l'implementazione di una rete LAN wireless (WLAN). Tale standard è nato per garantire l'affidabilità e la compatibilità tra sistemi prodotti da vari manufatturieri, favorendo quindi la diffusione delle WLAN.

Similmente ad Ethernet IEEE 802.3, lo standard IEEE 802.11 definisce: le funzioni necessarie ad una stazione 802.11 per operare sia in modalità peer-to-peer che integrata con una LAN esistente, la privacy e la sicurezza dei dati dell'utente trasportati sul mezzo radio, il sottostrato MAC per l'accesso al mezzo, lo strato fisico (PHY) e le relative interfacce.

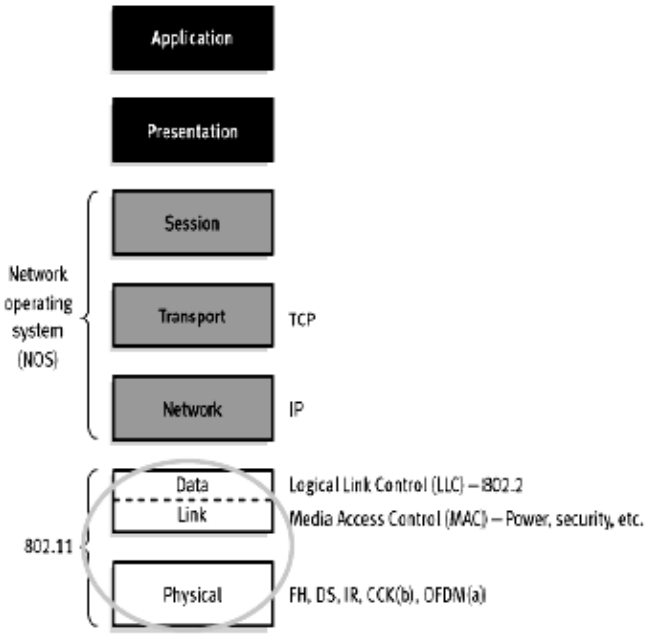

Figura 2.2 Posizione dello standard 802.11 nel modello OSI.

L'802.11 consentiva un data rate di 1 o 2 Mbps usando la tecnologia basata su onde radio nella banda 2.4 GHz o su raggi infrarossi. La limitata velocità dello standard determinò uno scarso successo e diffusione. L'evoluzione di tale tecnologia diversi anni dopo portò allo standard IEEE **802.11b** (denominato anche Wi-Fi, Wireless Fidelity) mantenendo la compatibilità con lo standard precedente. L'802.11b opera a 2.4 GHz, inoltre consente la possibilità di variare la velocità di trasmissione dei dati per adattarsi al canale, ha un data rate fino a 11 Mbps. Garantisce: possibilità della scelta automatica della banda di trasmissione meno occupata, possibilità di scelta automatica dell'access point in funzione della potenza del segnale e del traffico di rete. Inoltre permette la creazione di un numero arbitrario di celle parzialmente sovrapposte permettendo il roaming in modo del tutto trasparente. Nonostante queste caratteristiche ha comunque una bassa velocità di trasferimento ed inoltre sono possibili interferenze per l'utilizzo di una banda

non regolamentata. Sono stati sviluppati, successivamente, nuovi standard che hanno modificato i precedenti, cercando di migliorare l'implementazione delle WLAN, ad esempio lo standard **802.11a** che ha aumentato il data rate 54 Mbps e ha una frequenza pari a 5GHz, garantisce un maggior numero di utenti contemporaneamente collegati e utilizza una banda regolamentata, utilizzando una frequenza più alta, però, è meno tollerante ad ostacoli sul cammino. Tra il 2002 e il 2003 è stato sviluppato l'**802.11g**, utilizza la frequenza 2.4 GHz garantendo così una migliore portata e una migliore tolleranza agli ostacoli, supporta una velocità di 54 Mbps, ma ha gli stessi problemi dell'802.11b relativi all'utilizzo di una banda non regolamentata.

2.5.1 Topologie di reti per 802.11.

Esistono due modalità di funzionamento per una WLAN:

- **Independent Basic Service Set (IBSS) o Ad Hoc Network Infrastructure**.
- **Basic Service Set o Infrastructure Mode**.

Di seguito vengono descritte ponendo attenzione in particolar modo alla infrastructure local area network, poiché il lavoro di tesi si basa su questa tipologia.

2.5.1.1 Infrastructure local area network.

L'infrastructure mode prevede diversi elementi:

- Wireless NIC (Network Interface Card).
- Wireless Local Bridge comunemente chiamato Access Point (AP).
- BSS (Basic Service Set).
- ESS (Extended Service Set).
- DS (Distribution System).

I *NIC (o Wireless Terminal)* sono dispositivi che permettono di utilizzare i servizi della rete wireless, identificabili nei PC client. I Wireless Terminal sono delle schede elettroniche PCMCIA (Personal Computer Memory Card International Association) o USB (Universal Serial Bus) collegate sui notebook o sui PC desktop.

Gli *access point* sono punti di accesso, all'interno di un' infrastruttura (Distribution System), fungono da ricevitori/trasmettitori fissi con i dispositivi mobili che comunicano. Possono essere usati semplicemente come ripetitori di segnale o come elementi di interfaccia tra mondo senza fili (wireless) e mondo cablato svolgendo funzioni analoghe ad un bridge o router. Ciascun AP definisce intorno a sé una microcella di lavoro e può gestire diversi utenti.

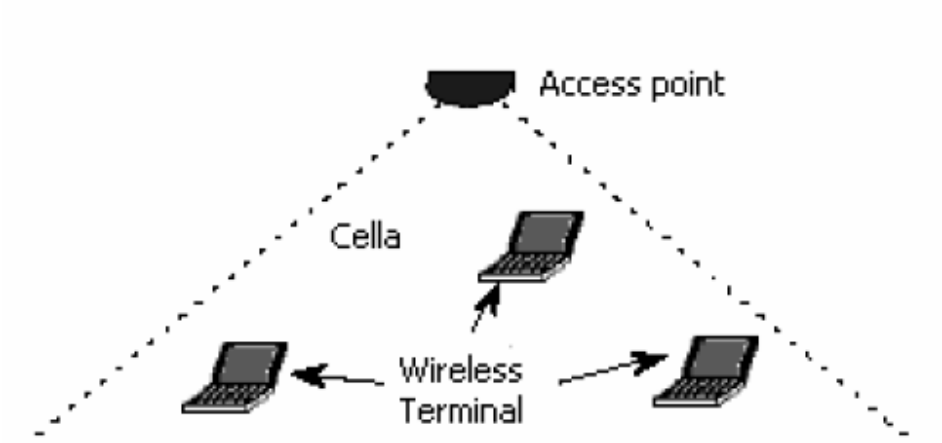

Figura 2.3 Access Point-Copertura cella radio.

Il *basic service set (BSS)* è costituito da un insieme di wireless terminal controllati da un access point. Quando un terminal vuole comunicare con un altro wireless terminal invia i dati all'AP, che si occupa di ridirigerli al terminale di destinazione, quindi la comunicazione fra terminali avviene solo attraverso gli AP.

L’*extended service set (ESS)* è un insieme di infrastrutture BSS, dove gli AP comunicano tra di loro attraverso il Distribution System (DS) per trasportare il traffico da una BSS all’altra, agevolando lo spostamento di WT tra BSS (roaming).
Il *Distribution System* è un’infrastruttura che permette a diversi AP di comunicare tra loro per scambiare pacchetti destinati ai terminali nei rispettivi BSS, girare pacchetti per inseguire le stazioni mobili che si spostano da un BSS ad un altro, scambiare pacchetti con una rete su cavo. Il DS può essere implementato sia come infrastruttura wired sia come infrastruttura wireless. Gli elementi di rete al di fuori dell’ESS vedono l’ESS e tutte le sue stazioni mobili come una singola rete al livello MAC dove tutte le stazioni sono fisicamente stazionarie. L’ESS quindi nasconde la mobilità delle stazioni mobili a quanto situato al di fuori di essa, ed è visto dai livelli superiori del modello OSI come una singola rete 802. Questa caratteristica di 802.11 consente ai protocolli di rete esistenti, che non possiedono il concetto della mobilità, di operare correttamente con una WLAN che supporta la mobilità. Nella figura seguente viene mostrato un esempio di infrastructure mode con tutti i suoi elementi fondamentali.

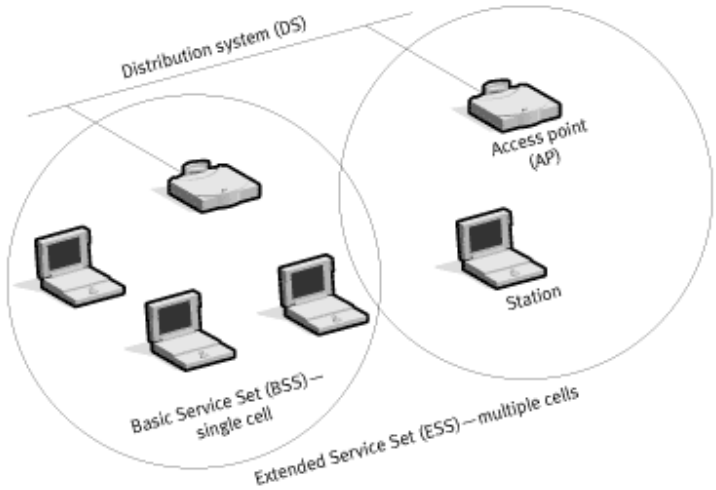

Figura 2.4 Esempio di Infrastructure local area network.

2.5.1.2 Ad-Hoc mode.

Una rete Ad-Hoc è una rete in cui sono presenti solo wireless terminal, i quali comunicano tra loro senza Access Point e senza bisogno di alcuna infrastruttura di supporto. In una rete Ad-Hoc non ci sono funzioni di relay, un terminale è raggiungibile solo se situato nel raggio di copertura. Lo standard 802.11 definisce una rete Ad-Hoc come IBSS (Indipendent Basic Service Set), in quanto essa non comunica con nessun altra rete Ad-Hoc o wired. Il suo utilizzo è molto importante negli ambienti dinamici e in quegli ambienti in cui è impossibile avere un'infrastruttura di sostegno.

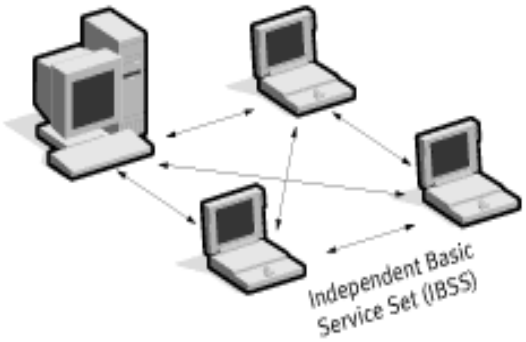

Figura 2.5 Esempio di Ad-Hoc mode.

2.5.2 Handoff in IEEE 802.11x.

Il problema dell'handoff all'interno delle WLAN è molto importante ed è stato esaminato anche dalla commissione IEEE 802.11x, che ha creato l'**802.11f**, anche se ancora in via di standardizzazione, vuole standardizzare il protocollo Inter-Access Point Protocol (IAPP) che si occupa della registrazione degli AP in una rete e dello scambio di informazioni tra AP quando una stazione si muove tra

aree di copertura supportate da AP di diversi produttori, definisce, dunque, il modo in cui più access point possono dialogare fra loro per trasferire più velocemente i dati associati ad una connessione.

Un altro standard è l'**802.11r**, che si occupa di fast roaming. Quando un client passa dalla copertura radio di un access point a quella di un altro, dovrebbero essere mantenute le condizioni di autenticazione e sicurezza. La questione diventa problematica per applicazioni, come il VoIP su WLAN, in cui il client si muove velocemente e la sua connessione non solo non deve cadere, ma deve avere una qualità di servizio inalterata, intende dunque, tale standard, definire una modalità di roaming veloce standard, in modo ad esempio che gli utenti non debbano autenticarsi a ogni nuovo access point che incontrano.

CAPITOLO 3

SIP: Session Initiation Protocol.

Di seguito sarà descritto il protocollo SIP, i suoi componenti e come estendere le sue funzionalità di base attraverso la creazione di event package.

3.1 Caratteristiche generali di SIP.

SIP (Session Initiation Protocol) [5] è un protocollo di segnalazione di livello applicativo, è stato standardizzato da Internet Engineering Task Force (IETF) nel 1999, consente la creazione, la modifica e la terminazione di sessioni tra uno o più utenti.

Il protocollo SIP proprio perché definito come parte dell'architettura IETF permette un'ottima integrazione con gli altri protocolli definiti per le reti IP, come RTP (Real-time Transport Protocol) / RTCP (RTP Control Protocol), SDP (Session Description Protocol, di cui si parlerà), ma non risulta dipendente da essi e così può essere utilizzato senza problemi insieme con altri protocolli di segnalazione.

Una caratteristica molto importante è inoltre l'indipendenza dal protocollo di trasporto sottostante, per questo è possibile utilizzare sia UDP (User Datagram Protocol) che TCP (Transmission Control Protocol) anche se, come si vedrà in seguito, è utilizzato, nella maggior parte dei casi, il protocollo UDP.

Consente la separazione dei dispositivi fisici, come il telefono, dagli utenti e la logica del server dal controllo centralizzato. La separazione del servizio dai

dispositivi fisici permette l’implementazione di funzionalità quali la gestione di informazioni sulla presenza o meno di utenti ed il supporto della mobilità.

Questa flessibilità porta SIP ad essere usato in molte applicazioni e servizi, inclusi giochi interattivi, musica e video on demand come anche nella telefonia e nel Web conferencing.

Tale protocollo non vuole definire il tipo di sessione che deve essere stabilito, ma come questa debba essere gestita.

SIP supporta 5 funzionalità fondamentali per stabilire e terminare comunicazioni multimediali [6]:

1. **User Location:** determinazione del sistema terminale da usare per la comunicazione multimediale.
2. **User Availability:** determinazione della volontà di una parte di essere coinvolta nella comunicazione.
3. **User Capabilities:** determinazione della tipologia dei media coinvolti nella comunicazione e dei parametri descrittivi.
4. **Session Set Up:** instaurazione della sessione sia nel lato originante che terminante.
5. **Session Management:** gestione della sessione, che prevede la modifica dei parametri per una sessione in corso, invocazione di servizi per conto dell'utente e terminazione della sessione.

3.2 Architettura SIP.

SIP è un protocollo di tipo *client-server* ed i suoi principali elementi sono l'User Agent, il Proxy Server, il Redirect Server ed il Registrar Server.

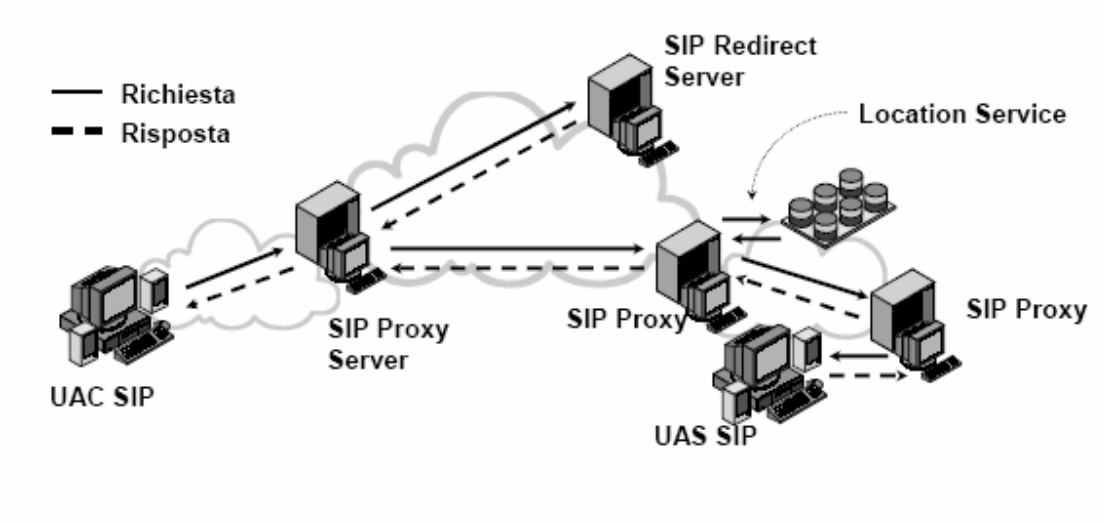

Figura 3.1 Architettura SIP.

- **SIP User Agent (UA):** o SIP *endpoint*, è un software, di PC o di terminale telefonico, che interagisce direttamente con un utilizzatore umano; funziona come *Client* (UAC) quando invia una richiesta, come *Server* (UAS) quando vi risponde. Due UA (es. PDAs, PCs, telefoni...) possono comunicare tra loro direttamente o tramite server intermedi.

- **SIP Registrar Server:** è un server che accetta richieste di registrazione da parte di tutti gli User Agent presenti all'interno del dominio di rete a lui assegnato e provvede a memorizzare le loro informazioni in un Location Server tramite un protocollo non-SIP (es. Tutti i terminali con

identificatori del tipo x@macrosoft.com si registrano presso il registrar server del dominio macrosoft.com).

- **SIP Proxy Server:** è un router di livello applicativo con il compito di inviare le richieste dall'UA al successivo SIP server o ad un altro UA interno alla rete. Tale server può essere di due tipi: stateful o stateless. Lo stateful proxy server ricorda le richieste inviate, le risposte fornite e i messaggi inoltrati. Lo stateless proxy server, invece, dimentica tutte le informazioni nel momento in cui ha inviato una richiesta.

- **SIP Redirect Server:** risponde alle richieste che riceve fornendo l'indirizzo di un user agent o di un server in grado di localizzare l'utente chiamato.

3.3 Il formato dei messaggi SIP.

Il protocollo SIP [7], poiché basato sul modello client-server, risulta essere perfettamente integrato con Internet e la sua filosofia. La sintassi, sia delle richieste sia delle risposte, segue le specifiche dei messaggi HTTP.

3.3.1 Messaggio di richiesta SIP.

Un messaggio di richiesta SIP è costituito da tre elementi:
-Request line.
-Header.
-Corpo del messaggio.

Nella **request line** troviamo una linea di codice che consiste in un *metodo*, un *Request-URI* (paragrafo 3.3.3) e la versione del protocollo SIP in uso. SIP prevede diversi tipi di metodi:

- INVITE: Inizia una chiamata invitando un utente a partecipare ad una sessione.
- BYE: Termina una chiamata tra due utenti.
- ACK: Conferma la ricezione di una risposta finale ad una richiesta INVITE da parte del chiamante.
- CANCEL: Termina una richiesta pendente senza concludere la chiamata.
- OPTIONS: Richiede informazioni riguardo alle caratteristiche di un server.
- REGISTER: Fornisce la mappa per la risoluzione degli indirizzi, consentendo ad un server di conoscere l'ubicazione di altri utenti.
- INFO: Utilizzato per messaggi di controllo scambiati durante una sessione.

Oltre a questi metodi di base, sono stati definiti altri sei metodi, chiamati "extension method":

- MESSAGE: consente di realizzare servizi di Instant Messaging basati su SIP. Il corpo di una richiesta di questo tipo contiene l'Instant Message da inviare.
- SUBSCRIBE: permette di ottenere lo stato corrente e gli aggiornamenti di stato da un nodo remoto.
- NOTIFY: permette di notificare i cambiamenti di stato di un nodo a tutti quelli che hanno richiesto tale notifica mediante il metodo SUBSCRIBE.
- PRACK: questo metodo svolge la stessa funzione dell'ACK, ma è riferito alle provisional response (si fa riferimento alle risposte 1xx, indicano che il server sta contattando l'utente chiamato, che il tempo necessario risulta

essere superiore a 200 ms, e che quindi non dispone ancora di una risposta definitiva, lasciando così il client in attesa di ulteriori informazioni).

- REFER: consente di effettuare dei trasferimenti di chiamata.
- UPDATE: permette di aggiornare i parametri di una sessione.

Gli ***header*** dei messaggi contengono informazioni di servizio del protocollo, di seguito ne analizziamo alcuni:

- From: indica il mittente della richiesta mediante un SIP URI e un tag, in altre parole una stringa random impiegata per scopi di identificazione.
- To: indica il destinatario della richiesta mediante un SIP URI e un tag(come il campo From).
- Call-ID: contiene un identificatore unico globale della sessione. Questo identificatore è generato con opportuni algoritmi e tipicamente possiede la forma local_id@host.
- CSeq: contiene un numero e un metodo. Il numero conta le richieste effettuate all'interno di un dialog, il metodo rappresenta il metodo della richiesta.
- Via: contiene l'indirizzo al quale il mittente si aspetta di ricevere delle risposte a questa richiesta. Oltre a ciò è presente anche un parametro branch che identifica la transazione a cui appartiene il messaggio.
- Max-Forwards: indica il numero massimo di proxy che la richiesta può attraversare. Questo campo corrisponde al campo TTL utilizzato in alcuni protocolli (come IP). Il valore di default è 70.
- Contact: contiene una lista di indirizzi dove un utente può essere trovato per le richieste future o per ulteriori comunicazioni.

- Content-Type: indica il protocollo usato nel corpo del messaggio. Di solito si tratta di SDP.
- Content-Length: indica la lunghezza del corpo del messaggio.
- Expires: indica il tempo di validità di una registrazione.

Il **corpo** del messaggio è separato dagli header mediante una riga vuota e, essendo indipendente da SIP, può contenere qualsiasi cosa, ad esempio, delle informazioni relative alle caratteristiche della sessione, il tutto codificato nel linguaggio di specifica SDP.

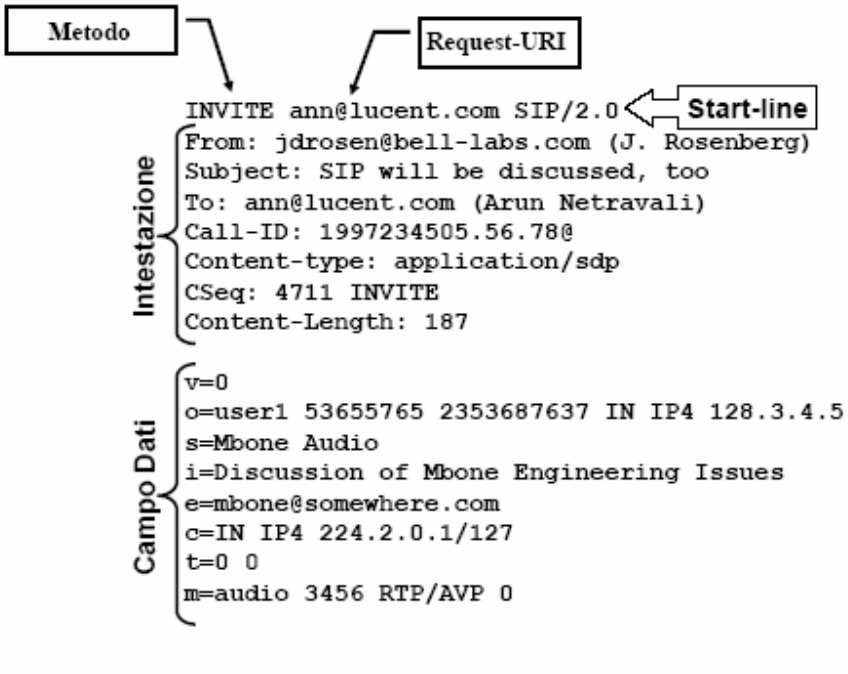

Figura 3.2 Esempio messaggio di richiesta SIP.

3.3.2 Messaggio di risposta SIP.

Un messaggio di risposta SIP è costituito anche da tre elementi:

-Status line.

-Header.

-Corpo del messaggio.

Per gli ultimi due elementi valgono le stesse proprietà evidenziate nel caso dei messaggi di richiesta.

Lo ***status line*** è un intero di tre cifre che indica il risultato del tentativo di capire e soddisfare la richiesta. La prima cifra definisce la classe della risposta, le altre non hanno un ruolo preciso. Sotto sono illustrate le possibili classi di risposta:

- 1XX: *Provisional* indica che una richiesta è stata ricevuta, ma che il server contattato non ha ancora una risposta definitiva e sta continuando a processare la richiesta (es. 100 Trying, 180 Ringing, 182 Queued);
- 2XX: *Success* indica che la richiesta ha avuto successo (es. 200 OK, 202 Accepted);
- 3XX: *Redirection* indica che si hanno nuove locazioni dell'utente e quindi sono necessarie altre azioni per poter completare la richiesta (es. 300 Multiple Choices, 301 Moved Permanently, 302 Moved Temporarily, 305 Use Proxy);
- 4XX: *Request Failure* indica che la richiesta contiene una sintassi errata o non può essere completata da questo server, deve essere quindi modificata ed in seguito si può ritentare (es. 400 Bad Request, 401 Unauthorized, 404 Not Found, 482 Loop Detected);
- 5XX: *Server Failure* indica che il server non riesce a completare una richiesta apparentemente valida (es. 501 Not Implemented);

- 6XX: *Global Failure* indica che la richiesta non può essere completata da alcun server (es. 603 Decline, 606 Not Acceptable).

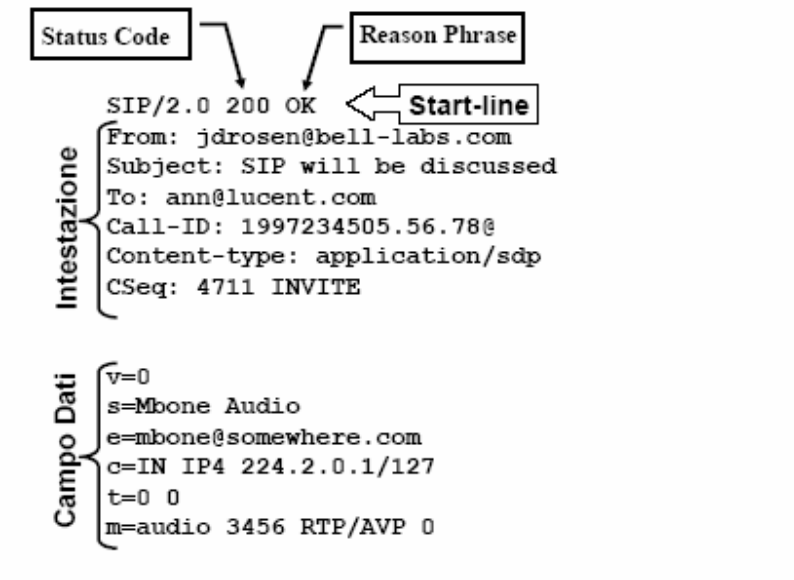

Figura 3.3 Esempio messaggio di risposta SIP.

3.3.3 Indirizzo utente: SIP-URI.

Un Request-URI è un *SIP URL* (Uniform Resource Locator) o un *URI* (Uniform Resource Identifier) generale, che permette di indicare l'utente o il servizio al quale si riferisce una particolare richiesta. Un indirizzo SIP può essere descritto nella forma user@host, dove user è il nome dell'utente o il numero di telefono, host è il nome del dominio o l'indirizzo della rete:

- sip:alice@unibo.it;
- sip:davide@100.7.5.3;
- sip:+39-02-1234-5678:3333@gateway.com:user = phone.

Gli indirizzi utilizzati per identificare i partecipanti ad una sessione, che qui saranno chiamati utenti o terminali, hanno una struttura simile a quella degli indirizzi e-mail o dei numeri telefonici e sono assegnati dai fornitori di servizi della rete a cui gli utenti appartengono. Per consentire una completa mobilità utente tali identificatori sono indipendenti dai dispositivi utilizzati per le connessioni e dai punti di accesso alla rete.

3.3.4 Session Description Protocol.

Il Session Description Protocol [8] è il protocollo utilizzato per la descrizione di sessioni multimediali, la descrizione SDP non è altro che un breve elenco testuale di tutte le caratteristiche che un utente deve comunicare ad altri utenti per partecipare ad una sessione. Le informazioni principali contenute in una descrizione SDP sono:

- Il nome della sessione e il suo scopo.
- Il tempo o gli intervalli di tempo in cui la sessione è attiva.
- Il tipo di dati multimediali trasmessi nella sessione.
- Le informazioni necessarie per la ricezione di questi dati (come gli indirizzi, le porte, i formati).

Per quanto riguarda i dati multimediali trasmessi, con SDP è possibile specificare:

- Il tipo di tali dati (video, audio).
- Il protocollo di trasporto usato.
- Il formato dei dati.

Poiché le risorse necessarie per partecipare ad una sessione possono essere limitate, è possibile aggiungere dati addizionali come:

- Informazioni sulla banda da utilizzare nella sessione.
- Indirizzo o altri dati della persona responsabile della sessione.

Una descrizione SDP consiste in diverse linee di testo nella forma <type>=<value>, dove <type> è sempre solo un carattere case-significant, <value>, invece, è costituito da una stringa la cui struttura dipende da <type>. Anche questo campo è case-significant, a meno che non venga dichiarato esplicitamente il contrario. Alcune linee della descrizione sono obbligatorie, mentre altre sono opzionali. In ogni caso tutte le linee devono rispettare esattamente l'ordine che vedremo di seguito (l'ordine fissato facilita il rilevamento degli errori). I campi opzionali sono contrassegnati con il carattere '*'.

v= (protocol version)
o= (owner/creator and session identifier)
s= (session name)
i= * (session information)
u= * (URI of description)
e= * (email address)
p= * (phone number)
c= * (connection information – not required if included in all media)
b= * (bandwidth information)
z= * (time zone adjustments)
k= * (encryption key)
a= * (zero or more session attribute lines)
t= (time the session is active)
r= * (zero or more repeat times)
m= (media name and transport address)
i= * (media title)
c= * (connection information – optional if included at session-level)
b= * (bandwidth information)

k= * (encryption key)
a= * (zero or more media attribute lines)

L'insieme delle lettere del campo <type> è stato scelto deliberatamente piccolo e non è espandibile (i parser SDP ignorano completamente tutte le righe che iniziano con un carattere non noto). Il codice 'a=' costituisce il mezzo principale per estendere il protocollo SDP e per adattarlo a particolari applicazioni o media.

Nonostante SIP abbia delle caratteristiche molto importanti, ha, in alcuni casi, un limite: i suoi messaggi non trasportano dati ma solo informazioni di controllo e di sessione, si sta cercando di estenderlo per renderlo ancora più efficace, cercando di utilizzarlo anche per il trasporto dei dati.

3.3.5 Protocollo di trasporto.

Il protocollo usato nel trasporto dei messaggi di segnalazione è l'*UDP*, è stato scelto per poter ridurre i tempi di Call Set Up, in questo modo infatti riusciamo ad eliminare la componente del ritardo di set up relativa al tempo necessario per l'instaurazione della connessione *TCP*. Per poter però garantire alla trasmissione dei messaggi l'affidabilità a livello applicazione è utilizzato un meccanismo di time-out ed il meccanismo di richiesta-risposta, tipico del protocollo. Se il chiamante non riceve una risposta alla sua richiesta entro un tempo prestabilito, il time out appunto, ne deduce che è andata persa e la ritrasmette. Il chiamato invece riceve un ACK dal chiamante quando a questi arriva la risposta. Se il chiamante registra per molte volte il fallimento della richiesta può decidere di aprire una connessione tramite il TCP.

3.4 Funzionamento degli User Agent.

Il **SIP User Agent** (UA) è un endpoint e può fungere da client (UAC) o da server (UAS), i due ruoli sono dinamici, nel senso che nel corso di una sessione un client può fungere da server e viceversa, ossia in una transazione, appartenente ad una sessione, funge da client (server), nella successiva potrebbe fungere da server (client). Quando funge da client, dà inizio alla transazione originando richieste. Quando funge da server, ascolta le richieste e le soddisfa, se possibile. Uno User Agent è in sostanza una macchina a stati, che evolve in dipendenza di messaggi SIP, e registra informazioni rilevanti del dialogo (il dialogo ha inizio quando si risponde positivamente al messaggio di ***Invite*** e termina con un messaggio di ***Bye***).
L'*user agent client* si occupa di creare e inviare messaggi di richiesta SIP ad un UAS. Il client può inviare diversi tipi di richieste, ad esempio:

- INVITE.
- BYE.
- ACK.
- CANCEL.
- OPTION.
- INFO.
- MESSAGE.

L'UAC è in grado di registrarsi presso un Registrar attraverso il messaggio di richiesta REGISTER. Inoltre interagisce con i Server (Redirect Server, Proxy Server) al fine di instradare i messaggi al destinatario.
L'*user agent server* si occupa, invece, di rispondere ai messaggi SIP che riceve, ossia crea e invia risposte al client che lo ha interrogato. L'UAS può dunque inviare diverse risposte, ad esempio:

- 1xx: Informativo.

- 2xx: Successo.
- 3xx: Redirezione.
- 4xx: Errore del client.
- 5xx: Errore del server.
- 6xx: Fallimento globale.

Anche l'UAS può interagire con i Server proprio come l'user agent client.
La decisione di voler iniziare un dialogo è sempre presa da uno user agent client, anche se è necessaria l'approvazione dello user agent server affinché il dialogo possa iniziare.

3.5 Funzionamento dei SIP Server.

Come già detto, le funzionalità base dei SIP server sono suddivise in tre categorie principali [9]:

- **SIP Registrar Server;**
- **SIP Redirect Server;**
- **SIP Proxy Server;**

Questa separazione è stata fatta dagli autori di SIP per rendere più chiara la strutturazione delle funzionalità nei SIP server, ma non necessariamente tali funzionalità sono implementate in applicazioni distinte, di solito, infatti, uno stesso server può svolgere differenti funzioni secondo le esigenze dell'ambiente in cui si trova ad operare.
Le specifiche di SIP, inoltre, non vietano che uno stesso User Agent possa svolgere una delle tre funzioni sopra elencate.

3.5.1 Registrar Server.

Un **registrar server** è definito come un'entità che accetta delle richieste di registrazione (metodo REGISTER) e che inserisce le informazioni contenute in esse nel location service del dominio che gestisce. I messaggi REGISTER sono inviati dai client per aggiornare una corrispondenza tra il/i SIP URI di un utente (header From) e la macchina su cui l'utente può essere trovato (indicata nel Contact header con uno o più SIP URI).
Una registrazione, può ad esempio, avere la seguente struttura:

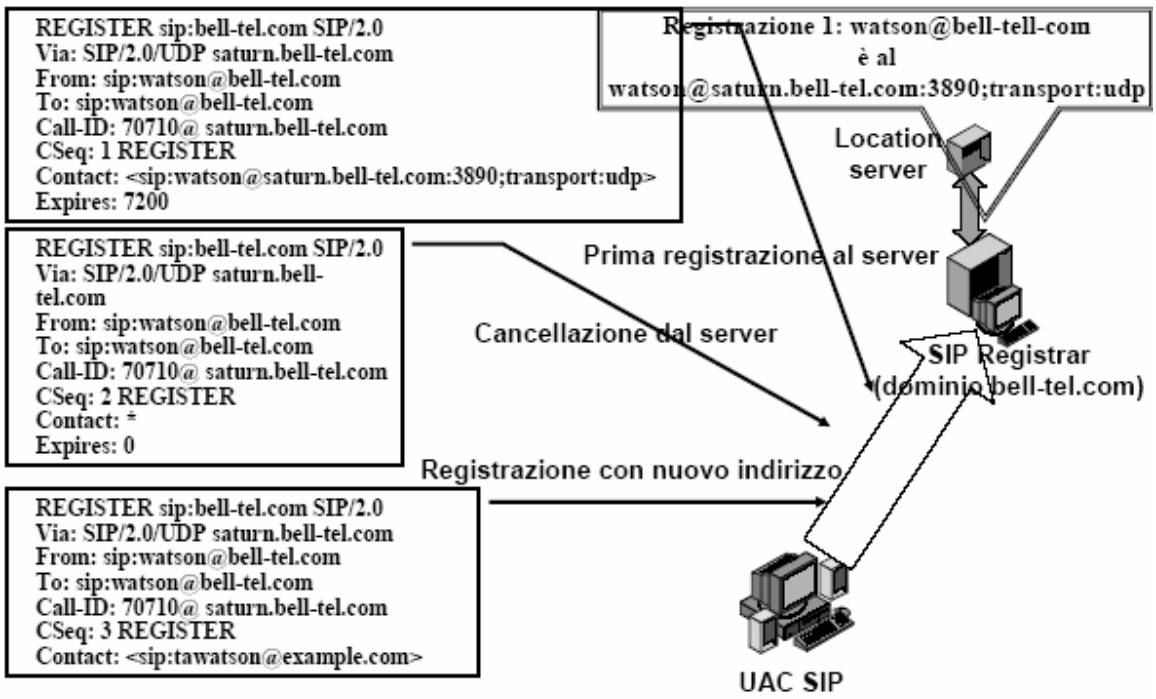

Figura 3.4 Esempio con SIP Registrar.

La richiesta di registrazione può, inoltre, essere utilizzata per ottenere tutte le corrispondenze associate a uno specifico indirizzo. Le informazioni contenute nelle

registrazioni sono depositate in un *Location Service* nel dominio di competenza del registrar server. Il Location Service è fuori degli scopi del protocollo SIP, può essere implementato in qualsiasi modo, ad esempio, può essere su una macchina remota cui si accede con LDAP (Lightweight Directory Access Protocol), oppure può risiedere anche sulla stessa macchina del registrar.
Infine il Registrar può essere co-locato in un Proxy o Redirect Server.

3.5.2 Redirect Server.

Un **redirect server** riceve delle richieste SIP e risponde ad esse con risposte del tipo 3xx (redirection), fornendo al mittente nuovi indirizzi SIP da contattare. Questi indirizzi sono inseriti in uno o più Contact header nella risposta. Alcune risposte fornite da un redirect server sono:

- **300 Multiple Choices:** l'indirizzo nella richiesta è stato risolto in differenti indirizzi che possono essere contattati dal client. E'compito del client scegliere una destinazione cui inviare la richiesta.
- **301 Moved Permanently:** l'utente non può essere più trovato all'indirizzo specificato nella richiesta e il chiamante deve riprovare contattando l'indirizzo contenuto nel Contact header.
- **302 Moved Temporarily:** l'utente può essere trovato temporaneamente a un indirizzo differente da quello nella richiesta. La durata della validità degli indirizzi forniti può essere espressa nel Contact header.
- **305 Use Proxy:** l'indirizzo di destinazione richiesto deve essere acceduto attraverso il proxy indicato nel Contact header.

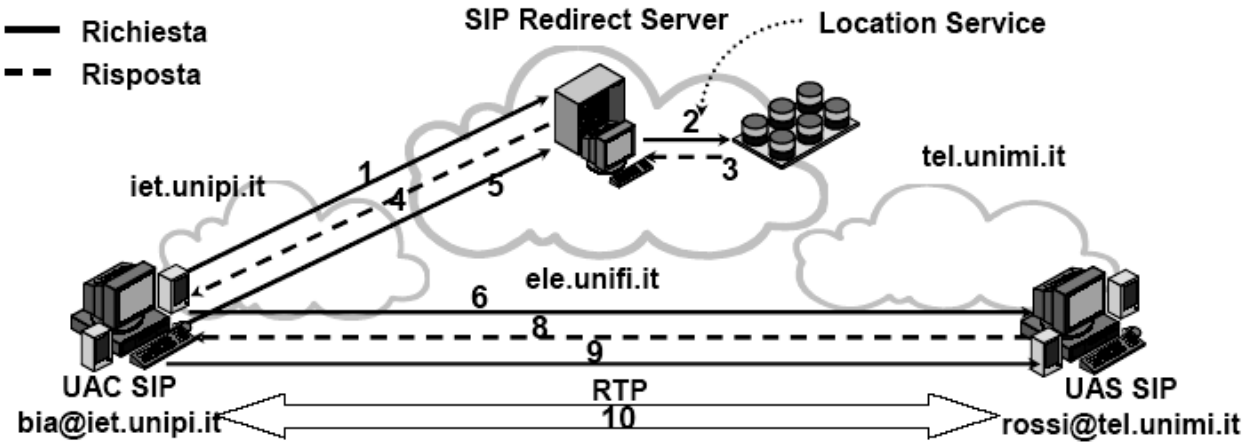

Figura 3.5 Esempio con SIP Redirect.

Notare che il secondo INVITE, di figura 3.5, possiede gli stessi campi From, To e Call-Id del primo, ma con CSeq header differente.

3.5.3 Proxy Server.

Lo standard SIP definisce il **proxy server** come router di livello applicativo che ha il compito di instradare i messaggi SIP fino all'UA di destinazione. Una richiesta può attraversare più server di questo tipo, ognuno dei quali può prendere decisioni di routing e modificare i messaggi prima di inoltrarli. Un proxy, comunque, non può modificare i messaggi a suo piacimento (ad esempio non può cambiare la descrizione SDP contenuta in una richiesta INVITE), non può generare

delle richieste di sua iniziativa (se non in casi eccezionali), o terminare delle chiamate già esistenti, mediante una richiesta BYE.
Le risposte seguono il percorso contrario della richiesta attraverso gli stessi server, ricostruendolo grazie alle informazioni (indirizzo e numero porta) registrate dai proxy in opportuni header del messaggio (es. header Via).

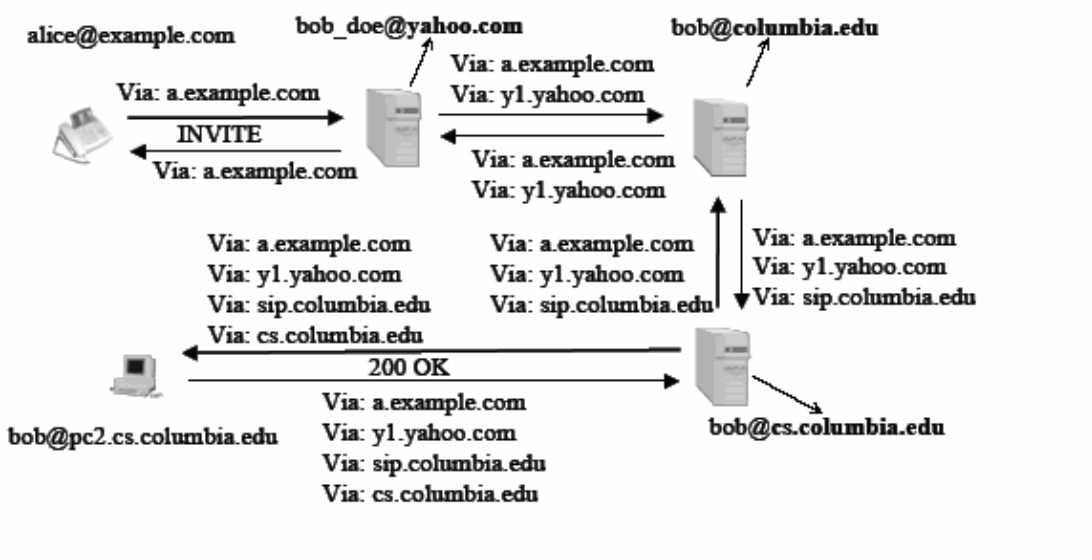

Figura 3.6 Instradamento risposte SIP.

I proxy possono decidere anche di rimanere nel percorso di segnalazione dopo che la sessione tra due UA è stata instaurata, ciò è possibile attraverso l'introduzione di un header *Record-Route* nel messaggio INVITE o SUBSCRIBE. In questo modo tutti i messaggi SIP scambiati durante una sessione passano attraverso il proxy. Di seguito è stato riportato un esempio di comportamento del proxy con header Record-Route nel messaggio INVITE.

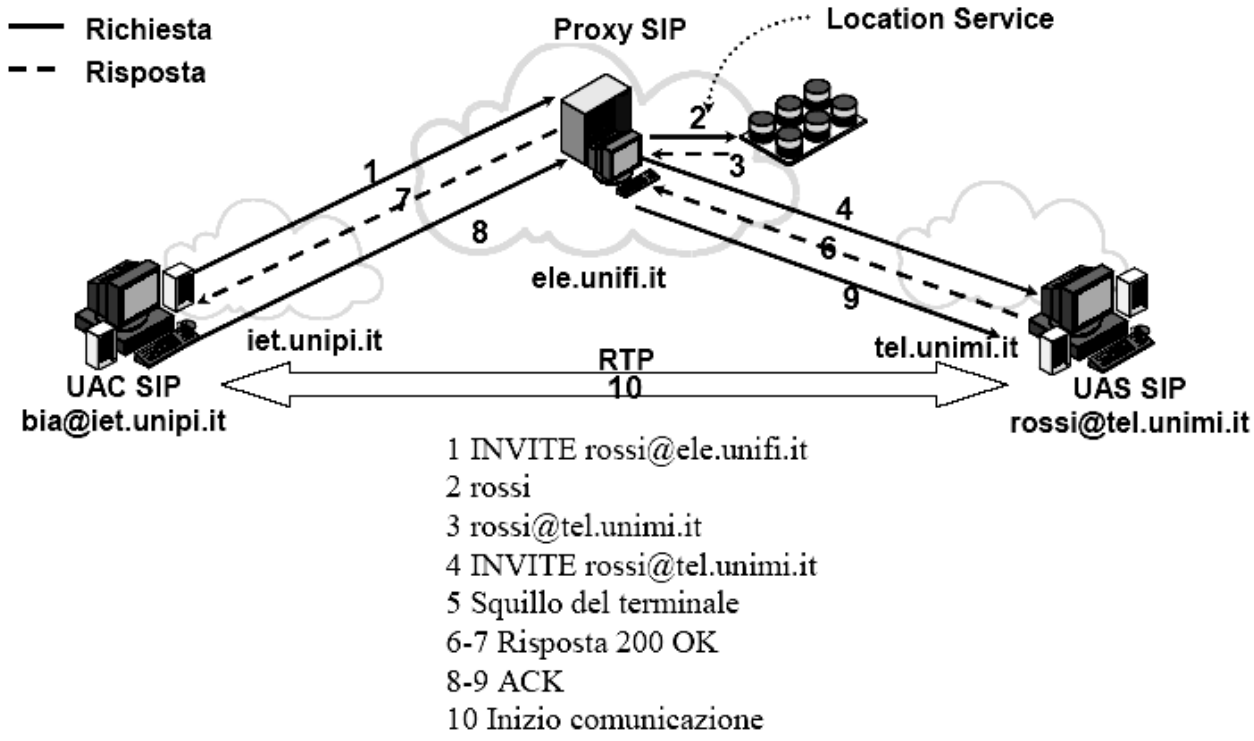

Figura 3.7 Esempio con proxy SIP.

Se un proxy decide di non utilizzare il Record-Route in una richiesta INVITE (o SUBSCRIBE), già dall'ACK corrispondente a tale richiesta, esso non riceverà più alcun messaggio SIP scambiato dagli UA durante la sessione. Questi messaggi, infatti, verranno inviati direttamente al destinatario, saltando i proxy server intermedi.

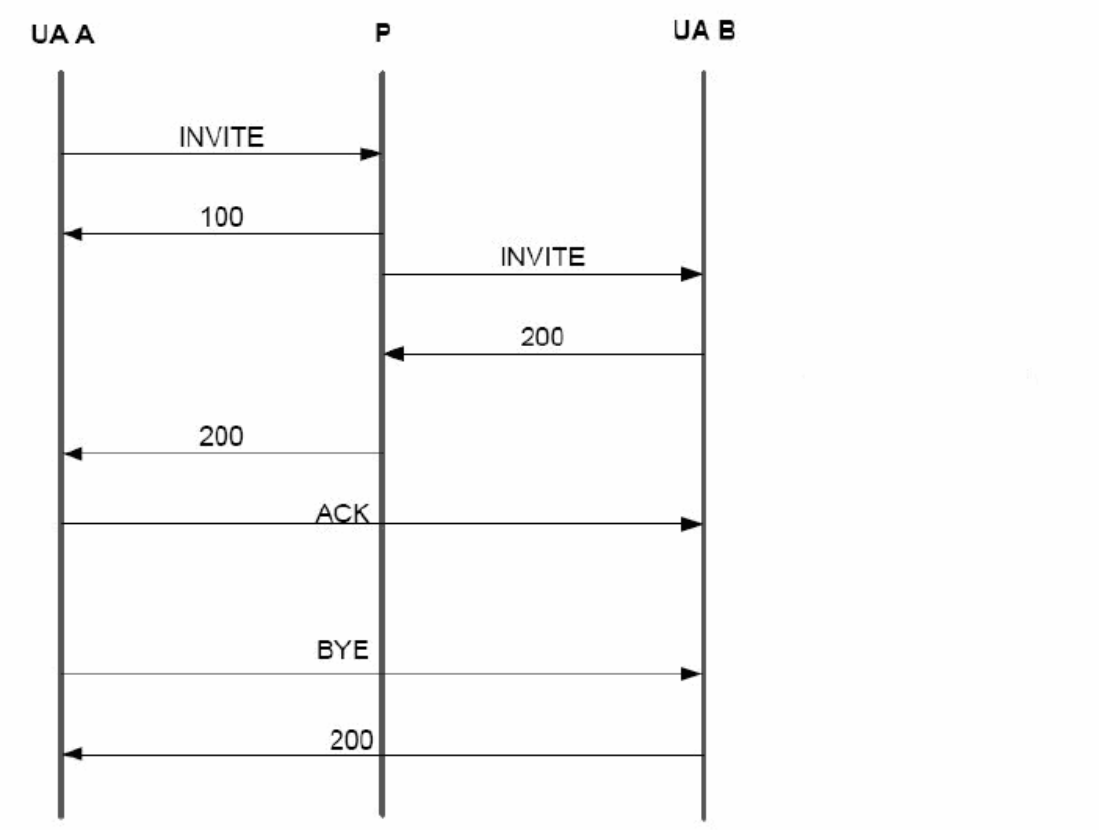

Figura 3.8 Comportamento di un proxy senza Record-Route.

Tutte queste operazioni sono eseguite dai proxy in modo del tutto trasparente per gli User Agent.

I proxy server possono essere di due tipi:

- Gli **stateful proxy.**
- Gli **stateless proxy.**

3.5.3.1 Stateless proxy.

Uno **stateless proxy** si limita semplicemente a trasmettere i messaggi che riceve senza occuparsi di alcuno stato associato ad essi. Questo significa che, una volta inoltrato il messaggio, il proxy "dimentica" ciò che ha fatto. Tale

funzionamento consente di avere delle prestazioni molto buone, tanto che gli stateless proxy normalmente costituiscono il backbone dell'infrastruttura SIP e possono essere impiegati anche come nodi per il bilanciamento del carico. Ovviamente la mancanza di stato comporta delle conseguenze:

- Uno stateless proxy non è in grado di associare le risposte alle rispettive richieste, dato che non possiede nessuna informazione riguardo a ciò che ha già inoltrato. Quindi tali server non offrono nessun supporto alle transazioni, e, quindi, non possono sapere se una transazione ha avuto successo o meno.
- Uno stateless proxy non è in grado di associare le ritrasmissioni delle richieste e delle risposte con le istanze precedenti di questi messaggi. Ciò significa che esso gestisce le ritrasmissioni come se quella che ha ricevuto fosse la prima copia del messaggio.
- In caso di perdita di un messaggio, il proxy non lo ritrasmette. Queste ritrasmissioni sono affidate agli User Agent o agli stateful proxy.

3.5.3.2 Stateful proxy.

Gli **stateful proxy** si occupano della gestione di transazioni più che di singoli messaggi. Questi proxy gestiscono due tipi di transazioni: i server transaction, per ricevere le richieste e restituire le risposte, e le client transaction per inviare le richieste e ricevere le risposte ad esse associate.

Le richieste in arrivo sono processate con dei server transaction e poi sono inoltrate in avanti con una o più client transaction.

Una risposta in arrivo è ricevuta dalla client transaction corrispondente e poi inoltrata indietro al server transaction. Tutte queste operazioni sono affidate ad un "proxy core object". Tale oggetto sceglie l'indirizzo (o gli indirizzi) di destinazione e stanzia uno o più oggetti client transaction per l'invio del messaggio.

Il proxy core object, inoltre, riceve tutte le risposte ricevute dalle client transaction stanziate e sceglie tra esse quella/e da mandare alla server transaction.
Il mantenimento dello stato delle transizioni gestite e della storia dei messaggi consente di effettuare operazioni più complesse rispetto a quelle di uno stateless proxy. Uno stateful proxy può, ad esempio, identificare le ritrasmissioni di un messaggio già ricevuto e decidere di inoltrarle solo se le circostanze lo richiedono. La mancanza di stato, invece, costringerebbe il proxy a inoltrare ogni messaggio ricevuto, dato che esso non sarebbe in grado di distinguere il messaggio originale dalle ritrasmissioni.

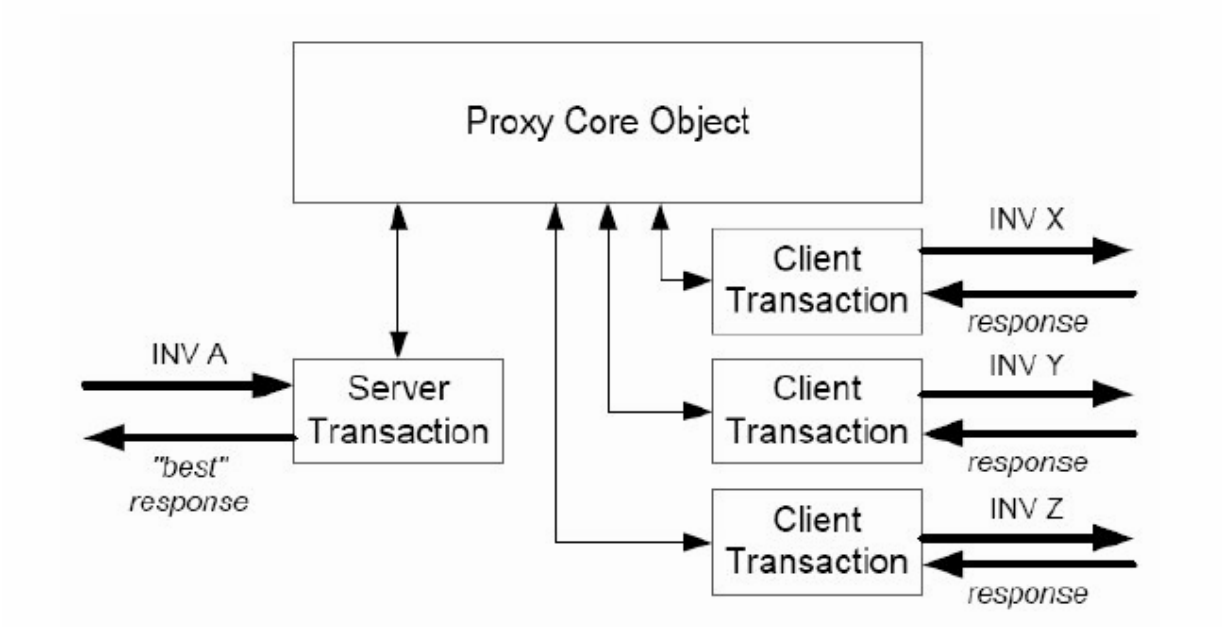

Figura 3.9 Schema di funzionamento di uno stateful proxy.

Uno stateful proxy, può, inoltre, ritrasmettere i messaggi perduti e generare delle richieste CANCEL se necessario.
Ovviamente la presenza di uno stato comporta diversi svantaggi:

- In uno stateful proxy il consumo di memoria è molto più alto che in un stateless proxy. Questo porta a dover limitare il numero di transazioni concorrenti che può essere gestito.
- Lo throughput di uno stateful proxy (vale a dire il numero di richieste gestite in un secondo) è fortemente penalizzato dal fatto che il tempo richiesto per spedire un messaggio è più alto rispetto a quello di uno stateless proxy, vista la complessità delle operazioni da svolgere. Per migliorare le prestazioni di uno stateful proxy sono richieste delle ottimizzazioni particolari, il che comporta un notevole aumento dei tempi di sviluppo, cosa non sempre accettabile.
- La realizzazione di uno stateful proxy è parecchio complessa e necessita di tecniche speciali.

3.6 Outbound proxy.

Spesso gli UAC più semplici inviano le loro richieste sempre ad un proxy (tipicamente vicino) indipendentemente dall'indirizzo del destinatario. Questi proxy sono detti **outbound proxy**. Tipicamente un UA viene configurato manualmente con un indirizzo di un proxy di questo tipo, o può ottenerlo utilizzando protocolli opportuni come DHCP (Dynamic Host Configuration Protocol). Gli outbound proxy sono molto importanti, poiché consentono la realizzazione di UA molto semplici, che non devono occuparsi di come instradare i messaggi e di effettuare delle query DNS (Domain Name System).

3.7 Back-to-Back User Agent.

Un **Back-to-Back User Agent** (B2BUA) è un'entità logica che riceve una richiesta, la elabora come un UAS e, per stabilire come rispondere ad essa, si comporta come un UAC e genera delle richieste. Il comportamento di questo particolare SIP server deve essere stateful. Il B2BUA è simile ad un proxy server, ma possiede un maggior controllo sui dialog in cui è coinvolto e non ha le limitazioni di un proxy, che, ad esempio, non può interrompere una sessione con un BYE o non può modificare a suo piacimento i messaggi gestiti.

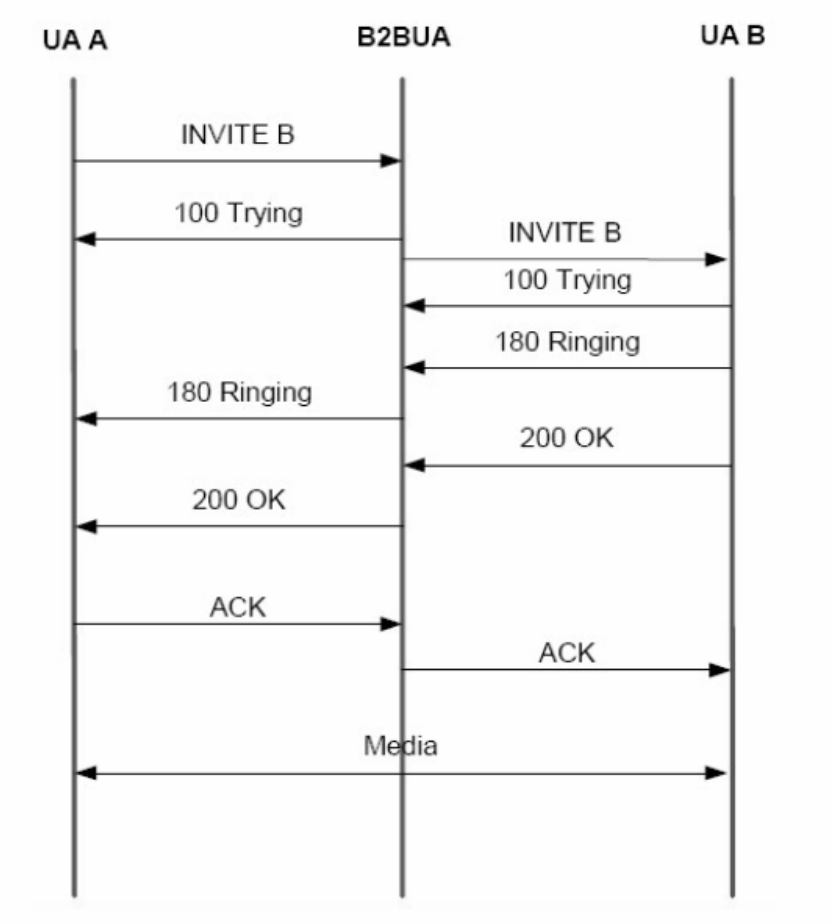

Figura 3.10 Creazione di un dialog tramite un B2BUA.

Il B2BUA viene dunque definito come una concatenazione di UAC e UAS. Poichè un B2BUA si comporta da un lato come un UAS e dall'altro come un UAC, esso ha un maggiore controllo sulle chiamate. Un B2BUA può, infatti, nascondere l'identità di chi ha effettuato la chiamata e modificare i messaggi (non solo gli header, ma anche il loro contenuto, come una descrizione SDP). Quindi, l'uso di B2BUA è particolarmente indicato quando si vuole:

- Nascondere gli utenti o la rete dietro al SIP server.
- Consentire alla rete di interrompere delle chiamate (ad esempio, quando il chiamante ha esaurito il tempo a lui concesso) o di modificarle.
- Consentire alla rete di creare nuovi dialog o di modificare lo stato di quelli esistenti.
- Consentire alla rete di modificare i messaggi in un modo non consentito ai proxy standard (modificare/aggiungere/rimuovere header o contenuto del messaggio, codificare/decodificare i messaggi o parte di essi, comprimere i dati, etc.).
- Tenere traccia dello stato dei dialog.

Lo svolgimento di tali funzioni comporta, ovviamente, una maggiore complessità nella realizzazione di questi server particolari e un peggioramento delle prestazioni rispetto ad un proxy standard.

3.8 Un esempio di chiamata SIP.

Nonostante il protocollo SIP offra anche la possibilità a più utenti di partecipare contemporaneamente ad una conferenza multimediale, per semplicità, sarà descritta solamente la creazione, la modifica e la chiusura di una conversazione a due.

3.8.1 Instaurazione della sessione ("Session Set-up").

Le sessioni SIP sono instaurate attraverso una procedura che si basa sul modello "Three-way-Handshaking". Nell'esempio di figura 3.11 l'utente A vuole instaurare una sessione con l'utente B, lo User Agent A invia una richiesta INVITE allo User Agent B, che contiene nel message body i parametri descrittivi della sessione che vuole creare. Il Proxy SIP, coinvolto nell'instradamento della chiamata, invia la risposta provvisoria "100 Trying", che serve ad inibire la ritrasmissione delle richieste da parte della Client Transaction associata all'UA mittente, e consulta il Location Service per stabilire qual è il contatto associato all'address-of-record per l'User Agent B; ricevuta tale informazione, il Proxy invia la richiesta verso l'User Agent B. L'User Agent B invia dapprima la risposta provvisoria "100 Trying" al proxy, e successivamente la risposta provvisoria "180 Ringing" all'utente A, a conferma della ricezione del messaggio INVITE e ad indicazione che l'utente B è stato notificato dell'arrivo della sessione. Quando la chiamata/sessione è accettata, B invia la risposta definitiva "200 OK", che permette di negoziare i parametri di sessione e scambiare informazioni per la ricezione dei media, come la porta e la tipologia di codec da utilizzare. La procedura "Three-way Handshaking" si conclude quando A invia il messaggio "ACK" (Acknowledgement) all'utente B, che conferma la ricezione della risposta. La

sessione multimediale ha quindi inizio: il trasporto dei flussi multimediali avviene attraverso un altro protocollo di rete, tipicamente RTP.

La destinazione della richiesta è identificata dalla Request-URI e ogni Proxy interessato nell'elaborazione della segnalazione deve risolvere la localizzazione dell'utente e inviare correttamente la richiesta. I Proxy possono modificare i campi del messaggio SIP ad eccezione degli header necessari ad identificare il dialogo. In genere, in mancanza dell'header Record-Route, solo la richiesta che instaura il dialogo attraversa tutta la catena di SIP Proxy, mentre le richieste successive sono scambiate direttamente tra i due User Agent. Alternativamente, un SIP Proxy che vuole rimanere nel percorso della segnalazione successiva all'interno del dialogo, introduce l'header Record-Route contenente la sua SIP URI.

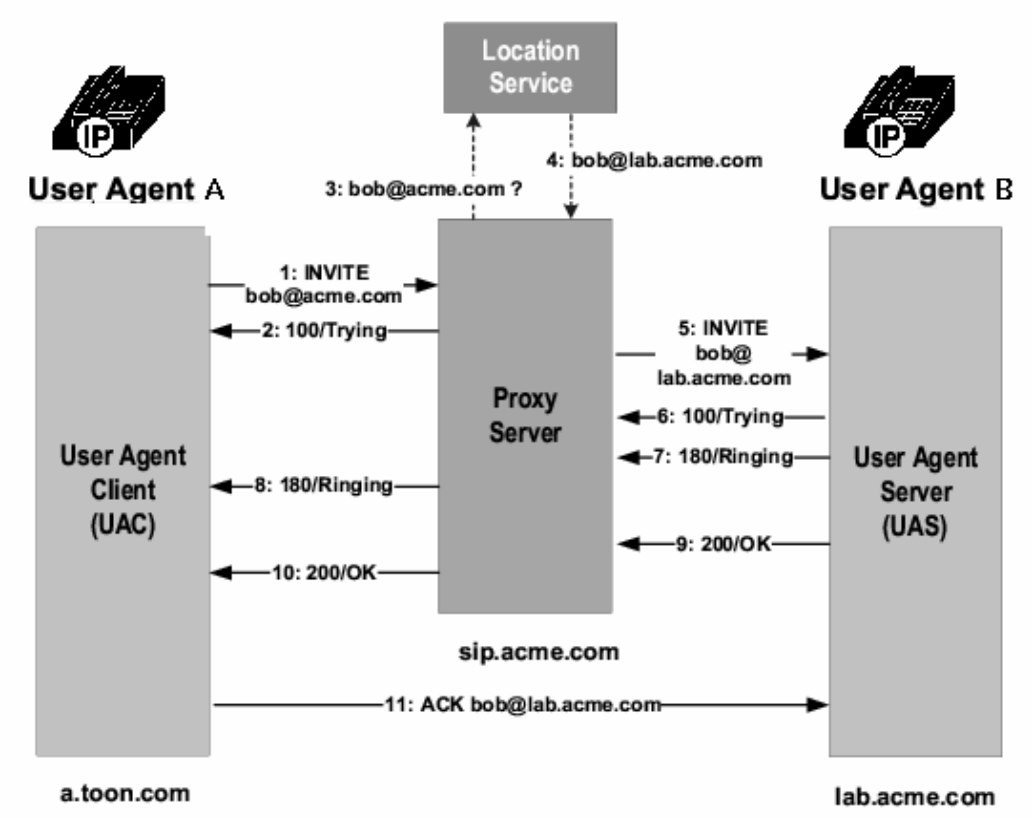

Figura 3.11 Chiamata SIP.

Va rilevato che non è necessario avere un SIP Proxy per instaurare una sessione multimediale tra due entità che dispongono dello Stack protocollare SIP (SIP enabled): due punti terminali SIP-enabled, a conoscenza del reciproco indirizzo IP, possono scambiarsi messaggi SIP per attivare una sessione multimediale.

3.8.2 Abbattimento della sessione ("Session Tear-down").

Dopo aver instaurato una sessione SIP, sia l'utente A che l'utente B possono terminare la sessione emettendo una richiesta SIP BYE. La struttura del protocollo SIP prevede, infatti, che un'User Agent possa svolgere il ruolo di client e server durante la sessione. La richiesta BYE deve essere invita all'interno di un dialogo già stabilito, quindi il messaggio deve contenere i corretti campi ("To tag", "From tag" e "Call-ID") che permettono di identificare il dialogo. La risposta di conferma prevista per il BYE è il 200 OK.

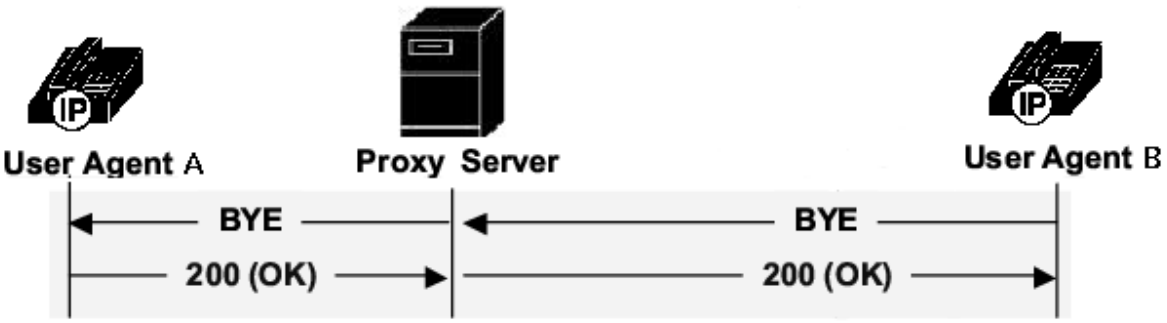

Figura 3.12 Chiusura chiamata SIP.

3.8.3 Modifica della sessione esistente.

Una volta instaurata, una sessione può essere modificata da una nuova sequenza INVITE/OK/ACK, indicata usualmente come re-INVITE con un body

SDP differente. Tale modifica può essere compiuta solo dopo l'ACK della fase iniziale, inoltre il secondo INVITE deve avere gli stessi "From" "To" e "Call-ID" dell'INVITE originale, se la re-INVITE fallisce o non viene accettata la sessione SDP originale rimane valida fino al termine della sessione.

3.9 SIP Event Notification.

SIP è stato ampliato creando un framework estendibile [10] che permette ad un nodo SIP di chiedere ad un nodo remoto notifiche riguardanti certi eventi. I meccanismi di notifica definiti in questo ambito definiscono un'infrastruttura generale per tutte le classi di eventi di sottoscrizione e notifica.

3.9.1 Architettura base.

Le entità interessate al meccanismo di notifica sono:

- **Subscriber:** un sottoscrittore è un user agent che genera e invia richieste di sottoscrizione (SUBSCRIBE) ai notifier e riceve notifiche (NOTIFY) provenienti dai notifier.
- **Notifier:** un notificatore è un user agent che genera richieste di notifiche (NOTIFY) per il subscriber e accetta richieste SUBSCRIBE, per creare le sottoscrizioni.

I metodi SIP utilizzati per effettuare sottoscrizioni e inviare notifiche sono:

- **SUBSCRIBE:** la richiesta inviata da un UA richiede l'esecuzione di un'operazione di sottoscrizione. Le richieste e le risposte 2xx devono contenere entrambe un header "Expires", il cui valore indica la durata della sottoscrizione. Un messaggio SUBSCRIBE con "Expires" nullo è equivalente ad una richiesta di terminazione della sottoscrizione: il

notifier può effettuare una tale operazione nel caso in cui la risorsa in questione non risulta più disponibile.

- **NOTIFY:** notifica al subscriber un cambiamento di stato della risorsa sottoscritta. L'invio di un messaggio NOTIFY al nodo che non ha richiesto alcuna sottoscrizione provoca la generazione di un errore "481 Subscription does not exist".

Un tipico flusso di messaggi potrebbe essere quello di figura 3.13.

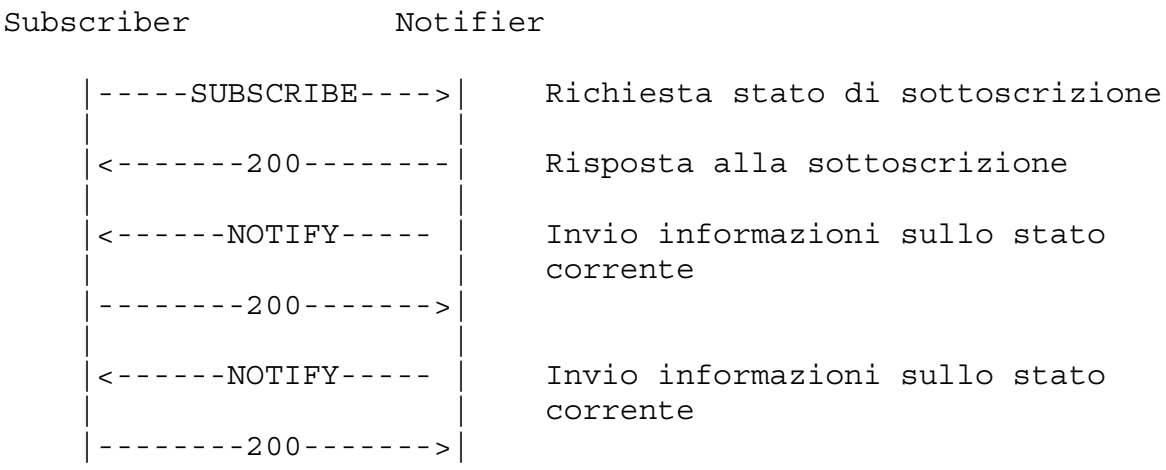

Figura 3.13 Esempio di sottoscrizione e notifica.

3.9.2 Subscriber.

Un user agent subscriber può effettuare diversi tipi di operazioni:

- Richiesta di una sottoscrizione tramite il metodo SUBSCRIBE. La richiesta deve essere confermata da una risposta 2xx, che informa del successo della sottoscrizione e dell'invio della notifica. In particolare una risposta 200 indica che la sottoscrizione è stata accettata e che l'utente è autorizzato a ricevere notifiche, mentre una risposta 202

informa il subscriber che la sottoscrizione non è stata compresa e che l'autorizzazione potrebbe essere stata negata. In caso di insuccesso, il server invia un messaggio non appartenente alla classe 200 per indicare che nessuna sottoscrizione o dialogo è stato creato, e di conseguenza non varrà inviato alcun messaggio di NOTIFY.

- Operazione di refresh: prima che la sottoscrizione diventi inattiva, il subscriber può effettuarne il refresh inviando un altro messaggio allo stesso dialogo con lo stesso parametro "id" nell'header "Event". Il subscriber può ricevere diversi tipi di risposta in base allo stato della sottoscrizione: una risposta 481 indica che la sottoscrizione richiesta è terminata e considerata inutilizzabile, mentre se essa risulta ancora valida il messaggio riporterà nel campo "Expires" o nel parametro "Expires" dell'header "Subscription-State" la durata dell'intervallo di validità.
- Eliminazione di una sottoscrizione inviando una richiesta con un valore nullo nel campo "Expires". Se l'operazione è avvenuta con successo, allora il notifier invierà un messaggio di NOTIFY.
- Conferma della creazione di una sottoscrizione: fino a che il subscriber non riceve il messaggio di NOTIFY, deve considerare la risorsa in uno "stato neutrale. In caso di successo il subscriber deve ricevere il messaggio di notifica prima che la transazione sia completata.

Alla ricezione di una richiesta NOTIFY il subscriber verifica se il messaggio di notifica riguarda una sottoscrizione fatta in precedenza sulla base del dialogo e dell'header "Event": in caso di match negativo, il subscriber ritorna un messaggio "481 Subscription does not exist" o una risposta della classe 400/500, altrimenti invia un messaggio 200 OK.

3.9.3 Notifier.

Un notifier deve poter elaborare la richiesta di SUBSCRIBE inviata dal subscriber e per far ciò deve eseguire una serie di operazioni preliminari:

- Verifica se l'event package del campo "Event" nell'header è comprensibile, altrimenti ritorna una risposta "489 Bad Event".
- Effettua autorizzazioni e autenticazioni in relazione alla propria politica locale.
- Verifica la durata del campo "Expires": se l'intervallo stabilito non è nullo ma è insufficiente, ritorna un messaggio "423 Interval too small".

Dopo aver creato la sottoscrizione e inviato al subscriber il messaggio della classe 2xx, il notifier memorizza il nome dell'event package e il parametro "id" di "Header" se presente. I valori nel campo "Expires" presente nella risposta della classe 200, ha lo stesso comportamento della risposta REGISTER: il server può quindi ridurre l'intervallo ma non può aumentare la dimensione. Non sempre le risposte della classe 200 contengono informazioni sulla durata della sottoscrizione, in quanto le informazioni di stato possono essere comunicate nell'operazione di notifica. Quando il notifier risponde ad una richiesta con un messaggio della classe 200, deve immediatamente costruire e inviare una richiesta NOTIFY al subscriber per dare inizio ad operazioni di autorizzazione e di accordo della politica locale. La richiesta NOTIFY deve contenere un Header "Subscription-State" che può assumere uno dei seguenti valori:

- Active: la sottoscrizione è stata accettata e autorizzata.
- Pending: la sottoscrizione è stata ricevuta, ma le informazioni sulle politiche da adottare sono insufficienti per accettare o negare l'autorizzazione.
- Terminated: la sottoscrizione non è attiva.

Nei primi due casi, l'Header dovrebbe includere un parametro "expires" per indicare il tempo di scadenza della sottoscrizione, mentre nell'ultimo occorre prevedere un header "Subscriber-State" con un parametro "reason" per indicare lo stato della sottoscrizione (es. reason = timeout) o un parametro "retry-after" che specifica il numero di secondi di attesa tra due sottoscrizioni consecutive. Se nell'header è presente sia un valore "terminated" che un codice di "reason", allora occorre distinguere tra diversi casi, ossia "reason" può assumere diversi valori:

- Deactivated: la sottoscrizione è stata terminata e il subscriber non può richiedere immediatamente una nuova sottoscrizione.
- Probation: la sottoscrizione è stata terminata, ma il cliente può riprovare più tardi. Il numero di secondi di attesa è specificato nel parametro "retry-after" quando riportato nella richiesta.
- Rejected: la sottoscrizione è terminata in seguito a modifiche della politica di autorizzazione e il cliente non può effettuare una nuova sottoscrizione.
- Timeout: la sottoscrizione risulta invalida a seguito della scadenza di un timeout e il cliente può inviare un nuovo messaggio SUBSCRIBE.
- Giveup: la sottoscrizione è ritenuta non valida in quanto il notifier non ha ottenuto l'autorizzazione.
- Noresource: la sottoscrizione è stata terminata a causa dell'indisponibilità della risorsa, di conseguenza il cliente non può richiedere una nuova sottoscrizione.

Se la notifica è stata accettata, il subscriber invia un messaggio con un codice 200 che possiede un body solo se nella richiesta NOTIFY era presente l'header "Accept".

3.9.4 Event package.

Sono stati sviluppati diversi package che si basano su questo framework, ogni package ha particolari regole che controllano le sottoscrizioni e le notifiche per gli eventi o classi di eventi che essi gestiscono.

J. Rosenberg ha definito diversi event package, ad esempio un package che permette ad un utente di sottoscriversi ad un altro utente e di ricevere notifiche quando si verifica un cambiamento nello stato di INVITE [11], un altro dedicato all'evento registrazione (REGISTER) [12].

In questa tesi ci siamo occupati di sviluppare un package che permette di velocizzare l'handoff, di avere una bassa percentuale di perdita dei dati durante lo spostamento del terminale, cerca quindi di evitare che ci siano interruzioni nella comunicazione quando un mobile effettua una migrazione. Tale package verrà discusso in seguito.

3.10 Conclusioni.

L'architettura distribuita di SIP permette la realizzazione di nuovi servizi e funzionalità da parte degli utenti, infatti, possono essere implementati definendo nuove estensioni del framework, ossia creando nuovi package con nuovi eventi da gestire, come si vedrà nei prossimi capitoli.

CAPITOLO 4

Stato dell’arte del supporto alla mobilità con SIP e mobile IP.

Con SIP è possibile offrire un supporto a diversi tipi di mobilità [13]: terminal, personal, session e service. In questo lavoro di tesi ci siamo concentrati esclusivamente sulla mobilità del terminale, per completezza nel seguito saranno descritte anche le altre mobilità. Poiché i dispositivi Wi-Fi implementano l’hard handoff, in questo capitolo saranno descritte alcune delle soluzioni esistenti per il fast-handoff intra-dominio (il mobile si muove all’interno di un dominio). Sono state analizzate soluzioni, per il traffico multimediale real-time RTP/UDP, basate su SIP e soluzioni di livello più basso, ossia basate su mobile IP.

4.1 Mobilità e SIP.

Nel seguito sono riportate le diverse mobilità che possono essere gestite con il protocollo SIP.

4.1.1 Terminal mobility.

Con terminal mobility si intende la capacità di un dispositivo di muoversi da una rete ad un’altra, continuando, durante lo spostamento, ad essere raggiungibile da nuove richieste e mantenendo le informazioni di sessione. La terminal mobility coinvolge SIP a tre livelli differenti: pre-call, mid-call e network partition recovery.

4.1.1.1 Pre-call mobility.

Prevede che un mobile host (MH) si sposti prima di ricevere o effettuare una chiamata. In tal caso è sufficiente che MH informi il server registrar della sua home network sulla sua nuova posizione. Dopo essersi spostato, MH indica la sua nuova posizione al redirect server (il servizio di registrazione normalmente è inglobato all'interno dei server proxy e redirect) nella home network. Quando l'host CH manda una richiesta INVITE a questo server è avvisato dello spostamento di MH e gli è fornita la sua nuova locazione. In questo modo CH può contattare MH nella foreign network.

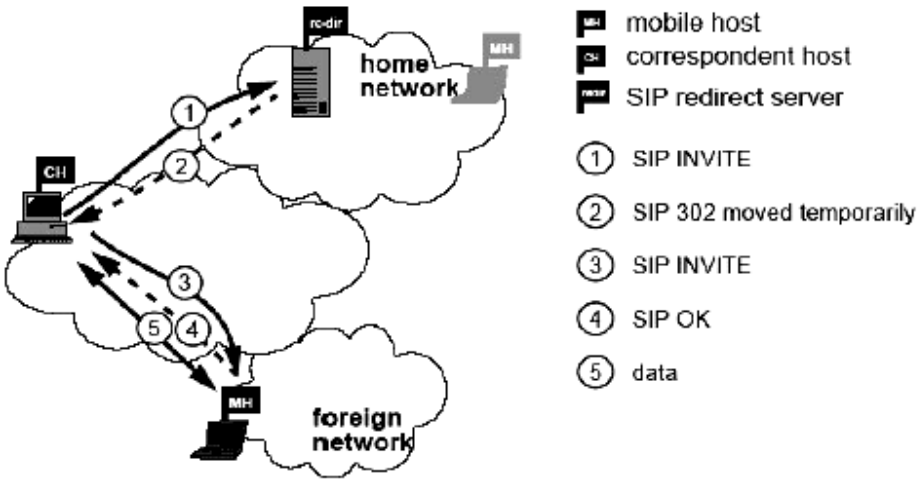

Figura 4.1 Esempio di pre-call mobility.

4.1.1.2 Mid-call mobility.

La mid-call mobility prevede che l'host MH si sposti quando una sessione è già stata stabilita ed è in corso un trasferimento di dati tra i terminali. In questo caso MH avvisa del suo spostamento l'endpoint con cui sta comunicando(CH)

inviandogli direttamente una richiesta Re-INVITE. In tale messaggio è inserita una descrizione aggiornata della sessione che contiene il nuovo indirizzo (host o IP) di MH.

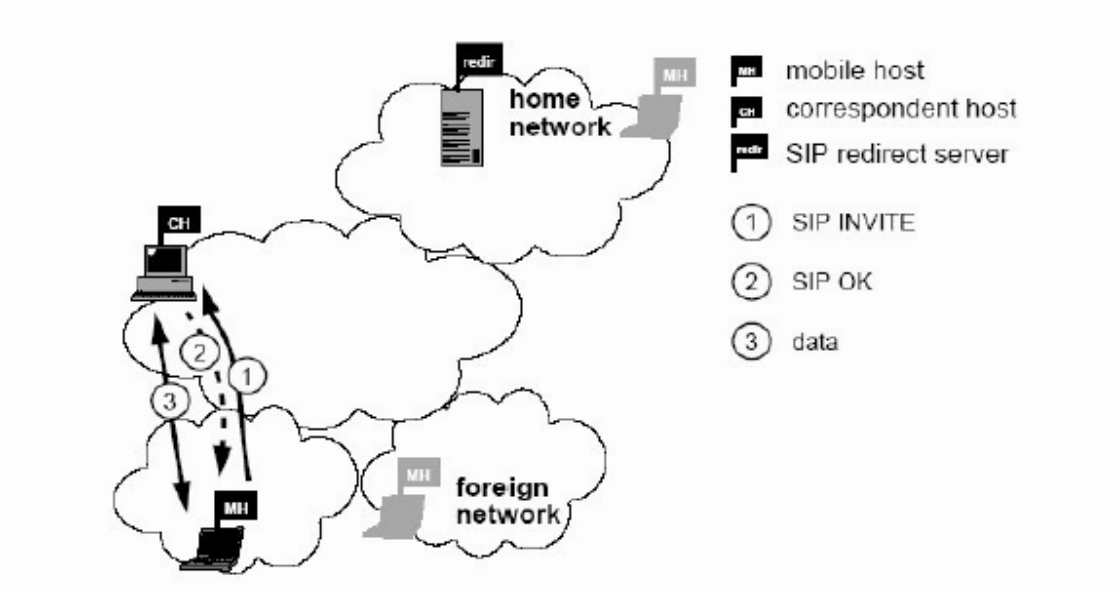

Figura 4.2 Esempio di mid-call mobility.

Il messaggio 1 della figura 4.2 può avere la seguente forma:

```
INVITE sip:alice@correspondent.com SIP/2.0
Via: SIP/2.0/UDP mh.current.location:5060
From: sip:betty@home.com
To: sip:alice@correspondent.com
Subject: a mobile session
Contact: betty@mh.current.location
CSeq: 781769870 INVITE
Call-ID: <call-id of ongoing session>
Content-Length: ...

v=0
o=betty916340046 916340046 IN IP4 mh.current.location
t=2208988800 2208988800
c=IN IP4 mh.current.location
m=audio 50000 UDP 0
```

Nell'esempio sopra riportato, l'indirizzo della nuova locazione di MH (cioè Betty) è rappresentato da "mh.current.location" ed è riportato nel Contact header e nei campi "o" (indica colui che dà origine alla sessione, fornisce inoltre un session id e un numero di versione della sessione) e "c" (indica il tipo di rete, il tipo di indirizzo e l'indirizzo di connessione per tale sessione che si vuole iniziare o modificare) della descrizione SDP della sessione. L'header From, invece, contiene l'indirizzo regolare di Betty, cioè betty@home.com, poiché questo campo è utilizzato per l'identificazione del mittente. Dopo essersi spostato, MH comunica la sua nuova posizione al server SIP nella home network, in modo che le nuove chiamate possano essere indirizzate correttamente (figura 4.3).

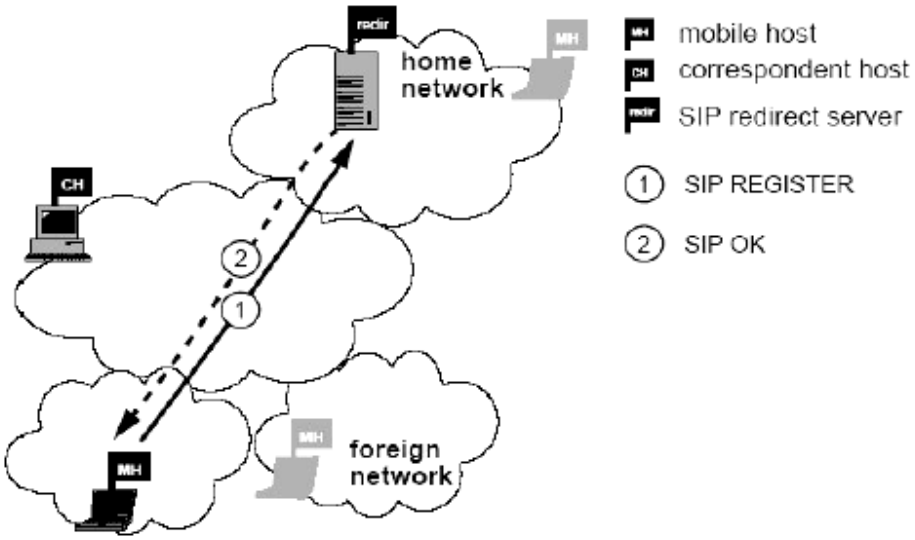

Figura 4.3 Aggiornamento della posizione dopo lo spostamento del mobile host.

In questa tesi viene affrontato il problema della mid-call mobility, ossia lo spostamento di un utente durante il trasferimento di dati tra i due utenti che hanno

instaurato una sessione, evidenzieremo le soluzioni per la micro-mobility (spostamento all'interno di una stessa sottorete), per la macro-mobility (spostamento tra due sottoreti dello stesso dominio), per la global-mobility (spostamento tra due domini).

4.1.1.3 Network partition.

Una partizione della rete può causare un temporaneo blackout delle trasmissioni. Se l'interruzione ha una durata inferiore ai 30 secondi, SIP è in grado di ripristinare il corretto funzionamento del sistema mediante le ritrasmissioni delle richieste che non hanno ricevuto risposta. Se, invece, l'interruzione dura di più, è possibile che CH si ritrovi con un indirizzo dell'host mobile MH non più valido. Per uscire da questa condizione di errore è necessario un meccanismo di fall-back.

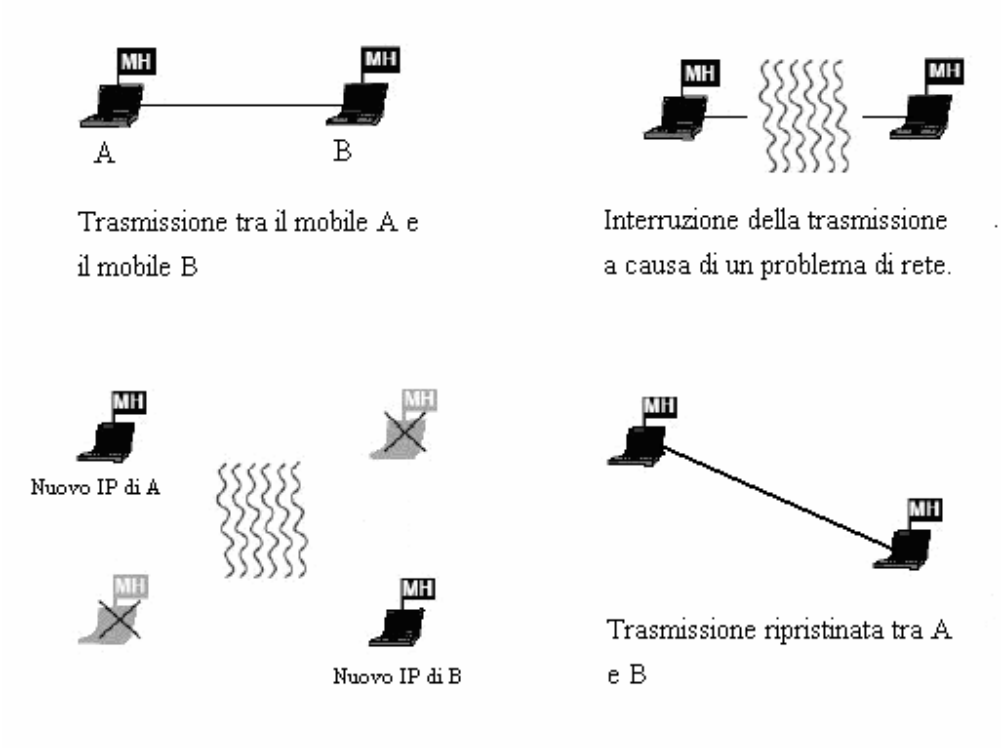

Figura 4.4 Esempio di network partition.

La figura 4.4 mostra due host mobili che comunicano tra loro; entrambi perdono il contatto per un certo intervallo di tempo (ad esempio, a causa dell'attraversamento di un tunnel), e quando la rete torna a funzionare entrambi hanno un nuovo indirizzo IP. Per evitare situazioni di questo tipo, un host può inviare ritrasmissioni delle richieste INVITE anche al SIP server nella home network dell'host mobile. Dato che tale server ha un indirizzo fisso, MH può sempre inviargli delle registrazioni. In questo modo CH può individuare nuovamente il compagno perduto, cioè MH. Per ridurre i tempi di localizzazione, un host può inviare una richiesta INVITE contemporaneamente all'ultimo indirizzo IP noto di MH e al suo home registrar.

4.1.2 Session mobility.

La session mobility consente ad un utente di mantenere una sessione multimediale anche quando sono cambiati i terminali utilizzati. Ad esempio il chiamante potrebbe voler continuare una sessione, iniziata su un terminale mobile, su il PC dell'ufficio dopo essere arrivato sul posto di lavoro. L'utente potrebbe, inoltre, avere l'esigenza di spostare solo una o più parti di sessione, per esempio potrebbe essere comodo dirottare la parte video della sessione su un video proiettore. La session mobility può essere garantita con SIP in due modi differenti: third-party call control e meccanismo REFER.

Negli esempi si assume che Alice ha instaurato una sessione con Bob, che utilizza un terminale mobile ("bob@mobile") e che vuole trasferire la sessione su un host fisso ("bob@fixed").

Nella figura 4.5, Bob invia una richiesta INVITE (1) a "bob@fixed", cioè la nuova destinazione della sessione, indicando (negli header del messaggio) i parametri della sessione, tra cui l'indirizzo IP di Alice. "bob@mobile" inoltra (3) ad Alice la descrizione di sessione generata da "bob@fixed" (2). Alice, a sua volta, invia la

descrizione della propria sessione (4) a “bob@fixed” (5) utilizzando come tramite “bob@mobile”. Bob può, inoltre, dividere la sessione in diverse parti, ognuna con un ricevente diverso secondo il tipo di media trasmesso.

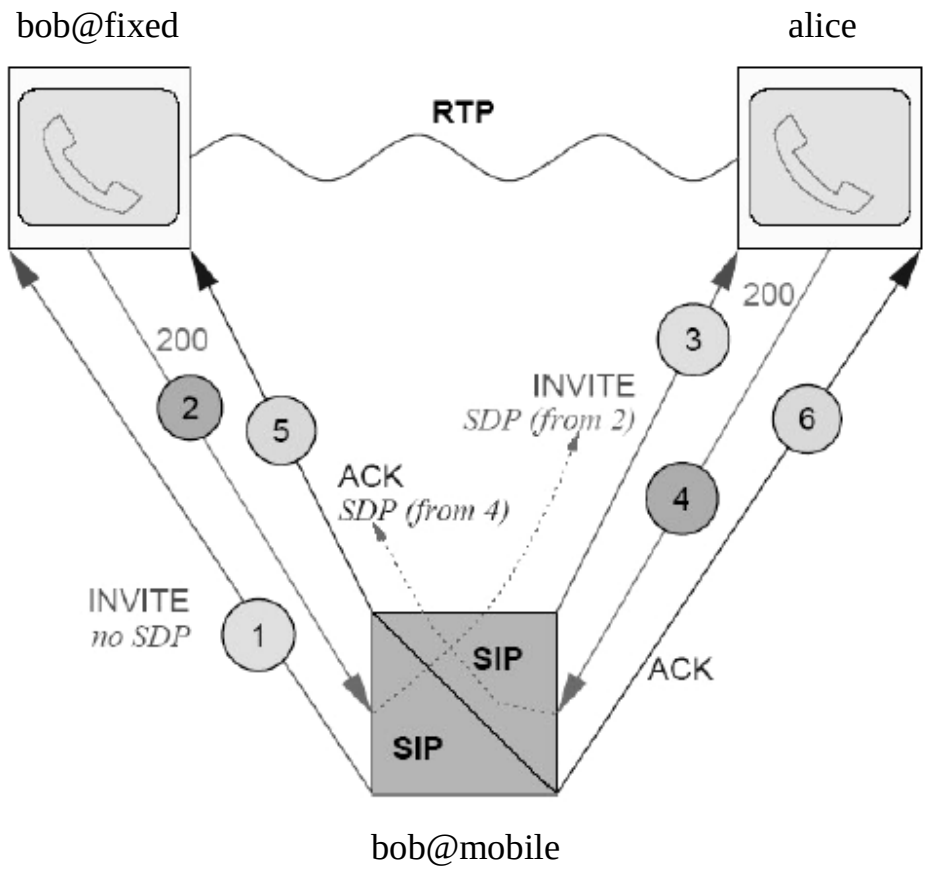

Figura 4.5 Session mobility con il third-party call control.

Lo svantaggio di questo approccio consiste nel fatto che bob@mobile rimane coinvolto nella sessione finché non è contattato direttamente per terminare la chiamata.

Il secondo metodo, figura 4.6, per garantire la session mobility trasferisce esplicitamente la sessione sulla nuova destinazione. In questo caso “bob@mobile” (B1) invia semplicemente una richiesta REFER ad Alice (1), indicando il nuovo destinatario “bob@fixed” (B2). In seguito Alice negozia una sessione con l’host fisso mediante una richiesta INVITE regolare (2). Infine B1 conclude la sessione con Alice inviandole un BYE (3). Un’alternativa al modello presentato prevede che Bob possa inviare un messaggio REFER anche all’host fisso B2, chiedendogli di mandare un INVITE ad Alice. Qualora Bob voglia suddividere la sessione su host differenti secondo il tipo di media trasmesso, egli deve indicare ad Alice ogni nuovo terminale mediante differenti richieste REFER.

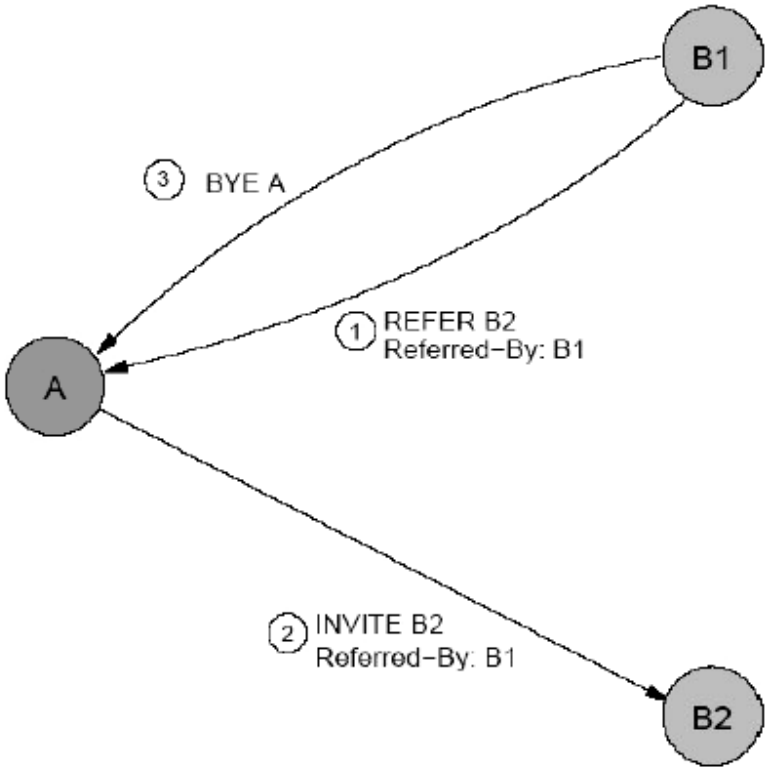

Figura 4.6 Session mobility mediante il trasferimento della chiamata.

4.1.3 Personal mobility.

La personal mobility consente di identificare un singolo utente localizzato su terminali differenti mediante lo stesso indirizzo logico. La figura 4.7 mostra una corrispondenza sia 1 a n (un indirizzo, diversi possibili terminali), sia m a 1 (più indirizzi che si riferiscono allo stesso terminale). Alice possiede un telefono tradizionale, un PC e un dispositivo wireless. Questi tre terminali possono essere utilizzati da Alice tutti contemporaneamente o in modo alternato. Attraverso il forking (operazione mediante la quale un proxy può decidere di inviare la richiesta a più indirizzi di destinazione), offerto dai proxy stateful, Alice può essere raggiunta su ogni terminale mediante lo stesso indirizzo logico, il che consente di rendere la scelta di tali dispositivi completamente trasparente agli endpoint con cui l'utente comunica. Un problema che si riscontra nella pratica, quando si deve realizzare una situazione di questo tipo, consiste nel fatto che i server registrar devono essere in grado di distinguere in qualche modo i differenti terminali che sono associati alla stessa persona.

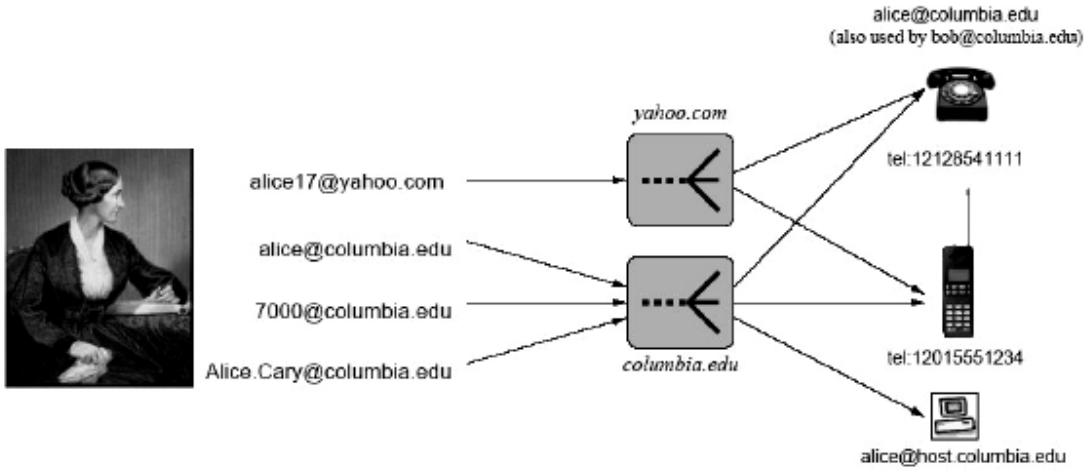

Figura 4.7 Un esempio di personal mobility.

4.1.4 Service mobility.

La service mobility consente agli utenti di mantenere l'accesso ai propri servizi anche durante gli spostamenti tra una rete ed un'altra, o tra differenti dispositivi. In un ambiente voice over-IP, ad esempio, gli utenti potrebbero volere conservare le liste di digitazione veloce dei numeri telefonici, le rubriche, i registri delle chiamate, e così via. Una soluzione a questo tipo di mobilità consiste nel costringere l'utente a portare con se tutte le informazioni di cui ha bisogno, utilizzando un PDA o una scheda di memoria. In ogni caso rimane il problema di sincronizzare i vari dispositivi dell'utente, che non è detto che si trovino tutti nello stesso posto, ogni volta che sono effettuate delle modifiche alle informazioni su un terminale. SIP offre un meccanismo di base con cui è possibile sincronizzare questo tipo di dati tra diversi server. Il modello cui si fa riferimento prevede l'esistenza di un home server associato all'indirizzo dell'utente. Ad esempio, l'utente identificato da "alice@wonderland.com" può utilizzare, per memorizzare le informazioni di servizio, il server SIP che si occupa del dominio wonderland.com. Le applicazioni SIP si registrano presso un registrar generalmente una volta l'ora oppure ogni volta che cambia l'indirizzo di rete. La registrazione contiene tre tipi di informazioni: l'indirizzo della rete corrente, le proprietà del dispositivo (come la lingua parlata dall'utente, i media supportati, ecc.) e uno o più parametri di configurazione decisi dall'utente. Uno UA (user agent) invia al registrar anche il timestamp relativo alle informazioni di configurazione. In questo caso il server o aggiorna le informazioni che possiede o restituisce una versione più recente dei dati nella risposta alla registrazione. Tutto ciò prevede che ci sia solo un server che gestisce tali informazioni per un dato utente in modo da evitare problemi di duplicazione.

4.2 Soluzioni basate su Mobile IP.

Prima di analizzare una delle diverse soluzioni sviluppate utilizzando la mobile IP, viene descritto, facendo riferimento alla figura 4.8, brevemente il funzionamento base di tale protocollo [14]:

- Il mobile host (MH) giunto nella foreign network (FN) si registra presso un foreign agent (FA).
- MH acquisisce l'agente FA.
- L'home agent (HA) di MH viene aggiornato in modo tale da poter inviare tutti i pacchetti destinati a MH al nuovo FA.
- L'host A contatta HA, che fa da tramite per MH.
- HA incapsula il pacchetto proveniente da A in un nuovo datagramma IP e lo inoltra a FA (operazione di tunneling).
- FA riceve il pacchetto e lo invia a MH.
- La risposta viene inviata direttamente da MH a A.

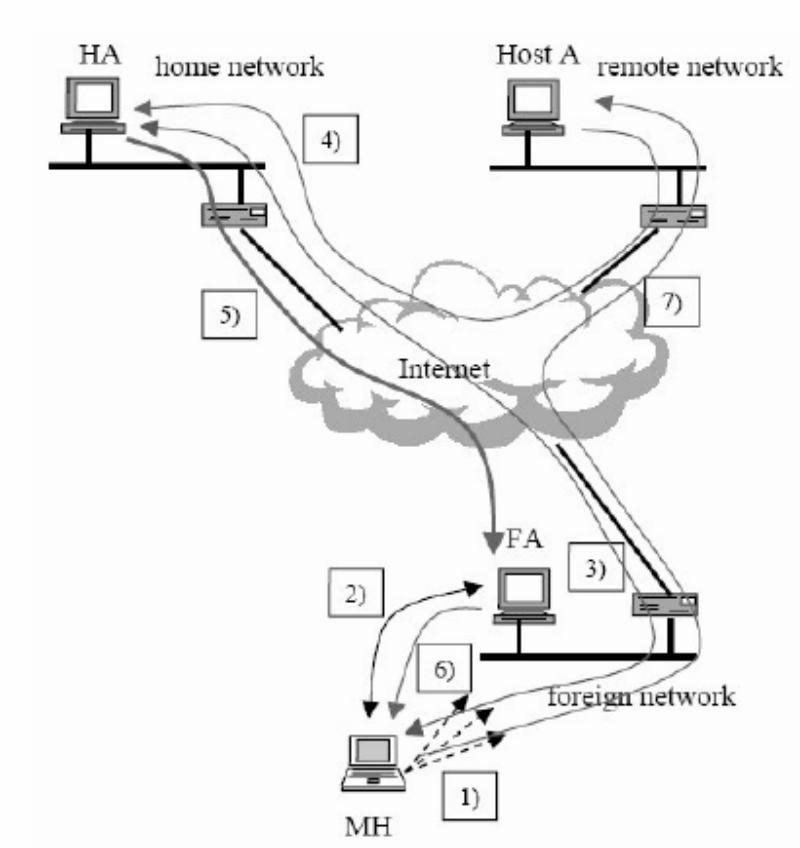

Figura 4.8 Schema di funzionamento di IP mobile.

Una delle soluzioni introdotte per cercare di ridurre il carico della segnalazione e il ritardo nella home network durante i movimenti all'interno di un dominio è la MIP-RR.

4.2.1 Mobile IP Regional Registration/Hierarchical Mobile IP.

MIP-RR [14] cerca di ridurre il numero di messaggi di segnalazione nella home network e di ridurre il ritardo dovuto alla segnalazione eseguendo le registrazioni localmente, in una regional network. Quando un nodo mobile (MN) arriva per la prima volta in una regional network, esegue una registrazione insieme al suo HA. Durante la registrazione HA registra il care-of address (CoA) di MN

presso un GFA ossia un gateway foreign agent. Quando un MN cambia FA all'interno della stessa regional network, esegue solo una regional registration presso GFA per modificare il suo CoA. Quando si muove da una regional network ad un'altra, esegue invece una home registration con il suo HA.
I pacchetti per MN sono prima intercettati dal suo HA, che li invia a GFA. GFA controlla la sua lista dei visitatori e invia i pacchetti al corrispondente FA di MN. FA invia poi i pacchetti a MN.
GFA introduce un livello di gerarchia tra HA e FA di MN. L'uso di GFA evita la segnalazione a HA finché MN è all'interno della regional network.

4.3 Soluzioni basate su SIP per servizi multimediali.

Prima di descrivere le soluzioni basate su SIP viene introdotto lo scenario in cui si sta operando. Per ogni movimento che il mobile compie all'interno di un dominio [15] invia un Re-INVITE al Correspondent Host in modo che il nuovo traffico sia diretto alla nuova destinazione del mobile. A causa della distanza tra CH (Correspondent Host) e MH (Mobile Host) e di una possibile congestione, SIP Re-INVITE potrebbe ritardare. Di conseguenza, durante questo ritardo, il traffico transiente è inviato alla vecchia destinazione e quindi non ricevuto dal mobile nella sua nuova posizione. Per evitare o almeno ridurre la perdita di questi dati ci sono diverse alternative, che vedremo di seguito. Prima di descrivere le diverse soluzioni proposte da Dutta, verranno introdotti il protocollo RTP e RTP translator.

4.3.1 RTP (Real-time Transport Protocol).

Real-time Transport Protocol è stato definito dall'IETF [16], fornisce i meccanismi di base per il trasferimento di dati multimediali (audio e/o video). E' stato suddiviso in due sottoprotocolli:

- RTP per trasportare dati con proprietà di tempo reale.
- RTCP (RTP control protocol) per "monitorare" la qualità del servizio e fornire informazioni sui partecipanti di una sessione in atto.

Le funzioni fornite da RTP e RTCP sono:

- Identificazione del tipo di payload (codifica).
- Gestione di numeri di sequenza.
- Gestione del timestamping.
- Servizi di monitoraggio e analisi delle prestazioni.
- Supporto identificazione partecipanti.

Una sessione RTP permette ad un certo numero di utenti di comunicare mediante l'uso del protocollo RTP, ossia attraverso lo scambio di pacchetti RTP. Una sessione è identificata da una coppia di indirizzi di trasporto:

- Un indirizzo di trasporto è costituito da un indirizzo IP e da una porta UDP.
- Un indirizzo di trasporto serve per RTP, l'altro per RTCP, ma l'indirizzo IP di tipo multicast è lo stesso per entrambe le coppie.

I pacchetti RTP vengono incapsulati in segmenti UDP, così come i pacchetti RTCP ma usano una porta diversa.

La figura 4.9 mostra il formato di un pacchetto RTP, di seguito viene riportato il significato di ogni campo dell'intestazione :

- V (versione) (2 bit): versione protocollo RTP utilizzata.
- P (padding) (1 bit): esistenza di padding nei dati.
- X (eXtension header) (1 bit): indica la presenza o assenza di un header extension alla fine dell'intestazione.
- CC (CSRC count) (4 bit): numero di campi CSRC presenti nell'header.
- Marker (1 bit): può essere usato per indicare estremi di un fotogramma.

- PT (Payload Type) (7 bit): indica il tipo di codifica usato nel payload del pacchetto.
- Sequence Number (16 bit): sequenza monotonica crescente (+1 per ogni RTP PDU).
- Timestamp (32 bit): si riferisce all'istante di creazione del primo campione contenuto all'interno del payload.
- SSRC (32 bit): identificativo della sorgente che ha creato il contenuto del payload. L'identificativo è scelto a caso dalla sorgente.

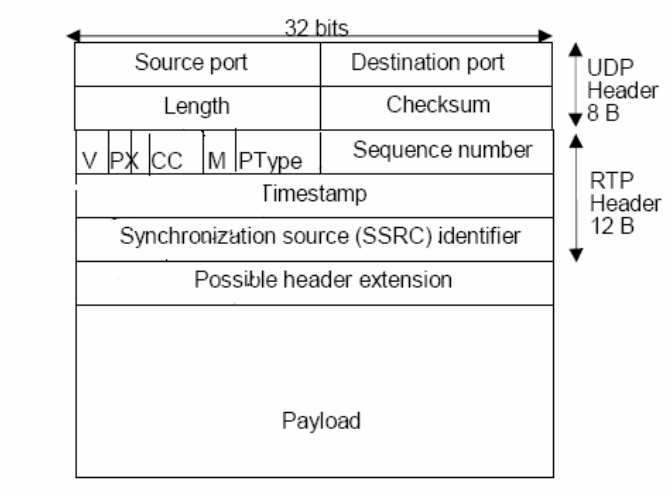

Figura 4.9 Formato pacchetto RTP.

Ogni partecipante ad una sessione invia periodicamente pacchetti di controllo RTCP a tutti gli altri partecipanti della sessione RTP.

4.3.2 RTP translator e RTP mixer.

Il protocollo RTP [17] prevede la possibilità di usare due entità dette Mixer e Translator.

- Se alcuni partecipanti hanno connessioni a bassa larghezza di banda, un Mixer collocato opportunamente può cambiare la codifica dei dati e ridurne la qualità**,** e riunire più stream audio/video in uno solo per non congestionare la rete lenta, ma mantenere la qualità per gli altri partecipanti. Lo stream ottenuto potrà essere multicast o unicast verso diversi processi. Il Mixer inserisce un suo identificatore come agente di sincronizzazione, ma mantiene le informazioni sui mittenti.
- Invece i translator sono dei traduttori di codifiche, modificano il tipo di codifica di un flusso e lo ritrasmettono sulla rete. Possono anche funzionare come gateway, infatti in questo capitolo vengono usati con questa funzione, ossia intercettano i pacchetti RTP e li inviano alla locazione corrente del mobile host. Il translator può effettuare operazioni di buffering per evitare interruzioni del flusso dati (in questo caso tutti i pacchetti duplicati sono individuati a livello RTP), inoltre, può anche offrire un servizio di correzione di errori e di adattamento della trasmissione alla banda disponibile, comportamento, quest'ultimo, simile a quello dei mixer, ma i translator non devono gestire anche la sincronizzazione tra flussi correlati.

4.3.3 SIP registrar e RTP translator.

In [15] viene suggerito un approccio di livello applicativo che utilizza SIP e RTP Translator per porre attenzione ai dati transienti e al fast-handoff.

La Figura 4.10 mostra la sequenza di operazioni che vengono eseguite quando un host mobile si muove da una rete ad un'altra. Ogni sottorete all'interno di un dominio è dotata di un RTP translator che fornisce l'invio a livello applicativo dei pacchetti RTP. Qui il SIP server può agire come un registrar o come un proxy. Il mobile dopo essersi spostato da una sottorete ad un'altra invia un messaggio di

registrazione al SIP server (registrar) per aggiornarlo della sua nuova posizione. Il registrar dopo aver ricevuto la registrazione aggiornata da MH, invia una richiesta al RTP translator nella sottorete che MH ha appena lasciato. La richiesta induce l'RTP translator a inoltrare i dati in arrivo (formato RTP) al vecchio indirizzo IP utilizzato da MH al nuovo indirizzo del mobile. Dopo un certo intervallo o dopo che RTP translator non ha ricevuto più pacchetti, si assume che il pacchetto Re-Invite abbia raggiunto il Correspondent Host, così RTP translator termina, terminando così l'operazione di handoff.

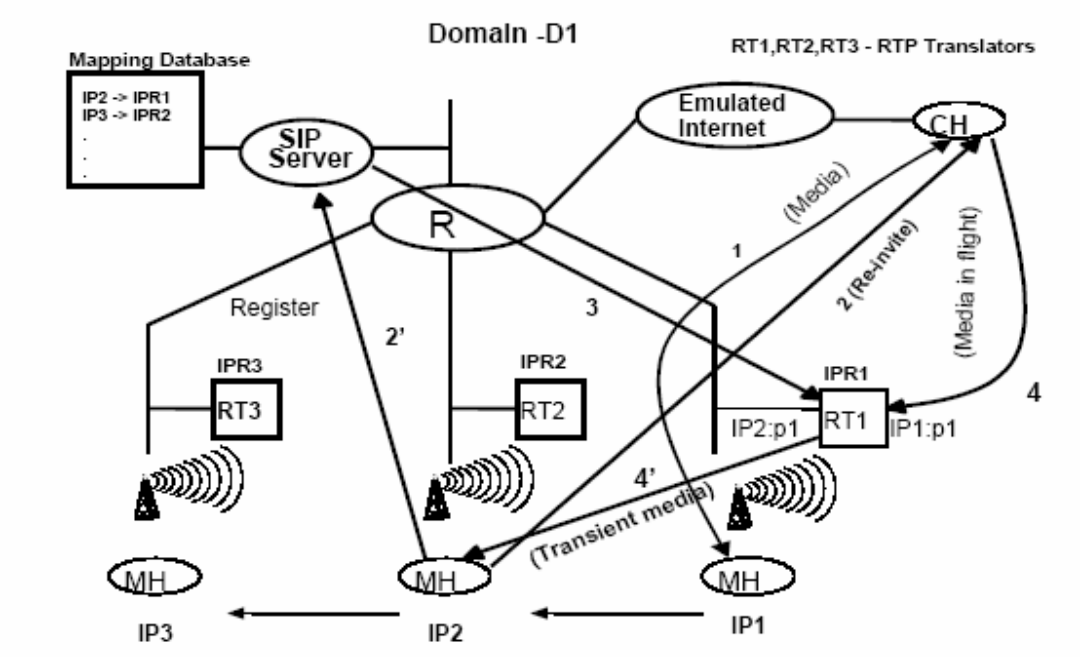

Figura 4.10 Fast-handoff basato su RTP translator.

Il diagramma di Figura 4.11 mostra il flusso per l'operazione di fast-handoff utilizzando l'approccio RTP translator. Tale soluzione migliora certamente la perdita dei dati transienti ma non totalmente, in quanto durante lo spostamento del

mobile da una sottorete ad un'altra e fino a che il registrar della rete visitata non avverte l'RTP translator, della sottorete che il mobile ha appena lasciato, MH può perdere parte dei pacchetti a lui destinati.

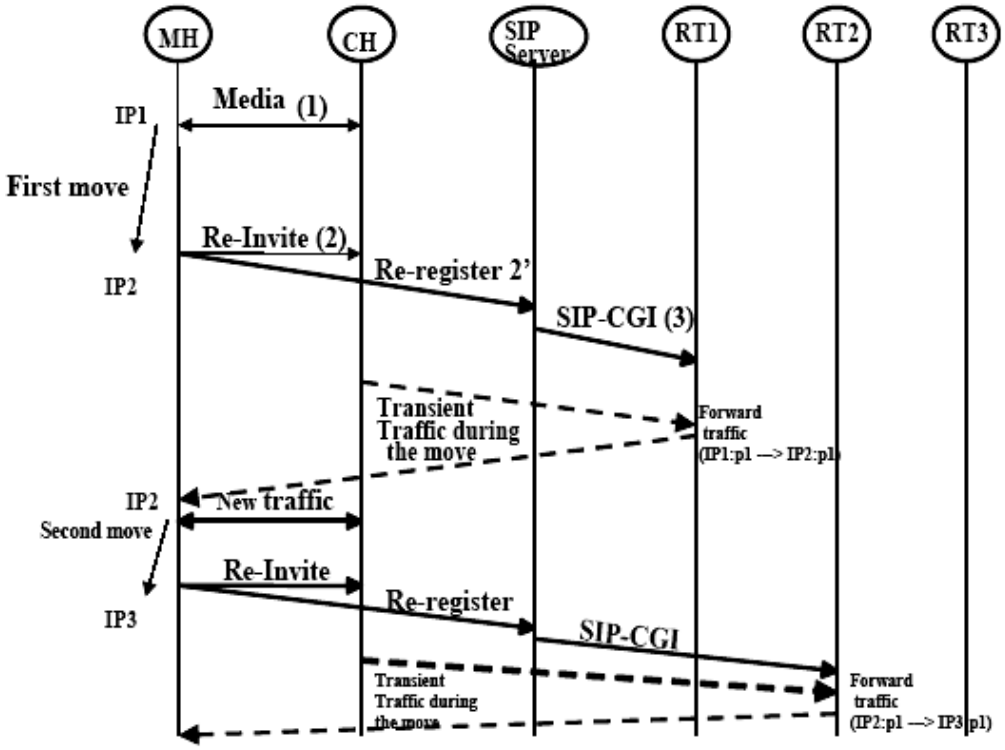

Figura 4.11 Flusso Fast-handoff.

4.3.4 SIP outbound proxy.

Le richieste SIP tipicamente attraversano un SIP proxy nella rete visitata, ossia un outbound proxy. L'outbound può utilizzare i dati che si trovano nel Re-INVITE che è spedito da MH a CH per configurare RTP translator o NAT (è possibile combinare questa soluzione insieme alla precedente).

Il vantaggio di questo approccio [15] è che l'outbound proxy di solito ha accesso alle informazioni SDP contenenti l'indirizzo e la porta del media MH, semplificando di conseguenza la configurazione di RTP translator o NAT. Questo outbound proxy ricorda, per una certa quantità illimitata di tempo, le informazioni presenti in INVITE e diventa chiamata stateful, visto che necessita della vecchia informazione quando un nuovo Re-INVITE è fornito da MH.
Anche con questa soluzione si ha la perdita di una parte dei dati durante lo spostamento, anche se la scelta dell'outbound proxy permette di creare un MH e un CH più semplici e con meno responsabilità.

4.3.5 B2BUA approach.

Un'altra tecnica [15] per fornire il fast-handoff è utilizzare il back-to-back SIP user agent (B2BUA). Il B2BUA è costituito da due SIP user agent, in cui uno user agent riceve una richiesta SIP, possibilmente trasforma i parametri SDP, e l'altro user agent rinvia la richiesta. In ogni dominio un B2BUA necessita di essere indirizzato da MH nel dominio visitato. Il B2BUA emette una richiesta al CH contenente il proprio indirizzo come destinazione dei pacchetti e poi invia questi ultimi, attraverso un RTP translator, a MH. Quando il mobile si muove in una nuova sottorete invia un messaggio di Re-INVITE al B2BUA.

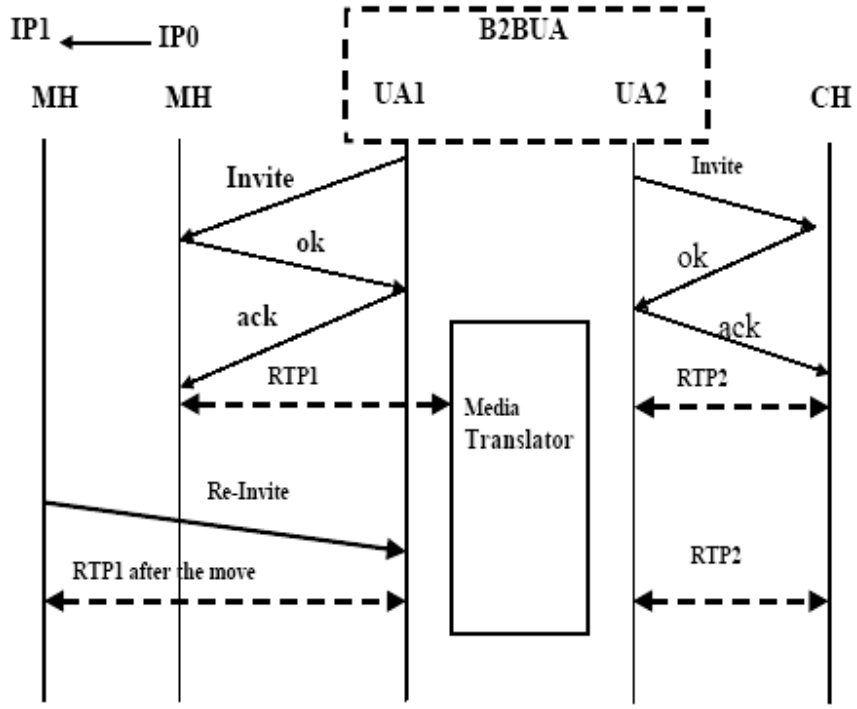

Figura 4.12 Flusso fast-handoff con B2BUA.

4.3.6 Multicast Agent e B2BUA.

Il multicast [15] può aiutare a limitare la perdita di dati se MH può predire quale sarà il suo movimento nella nuova sottorete tempestivamente. In questo caso, informa il registrar del dominio visitato o il B2BUA di un indirizzo multicast temporaneo come suo contatto. Nel momento in cui MH arriva nella sua nuova sottorete, aggiorna il registrar o B2BUA con il suo nuovo indirizzo unicast, continuando anche ad ascoltare l'indirizzo multicast.

L'utilizzo del multicast è efficace solo se MH può velocemente acquisire un indirizzo multicast e prevedere con una certa esattezza la sottorete di destinazione.

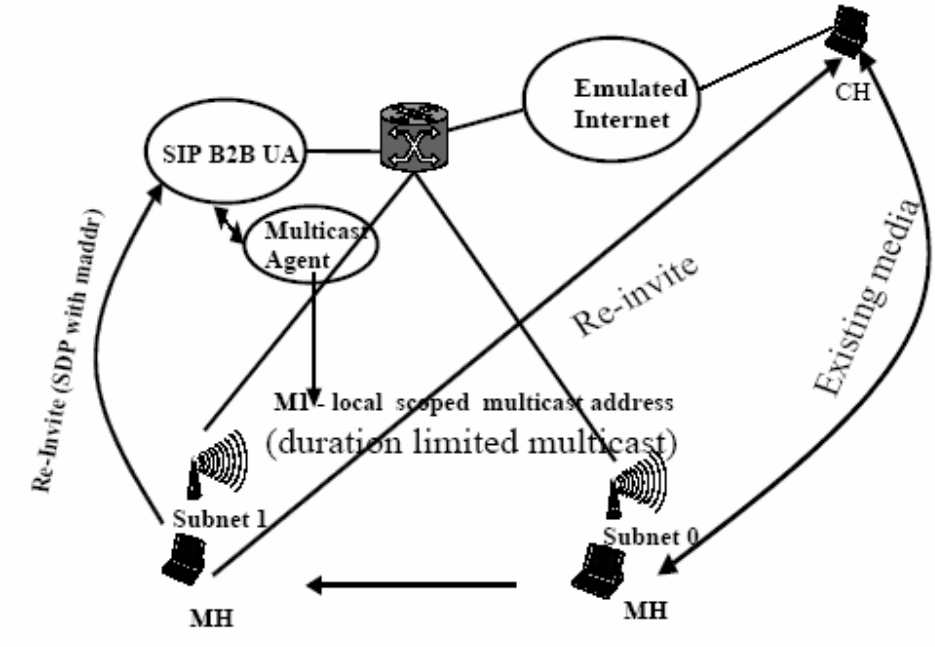

Figura 4.13 SIP fast-handoff con agent multicast.

4.4 Conclusioni.

La soluzione IP mobile è particolarmente adatta a tutte quelle situazioni in cui è necessario trasferire dei dati tramite connessioni TCP di lunga durata, mentre per connessioni di durata breve è preferibile usare SIP perché fornisce un supporto a livello applicativo, senza problemi di overhead sia in termini di tempo (problema del triangle routing dovuto all'home agent) sia in termini di dimensione dei pacchetti inviati (a causa del tunneling effettuato da HA). A differenza del mobile IP, la mobilità SIP-based non richiede funzionalità aggiuntive ai sistemi operativi, né l'installazione di home agent.

Le soluzioni descritte continuano a produrre, tuttavia, perdita di dati transienti, in questa tesi è stata proposta una nuova soluzione che riduce sensibilmente la perdita di dati, come sarà descritto nel capitolo seguente, attraverso l'introduzione di due livelli di bufferizzazione, lato client e lato proxy, di una tecnica di predizione dell'handoff, il tutto gestito a livello applicativo attraverso il protocollo SIP.

CAPITOLO 5

Analisi e progettazione della segnalazione durante l'handoff.

In questo lavoro di tesi sono stati affrontati i problemi derivanti dalla terminal mobility, in particolar modo il processo di handoff e la gestione della continuità di sessione/comunicazione durante gli spostamenti di un utente da una cella Wi-Fi ad un'altra. Sono state così sviluppate diverse soluzioni per velocizzare l'handoff, diminuire la perdita dei dati transienti ed evitare l'interruzione della comunicazione. In particolare, ci siamo concentrati su una soluzione di livello applicativo con l'intento di proporre alcune estensioni SIP per la gestione di handoff veloci. La scelta di gestire l'handover a livello applicativo è stata fatta in quanto si hanno così soluzioni specifiche per le diverse possibili applicazioni che si stanno fornendo/utilizzando e per poter operare indipendentemente dalla tecnologia utilizzata dai livelli sottostanti. La nostra soluzione è stata sviluppata in modo tale da poter essere utilizzata in qualsiasi ambiente e supportare sia l'handoff verticale sia l'handoff orizzontale.

5.1 Hard handoff.

Per poter introdurre le soluzioni proposte bisogna precisare che una wireless internet è costituita da diverse reti. Ogni rete, di solito, copre una larga area geografica ed è composta a sua volta da tante altre sottoreti, in ogni sottorete possiamo avere una o più base station, chiamate anche access point (AP).

I dispositivi Wi-Fi, come specificato nel secondo capitolo, realizzano un handoff di tipo hard, il dispositivo mobile non è capace di interagire con due base station contemporaneamente, infatti, interrompe completamente la connessione con la precedente base station prima di stabilirne una nuova con un'altra, minimizzando così l'overhead di segnalazione, ma incrementando la latenza e la perdita dei pacchetti. Nell'ambito dell'hard handoff è stata fatta una classificazione delle possibili tipologie che si possono avere [18] micro, macro e global handoff:

- Micro handoff (handoff intra-subnet), il cliente si muove tra due differenti AP (o base station) della stessa sottorete, senza cambiare indirizzo IP. Può produrre perdita di pacchetti. A livello applicativo è percepito come una disconnessione temporanea: il tempo di disconnessione non è trascurabile, molto variabile e dipendente dall'implementazione della card Wi-Fi del cliente.

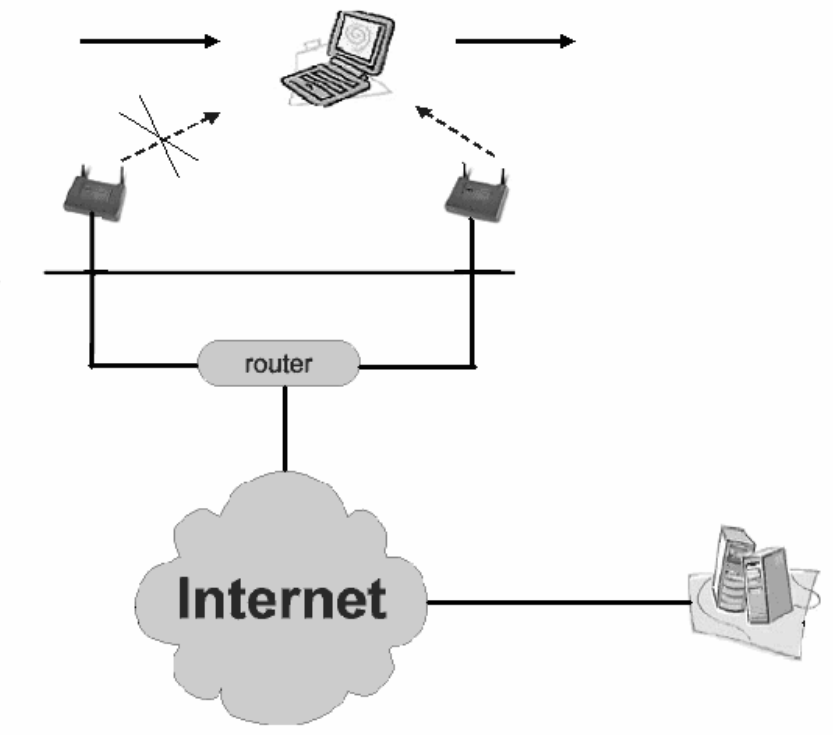

Figura 5.1 Micro-handoff.

- Macro handoff (handoff intra-dominio), il cliente si muove tra due AP appartenenti a due sottoreti diverse dello stesso dominio, provocando il cambiamento dell'indirizzo IP del cliente. Il macro handoff ha una durata superiore rispetto al micro handoff, poiché il cliente deve acquisire anche un nuovo indirizzo nella sottorete di destinazione attraverso, ad esempio, il server DHCP che richiede diversi secondi per terminare la configurazione dell'indirizzo.

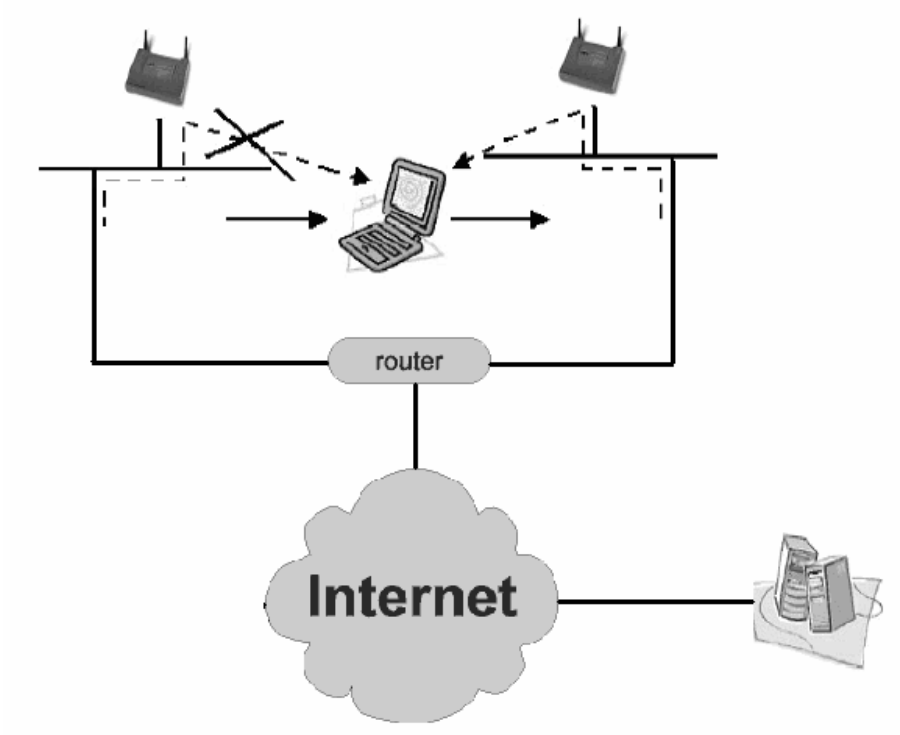

Figura 5.2 Macro-handoff.

- Global handoff (handoff inter-dominio), il cliente si muove tra due AP disposti in due domini Internet diversi. Questo richiede non solo un cambiamento di indirizzo del cliente ma anche del tempo addizionale

per eseguire le operazioni di autenticazione, autorizzazione e accounting (AAA) nel nuovo dominio.

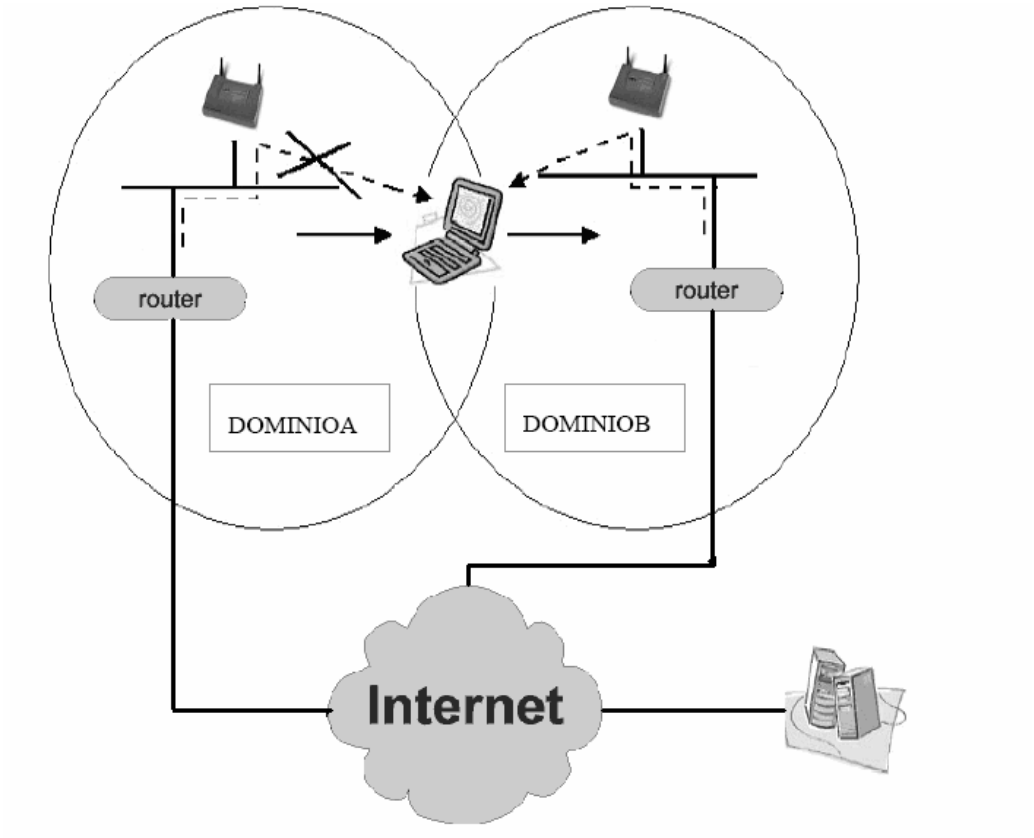

Figura 5.3 Global handoff.

5.2 Componenti delle soluzioni.

Le soluzioni da noi sviluppate per il micro, macro e global handoff utilizzano il concetto di predizione degli spostamenti del terminale. La predizione è di fondamentale importanza, in quanto è possibile così conoscere in anticipo gli spostamenti di un terminale all'interno o all'esterno di una sottorete e/o di un dominio. Prevedendo le migrazioni si riesce ad anticipare le operazioni necessarie per velocizzare l'handoff ed avere continuità di sessione/comunicazione senza interruzioni. Nelle soluzioni sviluppate sono presenti le seguenti entità:

- Client.
- Redirect Server.
- Nodo Proxy.
- Registrar Server.
- Correspondent Host.

Di seguito saranno descritte le funzioni che esse svolgono, nel prossimo capitolo invece si parlerà di come sono state implementate, in particolar modo per il micro handoff.

5.2.1 Client.

L'entità client è il terminale di un utente umano. Deve essere in grado di accedere ai diversi servizi di cui l'utente finale ha bisogno: poter usufruire dei servizi internet, ad esempio richiedere e ricevere audio, visualizzare video (livello applicativo e di presentazione dello stack OSI), …. Per poter eseguire le operazioni precedenti è necessario che il terminale instauri una comunicazione con il server (o con il proxy) che fornirà i servizi richiesti, è necessario, dunque, un livello che si occupi di instaurare una sessione con il correspondent host. Il nostro lavoro di tesi si colloca proprio nel livello sessione dello stack OSI, infatti, attraverso il

protocollo SIP riusciamo ad inizializzare una sessione con il correspondent host e a seguito dell'instaurazione della sessione/comunicazione si può così avere il servizio desiderato. Con la terminal mobility il terminale del client potrebbe muoversi dalla sottorete di appartenenza e quindi dall'AP di origine ad un'altra sottorete e AP di destinazione, ma è necessario che non ci sia un'interruzione sul lavoro che il terminale client sta svolgendo (visualizzazione del video, ricezione di audio, …) bisogna che il processo di handoff sia il più veloce possibile, per la gestione di tale problema abbiamo utilizzato il protocollo SIP con le sue estensioni e un buffer sul lato client, al fine di bufferizzare i dati in arrivo dal server/proxy, in modo tale da fornirli all'utente nel momento in cui si sta effettuando il cambio di AP, cosicché non ci siano interruzioni visibili e l'utente finale non si accorga del cambio di sottorete e/o dominio. Oltre all'introduzione del livello di bufferizzazione e dell'utilizzo del protocollo SIP, è stato introdotto un handoff predictor che si occupa di predire i futuri spostamenti del terminale attraverso la tecnica RSSI-GM (Received Signal Strength Indication-Grey Model). Tale predizione verrà poi inviata al proxy (la gestione da parte del proxy di tale informazione sarà spiegata in seguito).

Naturalmente, dopo aver instaurato la sessione, sarà necessario il trasporto dei dati di interesse, livello 3 dello stack OSI, in questo ambito ci sono due possibili protocolli di utilizzo: TCP (Transmission Control Protocol) e UDP (User Datagram Protocol). In questo lavoro di tesi non ci siamo occupati di tale livello.

5.2.2 Redirect Server.

Il Redirect Server è un componente SIP presente in ogni subnet, si occupa di ricevere le richieste dai client che accedono alla sottorete e di inviar loro risposte. Il redirect assume un ruolo fondamentale in quanto riesce ad ottenere la creazione di un proxy, che il client dovrà contattare per richiedere i servizi di

interesse, l'indirizzo di tale proxy viene poi inviato al client che si occuperà di contattarlo.

5.2.3 Nodo Proxy.

Il nodo proxy è un'entità posta tra l'end-point client e l'end-point server. Come detto, il client ha un buffer in cui bufferizza i dati che saranno forniti all'utente finale durante l'intervallo di handoff, è necessario che i pacchetti inviati dal server in tale intervallo vengano ricevuti lo stesso dal client, allora è necessario che i dati transienti siano bufferizzati sul nodo proxy e poi inviati al client nel momento in cui si connetterà all'AP di destinazione. Mentre la dimensione del buffer client è data dal tipo di terminale che l'utente utilizza, la dimensione del buffer del proxy può variare secondo il buffer e la card Wi-Fi del client che esso serve ed in base alla probabilità dello spostamento del client stesso (infatti la predizione che avviene sul nodo client viene inviata al proxy che la utilizzerà per dimensionare il proprio buffer). Il proxy si occupa, quindi, di ricevere i messaggi dal server e inviarli al client e viceversa.

5.2.4 Registrar Server.

Il Registrar Server è un componente SIP presente in ogni dominio, tiene traccia di tutti i terminali presenti in ogni subnet facente parte del suo dominio di appartenenza. Le informazioni relative ad ogni terminale sono memorizzate in un Location Service, che può essere interrogato direttamente da un proxy per individuare i server.

5.2.5 Correspondent Host.

Il correspondent host è l'end-point server che si occupa di ricevere richieste da un proxy (nel nostro caso) o direttamente da un client e di inviare risposte. I dati inviati possono essere audio, video, testo, ...

5.3 Segnalazione tra le entità durante il processo di micro-handoff.

Nei prossimi paragrafi sarà descritto la soluzione sviluppata per evitare la perdita dei dati e velocizzare l'handoff quando un cliente, durante la sessione/comunicazione con un utente, si muove con il suo terminale da un punto di accesso ad un altro sempre nella stessa sottorete, viene così, nel seguito, evidenziato il protocollo di segnalazione che si ha durante il processo di handoff tra le entità coinvolte nella comunicazione.

5.3.1 Fase iniziale del protocollo: Client-Redirect Server.

Il client nel momento in cui si connette ad un AP di una sottorete, chiede e ottiene dal server DHCP un indirizzo IP da poter utilizzare in tale sottorete di accesso, viene, inoltre, configurato con l'indirizzo del Redirect Server presente in quella sottorete. Il client invia così un messaggio di INVITE al Redirect per invitarlo a partecipare ad una sessione, il Redirect si occupa di far creare un nodo proxy per tale client e risponde inviando una risposta con l'indirizzo del nodo appena creato (figura 5.4).

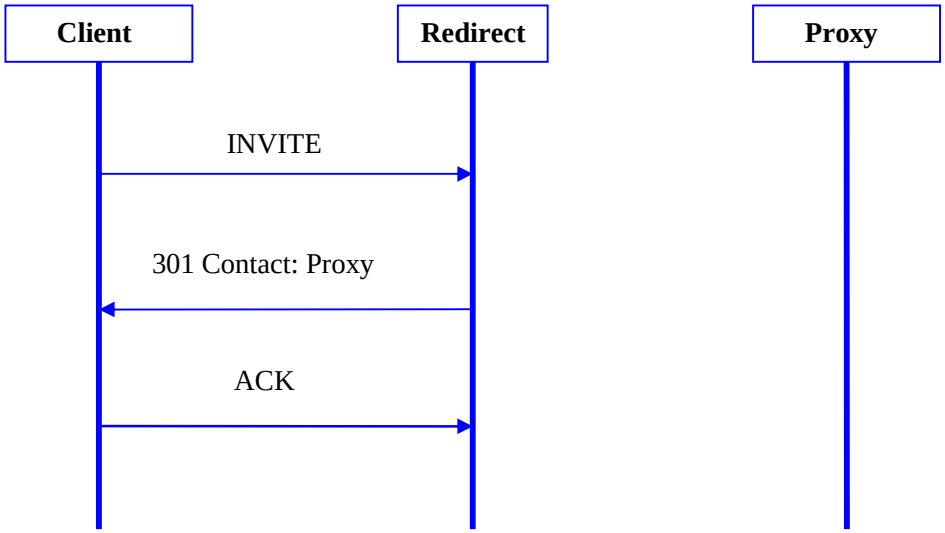

Figura 5.4 Esempio flusso client-redirect server.

5.3.2 Fase intermedia del protocollo: Client-Proxy.

Il client, avendo ottenuto l'indirizzo, contatta il nodo proxy, il quale prima di rispondere registra il client presso il Registrar del dominio di appartenenza.

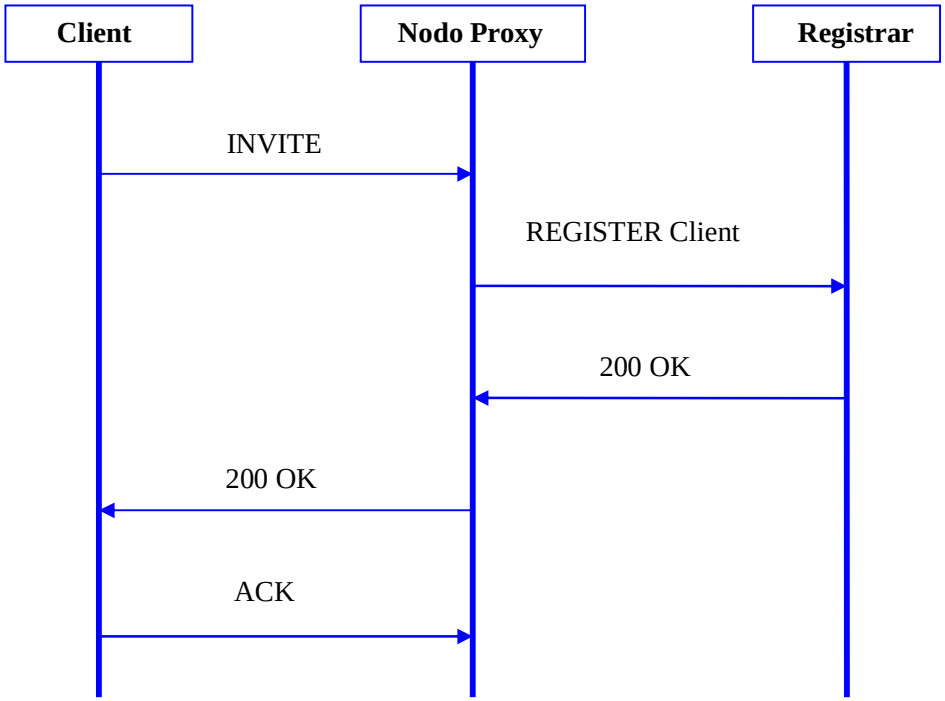

Figura 5.5 Esempio flusso client-proxy.

Il proxy si sottoscrive al client per ricevere notifiche riguardanti il profilo del client e la predizione per così poter dimensionare il proprio buffer, inoltre per ricevere il timestamp dell'ultimo pacchetto ricevuto dal client prima della disconnessione dall'AP di origine.

Il client invia una notifica al proxy indicando il suo profilo. Il proxy si occupa così di instaurare una sessione con il Correspondent Host (CH) con cui il client vuole comunicare. Segue dunque lo scambio dei pacchetti RTP tra CH e il proxy, tra il proxy e il client mobile (figura 5.6).

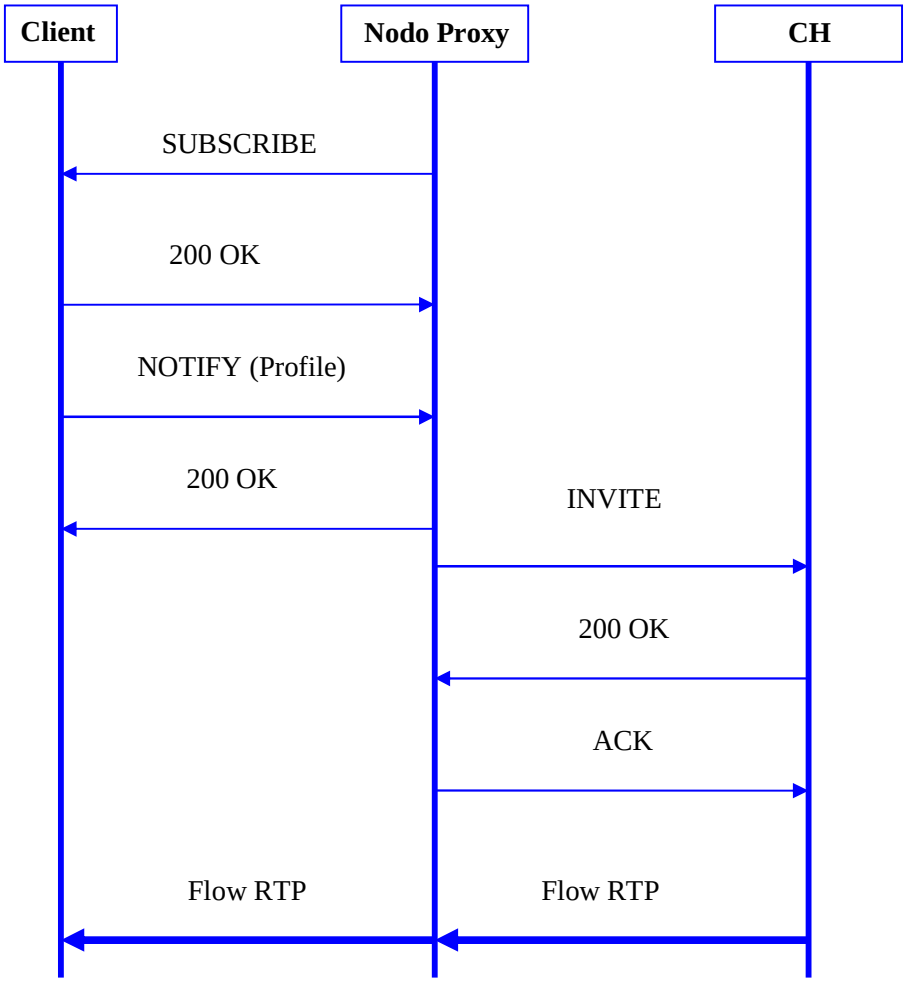

Figura 5.6 Flusso client-proxy-CH.

Nel momento in cui l'handoff predictor predice lo spostamento, l'user agent client invia una notifica al proxy, con l'indicazione della sottorete di destinazione.

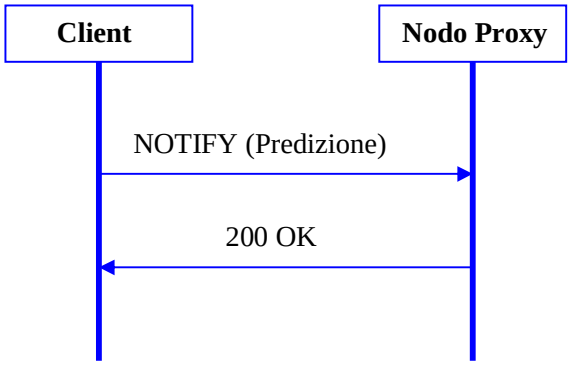

Figura 4.7 Flusso client-proxy.

Il nodo accorgendosi che la migrazione avviene nella stessa sottorete, ossia c'è solo un cambio di AP, aumenta la dimensione del buffer e continua a bufferizzare i dati in arrivo dal server.

5.3.3 Fase finale del protocollo: Client-Proxy.

Quando il client si riconnette al nuovo AP di destinazione invia una notifica al proxy indicando il timestamp dell'ultimo frame che ha ricevuto, quindi il pacchetto RTP che si aspetta di ricevere, riprendendo il normale scambio di flusso tra CH e proxy, proxy e client.

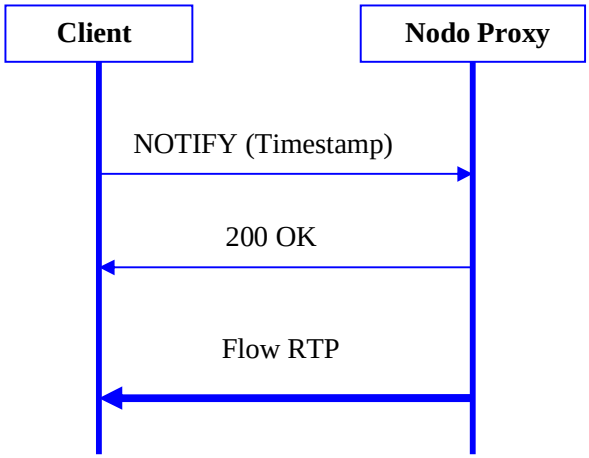

Figura 5.8 Flusso client-proxy.

5.4 Segnalazione tra le entità durante il processo di macro-handoff.

Il processo di macro-handoff si verifica quando un terminale mobile si muove da un AP di una certa sottorete ad un AP di un'altra sottorete sempre comunque nello stesso dominio.

5.4.1 Fase iniziale del protocollo: Client-Redirect Server.

Il client mobile nel momento in cui si connette ad un AP di una sottorete, chiede e ottiene dal server DHCP un indirizzo IP da poter utilizzare in tale sottorete di accesso, viene, inoltre, configurato con l'indirizzo del Redirect Server presente in quella sottorete. Il client invia così un messaggio di INVITE al Redirect per invitarlo a partecipare ad una sessione, il Redirect si occupa di far creare un nodo proxy per tale client e risponde inviando una risposta con l'indirizzo del nodo appena creato.

5.4.2 Fase intermedia del protocollo: Client-Proxy.

Il client contatta il nodo proxy, il quale prima di rispondere registra il client presso il Registrar di quel dominio. Il nodo si sottoscrive al client mobile per ricevere notifiche riguardanti il profilo del cliente, la futura predizione e il timestamp dell'ultimo pacchetto ricevuto dal client prima della disconnessione dall'AP di origine.

Il client invia una notifica al proxy indicando il suo profilo, affinché possa dimensionare opportunamente il buffer. Il proxy si occupa così di instaurare una sessione con il Correspondent Host (CH) con cui l'utente vuole comunicare. Segue dunque lo scambio dei pacchetti RTP tra CH e il proxy, tra il proxy e il client mobile. Nel momento in cui l'handoff predictor predice lo spostamento, il client invia una notifica al proxy con l'indicazione della subnet di destinazione. Il proxy accorgendosi che la migrazione avviene in una subnet diversa si occupa di inviare il buffer al nodo proxy di destinazione.

5.4.3 Fase finale del protocollo: Client-Proxy.

Il terminale mobile quando si riconnette alla nuova sottorete invia una notifica al Redirect Server di tale subnet, quest'ultimo invia al client una risposta con l'indirizzo del nodo proxy da contattare. Il cliente invierà allora la NOTIFY, con l'indicazione del timestamp dell'ultimo pacchetto ricevuto prima della disconnessione dalla sottorete di origine, al proxy. Il proxy allora modificherà la registrazione di tale terminale aggiornandolo con il nuovo indirizzo.

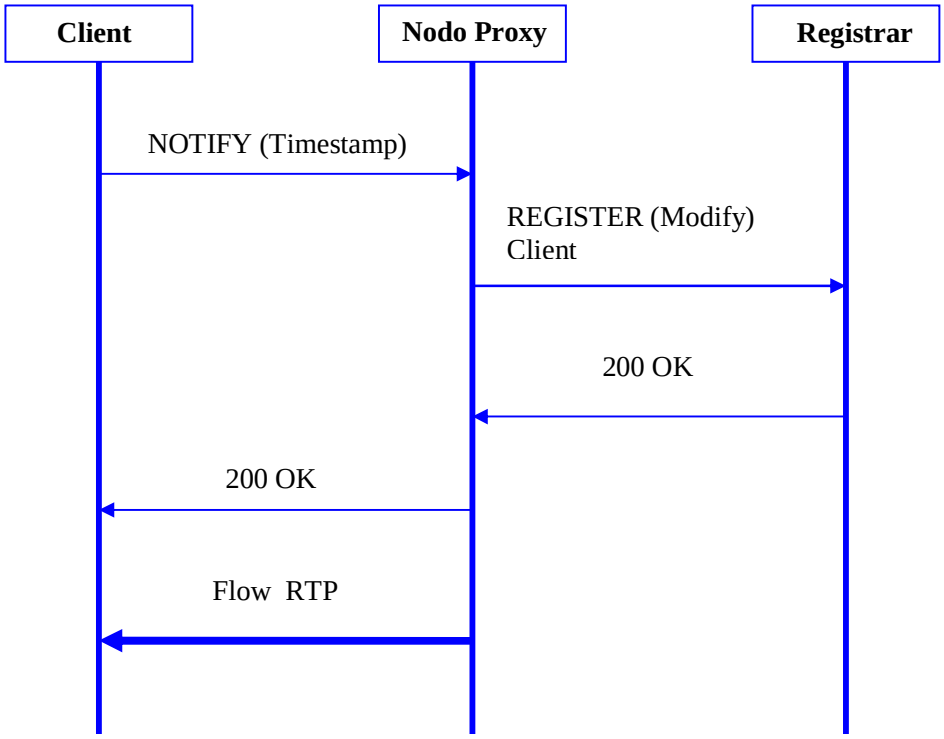

Figura 5.9 Flusso client-proxy-registrar.

Riprende l'invio di pacchetti RTP dal proxy al client.

Il proxy della sottorete di origine nel momento in cui non riceve più il flusso decide di inviare al proxy della sottorete di destinazione il flusso che non ha ancora ricevuto. Dopo aver inviato il flusso il nodo proxy di origine termina.

5.5 Segnalazione tra le entità durante il processo di global-handoff.

Il global-handoff è il processo di handoff che si ha quando un terminale si sposta da una sottorete di un dominio ad una sottorete di un altro dominio.

5.5.1 Fase iniziale del protocollo: Client-Redirect Server.

Il client mobile nel momento in cui si connette ad un AP di una sottorete, chiede e ottiene dal server DHCP un indirizzo IP da poter utilizzare in tale sottorete di accesso, viene, inoltre, configurato con l'indirizzo del Redirect Server presente in quella sottorete. Il client invia così un messaggio di INVITE al Redirect per invitarlo a partecipare ad una sessione, il Redirect si occupa di far creare un nodo proxy per tale client e risponde inviando una risposta con l'indirizzo del nodo appena creato.

5.5.2 Fase intermedia del protocollo: Client-Proxy.

Il client contatta il nodo proxy, il quale prima di rispondere registra il client presso il Registrar di quel dominio. A questo punto il proxy si sottoscrive al client per ricevere notifiche riguardanti il profilo del client, la futura predizione e il timestamp dell'ultimo pacchetto ricevuto dal client prima della disconnessione dall'AP di origine. Il client invia una notifica al proxy indicando il suo profilo, affinché possa dimensionare opportunamente il buffer. Il proxy si occupa così di instaurare una sessione con il Correspondent Host (CH) con cui l'utente vuole comunicare. Segue dunque lo scambio dei pacchetti RTP tra CH e il proxy, tra il proxy e il client mobile. Nel momento in cui l'handoff predictor predice lo spostamento, l'user agent client invia una notifica al proxy con l'indicazione della sottorete di destinazione. Il proxy della sottorete di origine si occupa, allora, di inviare al proxy della subnet di destinazione il proprio buffer e di eliminare la registrazione del client dal Location Service di tale dominio, inviando una REGISTER di distruzione al Registrar, dopo questa operazione tale nodo termina.

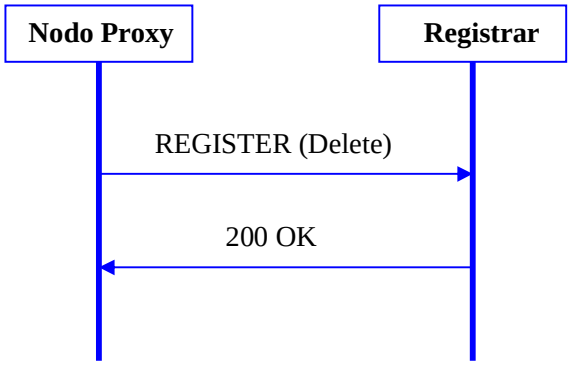

Figura 5.10 Flusso proxy-Registrar.

5.5.3 Fase finale del protocollo: Client-Proxy.

Il nodo proxy del dominio di destinazione mentre aspetta che il cliente si connetta invia a CH un INVITE contenente il frame da cui ritrasmettere. Il proxy inizia a bufferizzare i pacchetti provenienti da CH. Il mobile nel momento in cui si riconnette invia al Redirect di tale sottorete una NOTIFY con l'indicazione del timestamp dell'ultimo pacchetto ricevuto, il Redirect invia al nodo client l'indirizzo del proxy cui fare riferimento. Il client lo contatta, il nodo proxy registra tale terminale presso il Registrar, inizia l'invio dei pacchetti RTP.

CAPITOLO 6

Implementazione della soluzione per il micro-handoff.

In questo capitolo sarà descritta l'implementazione della soluzione sviluppata relativamente alla parte del micro handoff. Saranno indicate le scelte implementative e le tecniche adottate per realizzare le diverse entità coinvolte nella comunicazione: client, redirect server, creator, nodo proxy (B2BUA).

6.1 Tecnologie utilizzate.

Nei paragrafi seguenti saranno introdotte le tecniche implementative utilizzate nella realizzazione della soluzione, in particolar modo si analizzeranno le API JAIN SIP con i relativi package utilizzati e il parser XML utilizzato per processare il body dei messaggi NOTIFY inviati dal Client al B2BUA.

6.1.1 JAIN SIP.

La soluzione è stata implementata utilizzando Java e le API JAIN SIP. L'API JAIN SIP [19] fornisce un'interfaccia Java standard e portabile per condividere informazioni tra client SIP e server SIP, mettendo a disposizione funzionalità per il controllo delle chiamate e quindi fornendo un'infrastruttura per lo sviluppo di applicazioni. JAIN SIP può essere utilizzato per implementare i proxy server, i registrar e gli user agent. Queste API gestiscono tutti gli aspetti relativi alla comunicazione attraverso SIP, ad esempio la risposta alle richieste che arrivano al

programma, attraverso opportune interfacce che gestiscono gli eventi relativi ai vari eventi SIP (come l'arrivo di richieste o di risposte, o la scadenza di timeout), o la creazione e gestione degli indirizzi, messaggi ed elementi del pacchetto SIP. JAIN SIP supporta le funzionalità del protocollo SIP definite in RFC 3261 e molte altre estensioni presenti in diversi RFC:

- 3428: permette il trasferimento di Instant Message.
- 3265: permette di richiedere notifiche per certi eventi.
- 3326: permette di capire perchè una richiesta SIP è stata inviata.
- 3311: permette di modificare i parametri di una sessione.

6.1.1.1 Architettura JAIN SIP per la comunicazione fra due user agent.

La figura 6.1 mostra l'architettura JAIN SIP necessaria affinché due User Agent SIP possano comunicare con messaggi SIP. Sono presenti diversi elementi fondamentali affinché ci sia la comunicazione fra le due entità:

- SIPStack: è associato ad un indirizzo IP, ha diverse proprietà che possono essere configurate, gestisce i Listening Point (ci possono essere più listening point su uno stesso Stack) e i Provider. Una volta creato non può essere distrutto. Un'applicazione può avere più SIPStack.
- Listening Point: sono punti di ascolto, definiscono il protocollo e la porta su cui il Provider è in ascolto.
- SIPProvider: è in grado di registrare un SIPListener, in maniera tale da inviargli eventi (messaggi di Request e Response sottoforma di eventi) quando sopraggiungono dalla rete. Rende possibile l'invio di Request e Response in maniera stateless. Fornisce dei metodi per creare Client Transaction e Server Transaction così da inviare Request e Response stateful. C'è un unico Provider per ogni Listening Point.

- SIPListener: espone un’interfaccia per ricevere richieste e risposte, per ogni Stack può esserci un solo Listener.
- Application: a livello architetturale è l’application che implementa l’interfaccia del SIPListener, si occupa di registrarsi presso un SIPProvider al fine di interagire con lo Stack. Riceve i messaggi come eventi attraverso l’interfaccia SIPListener.

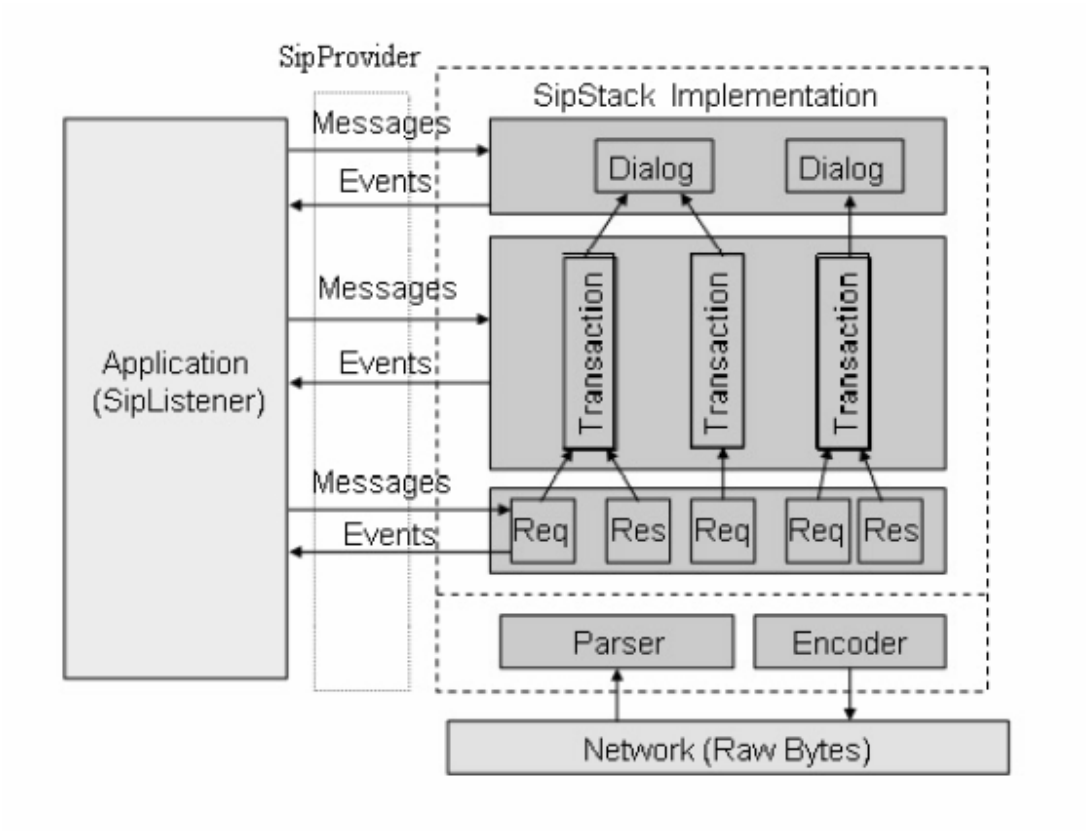

Figura 6.1 Schema di un’applicazione JAIN SIP.

6.1.1.2 Package utilizzati.

Sono quattro i package contenuti nell'API JAIN SIP, sono stati tutti utilizzati nella nostra implementazione:

- General package: contiene le interfacce per le transazioni (Client e Server), per il dialogo, per gli oggetti eventi, per lo Stack, per il Provider, per il Listener. Inoltre definisce un SipFactory, un'interfaccia che fornisce i metodi per creare un nuovo oggetto Stack e altri oggetti Factory.
- Address package: contiene l'interfaccia per il SipURI e il TelURI, l'interfaccia AddressFactory che definisce i metodi per creare nuovi SipURI e TelURI
- Header package: contiene le interfacce di tutti gli header supportati e di tutte le estensioni di tali header e l'HeaderFactory, un'interfaccia che definisce i metodi per creare nuovi oggetti header.
- Message package: definisce le interfacce necessarie per i messaggi di Request e Response, l'interfaccia MessageFactory che definisce i metodi per creare nuovi oggetti Request e Response.

Nell'ambito di questa tesi è stata utilizzata l'implementazione di riferimento definita da NIST (National Institute of Standards and Technology) [20].

6.1.2 Parser XML.

Per poter estrarre i dati presenti nel body dei messaggi NOTIFY, scritti in xml, è stato utilizzato il parser xml SAX [22], è stata così creata la classe *public class XMLParser extends DefaultHandler{…}* in cui sono stati implementati i metodi dell'interfaccia DefaultHandler:

- *public void startDocument() throws SAXException{…};*
- *public void endDocument() throws SAXException{…};*

- *public void setDocumentLocator() throws SAXException{…};*
- *public void startElement() throws SAXException{…};*
- *public void endElement() throws SAXException{…};*
- *public void characters() throws SAXException{…};*
- *public void ignorableWhiteSpace() throws SAXException{…};*
- *public void processingInstructions() throws SAXException{…};*

in tale classe è stato definito anche un metodo per creare il corpo del messaggio di ciascuna NOTIFY: *public String createXMLBody(){…}.*

6.2 Applicazione Client.

Il client è stato realizzato come SIP User Agent, durante un dialogo può alternare il ruolo di client a quello di server, quindi in ogni transazione può avere un ruolo diverso. Ricordiamo che una transazione consiste in una richiesta, seguita o meno da una o più risposte provvisorie (provisional 1xx) che indicano lo stato in cui si trova la chiamata e da una risposta finale che indica il successo o il fallimento della richiesta.

La classe Client implementa l'interfaccia SipListener delle API JAIN SIP, definendo diversi metodi:

- ***public void start() throws Exception {…}***, si occupa di: creare il SipStack con determinate proprietà (*properties*) per lo user agent in questione, *sipStack = sipFactory.createSipStack(properties);*, inizializzare il Listening Point con una determinata porta e protocollo di trasporto *ListeningPoint lp = sipStack.createListeningPoint(getPort(), "UDP");*, creare il SipProvider *sipProvider = sipStack.createSipProvider(lp);* e registrare il client presso tale entità *sipProvider.addSipListener(this);* affinché il client riceva i messaggi

sottoforma di eventi attraverso il provider. Nella figura sottostante viene visualizzata l'architettura del Client che si ha dopo l'invocazione del metodo start();.

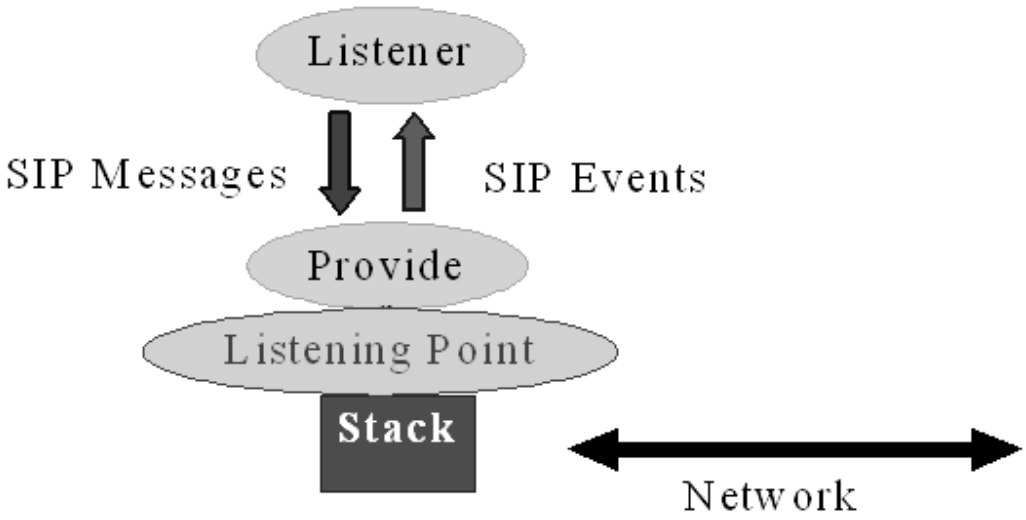

Figura 6.2 Architettura dell'applicazione client.

- ***public void stop() throws Exception {…},*** permette di fermare l'esecuzione del client, è stato implementato in quanto si ha così la possibilità di controllare il funzionamento dello user agent, ad esempio un'applicazione grafica potrebbe avviare e fermare l'esecuzione a seguito di alcuni eventi che si sono manifestati.
- ***public void processRequest(RequestEvent requestEvent) {…} ,*** utilizzando questo metodo il client riesce a ricevere i messaggi di richiesta provenienti da proxy o server, inviati sottoforma di eventi dal SipProvider sottostante. All'interno di tale metodo vengono svolte diverse operazioni per ogni possibile richiesta ricevuta (come si vedrà

nella sezione 6.6), nella nostra implementazione, ad esempio, nel caso in cui dovesse arrivare un messaggio di SUBSCRIBE il client si occuperebbe di processare il messaggio ricevuto con il seguente metodo da noi implementato *processSubscribe(requestCloned, serverTransaction);*: ha il compito di creare e inviare al proxy un messaggio 200 OK e un messaggio di NOTIFY.

- ***public void processResponse(ResponseEvent responseEvent) {…},*** il client riesce così a ricevere e a gestire le risposte provenienti da proxy o server. Come le richieste anche le risposte sono ricevute prima dal SipProvider, poi quest'ultimo li invia al SipListener sottoforma di eventi (responseEvent). Se il client dovesse ricevere una risposta allora all'interno di tale metodo avremmo un ulteriore metodo che si occuperebbe di eseguire delle operazioni in base alla richiesta che ha scatenato questa risposta, nel nostro caso se la risposta 200 OK dovesse essere una conseguenza di un INVITE si avrebbe *sendAck(localSipURI,remoteSipURI,callid,cseq,localTag,remoteTag);*, si occupa di creare e inviare al proxy o al server un messaggio contenente la richiesta ACK, dopo essere stata inviata al SipProvider .

L'applicazione client viene, dunque, messa in esecuzione con il metodo start(), che pone il client in attesa di eventi provenienti dal SipProvider e che gestisce attraverso i metodi *processRequest(…)* e *processResponse(…)*.

6.3 Applicazione Redirect Server.

Il Redirect Server è stato implementato come user agent e non come proxy in quanto, può così, inviare messaggi di richiesta e non soltanto messaggi di risposta, come invece si comporterebbe se fosse implementato come proxy SIP. Anche

l’applicazione Redirect Server implementa l’interfaccia SipListener, quindi anche i metodi descritti nel precedente paragrafo, naturalmente svolgerà, all’interno di tali metodi, operazioni diverse da quelle svolte dal client.

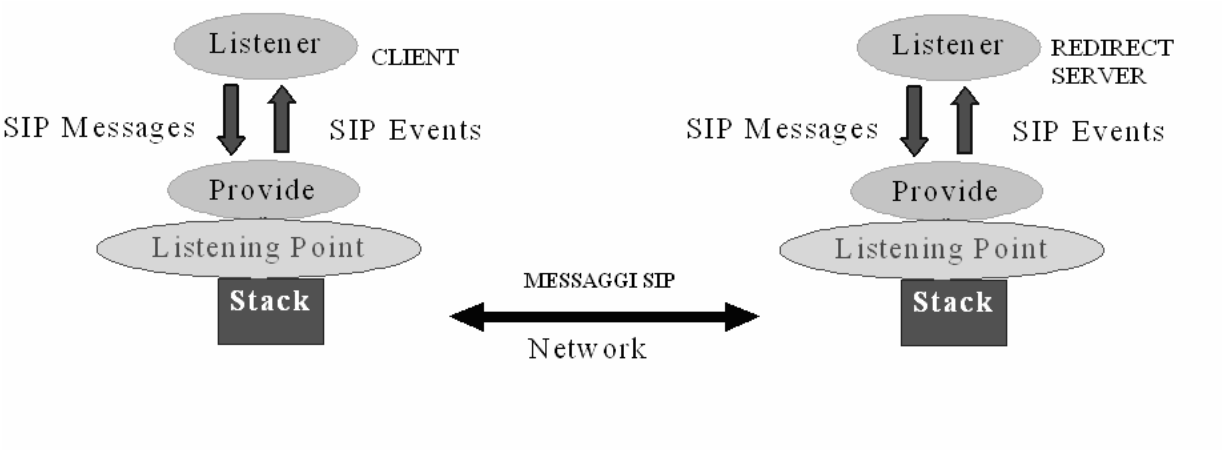

Figura 6.3 Client che comunica con l’architettura dell’applicazione Redirect Server attraverso l’invio e la ricezione di messaggi SIP.

In ogni sottorete c’è un Redirect Server che si occupa di redirigere i messaggi dei Client verso altre destinazioni, in particolare, nel momento in cui arriva una richiesta INVITE si occupa di chiedere la creazione di un proxy ad un’entità chiamata Creator.

La richiesta viene effettuata attraverso una Socket Stream, viene dunque creata la connessione con il Creator (remoteAddr, remotePort), sono creati gli stream di input e di output di tale socket al fine di inviare richieste e ricevere risposte dal pari. Di seguito è stata inserita la sequenza di codice che si occupa di ciò:

Socket socket = new Socket(remoteAddr, remotePort);

inSock = new DataInputStream(socket.getInputStream());

outSock = new DataOutputStream(socket.getOutputStream());

Il Creator si occupa di creare un proxy che verrà utilizzato dal Client e restituisce l'indirizzo e la porta del nodo creato al Redirect.

Il Redirect legge attraverso lo stream di input i dati ricevuti:

String indirizzo = inSock.readUTF();

int porta = inSock.readInt();

a questo punto invierà la risposta al Client.

6.4 Applicazione Creator.

Riassumiamo brevemente le responsabilità del Creator:

- Creazione di una Socket Stream, capace di ricevere le richieste provenienti dal Server Redirect.
- Creazione del proxy necessario al client per poter comunicare con il correspondent host riducendo le possibili interruzioni durante il dialogo.
- Invio, al Server Redirect attraverso la Socket, dell'indirizzo e della porta del proxy creato.

Analizziamo ora l'implementazione dei singoli punti senza tuttavia seguire l'ordine con il quale sono stati introdotti.

L'applicazione Creator nel momento in cui viene messa in esecuzione crea un SipStack, un SipProvider, un Listener (attraverso il metodo start() della classe UserAgent) ed un Thread B2BUA così da avere un B2BUA attivo nel momento in cui arriva una richiesta dal Server Redirect (Figura 6.4).

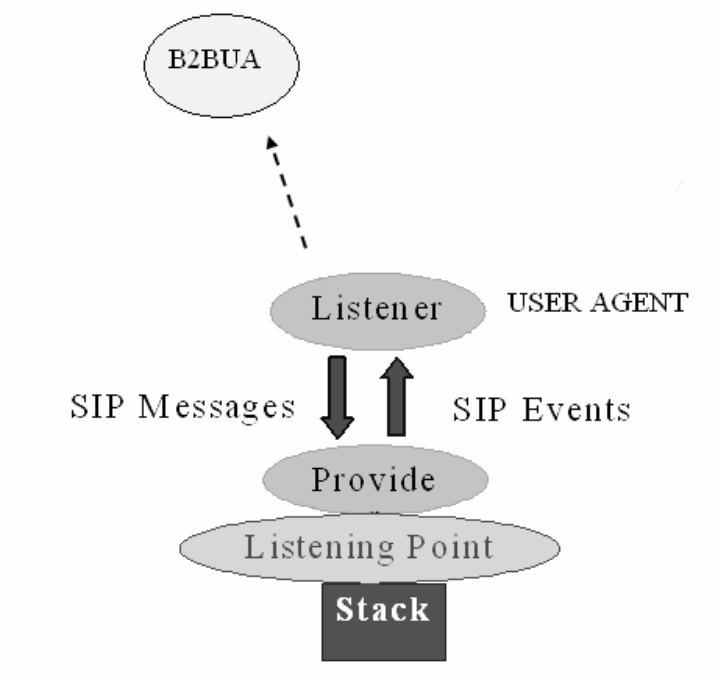

Figura 6.4 Architettura creata dal Creator attraverso il metodo start() della classe UserAgent e il metodo run() del Thread B2BUA.

Dopo queste prime operazioni crea una Socket Stream *ServerSocket serverSocket = new ServerSocket(port);*, si mette in attesa di richieste *clientSocket = serverSocket.accept();*, alla ricezione di una richiesta si occupa prima di controllare se ha già un Thread disponibile attivo, altrimenti si occupa di creare un nuovo SipProvider su tale SipStack, associa al nuovo provider il listener iniziale, e attiva per tale provider un B2BUA che si occuperà di gestire i messaggi spediti dal client.

Dopo la creazione invia l'indirizzo *outSock.writeUTF("alice@127.0.0.1");* e la porta *outSock.writeInt(nPorta);* del nodo creato (IP e porta del SIPProvider legato al B2BUA attivato) al Redirect.

Nel precedente capitolo abbiamo parlato della creazione di un proxy capace di servire un client, nell'implementazione parliamo di B2BUA, in quanto deve

essere capace di inviare richieste e risposte SIP, prendere decisioni sia nei confronti del client sia nei confronti del correspondent host.

6.5 Classe User Agent.

La classe user agent implementa l'interfaccia SipListener e diversi metodi che vengono descritti di seguito:

- ***public void start() throws Exception {…}***, si occupa di creare il SipStack, il SipProvider e il SipListener.
- ***public void processRequest(RequestEvent requestEvent) {…} ,*** il SipListener si mette in ascolto dei possibili messaggi che possono essere ricevuti dal SipStack e dai diversi SipProvider che ad esso sono associati. Poiché le specifiche JAIN SIP affermano che ogni Stack può avere un unico Listener e più Provider, abbiamo deciso di creare un unico Stack con più Provider ed un unico Listener che ascolta tutti i SipProvider e fornisce al livello superiore gli eventi sopraggiunti, svolge dunque una funzione di "routing", dei messaggi che riceve, verso il corrispondente Thread B2BUA, infatti sulla base del Provider che ha inviato tale evento, poiché ogni Provider è legato ad un B2BUA, riesce ad individuare il destinatario del messaggio. Invia così la notifica e l'evento richiesta al thread coinvolto.
- ***public void processResponse(ResponsetEvent responseEvent) {…} ,*** invia una notifica e l'evento risposta che ha ricevuto al thread a cui è destinata tale risposta.
- ***public void newProvider(int porta, B2BUA b2bua) {…} ,*** tale metodo riesce a creare un nuovo provider da associare ad un nuovo thread, così da essere messo in ascolto dei messaggi provenienti dal nuovo client.

Tale entità viene messa in esecuzione dal Creator e persiste finchè ci sono Provider attivi sullo Stack, si occupa di ricevere tutti i messaggi provenienti dai vari Client e di inviare i messaggi ai Client.

6.6 Applicazione B2BUA.

L'applicazione B2BUA è stata implementata come Thread, si ha infatti *public class B2BUA extends Thread {…}*. Ogni Thread è legato ad un SipProvider presente sullo stack SIP e quindi ha una porta e un protocollo stabilito (TCP o UDP). Ogni Thread ha due Vector in cui inserisce gli eventi inviati dal SipListener, un Vector per gli eventi richiesta e un Vector per gli eventi risposta. Nel momento in cui viene messo in esecuzione controlla se ci sono eventi nelle due code, se non ci sono eventi entra nello stato di *wait();*, da questo stato esce nel momento in cui arriva una *notify()* dal listener, che si occupa di risvegliarlo e di inviargli l'evento richiesta o risposta. In base all'evento ricevuto eseguirà diverse operazioni, rigorosamente *synchronized*, affinché non ci siano problemi di condivisione delle risorse, e invierà messaggi di risposta SIP o richiesta SIP al client che lo ha interrogato attraverso il proprio SipProvider.

Nella nostra implementazione il B2BUA comunica solo con il client, ma è stato predisposto anche per comunicare con un registrar di dominio e con un correspondent host o ulteriore proxy.

Nella nostra soluzione avremo dunque un unico SipStack, tanti SipProvider quanti sono i B2BUA attivi e un unico Listener per tutti i possibili messaggi che vengono inviati da ogni Client ad un determinato B2BUA, Listener che si occuperà di redirigere gli eventi ricevuti al Thread di destinazione. Quindi avremo un'architettura simile a quella di figura 6.5.

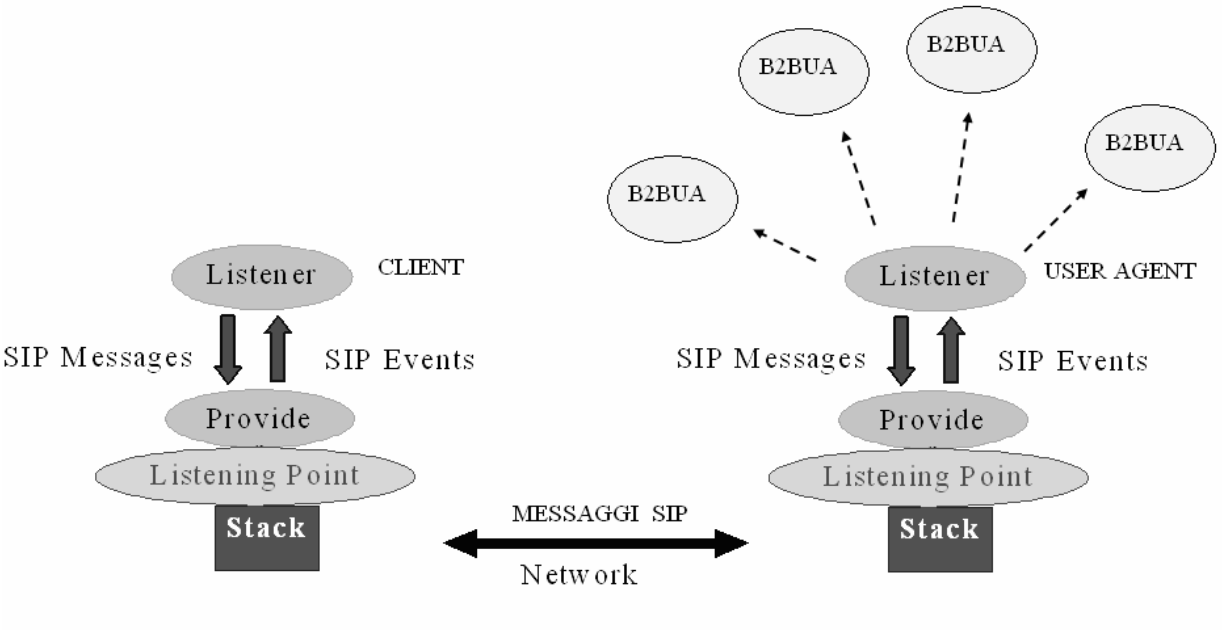

Figura 6.5 Architettura realizzata.

6.7 Implementazione messaggi.

Nel capitolo precedente è stato descritto il protocollo di segnalazione tra le varie entità, di seguito saranno descritti i metodi che ogni entità utilizza per creare i messaggi e la struttura dei messaggi.

6.7.1 Messaggi lato Client.

Nel momento in cui un terminale, ad esempio elisabetta, accede in una sottorete lo user agent client si occupa di inviare il messaggio SIP INVITE al Server Redirect di tale sottorete, ad esempio betty. Nel messaggio INVITE il client introduce un body SDP in cui inserisce le informazioni riguardanti le caratteristiche della sessione. Di seguito è riportato un messaggio INVITE che potrebbe essere inviato dal client al server redirect.

```
INVITE sip:betty@127.0.0.1:4102;transport=udp SIP/2.0
Call-ID: nist-sip-invite-callId 1
CSeq: 1 INVITE
From: <sip:elisabetta@127.0.0.1:6102>;tag=8826
To: <sip:betty@127.0.0.1:4102>
Via: SIP/2.0/UDP 127.0.0.1:6102;branch=z9hG4bK3e8c90855d371fdf602e6bb1cb218eb5
Max-Forwards: 70
Content-Type: application/sdp
Contact: <sip:127.0.0.1:6102;transport=udp>
Content-Length: 44

m=audio 3456 RTP/AVP 0
c=IN IP4 127.0.0.1
```

Per poter creare e inviare tale messaggio è stato implementato il metodo: *sendInvite(localSipURI,remoteSipURI,mediaDescription,cseq);*, i primi due argomenti sono gli indirizzi SIP, rispettivamente l'indirizzo locale e l'indirizzo del Redirect Server, il terzo argomento il corpo del messaggio e il quarto indica il numero progressivo delle richieste inviate tra client e server. L'invio è stato eseguito attraverso una client transaction. Per quanto riguarda il body del messaggio è stato utilizzato il package *javax.sdp.*;* che contiene il metodo *createMediaDescription("audio", 3456,0,"RTP/AVP",list);* per poter creare il testo da inserire nei messaggi INVITE.

Nel momento in cui riceve, dal server redirect, la risposta **301 MOVED_PERMANENTLY** invia un messaggio **ACK** con il metodo *sendAck(localSipURI,remoteSipURI,callid,cseq,localTag,remoteTag);*

```
ACK sip:betty@127.0.0.1:4102;transport=udp SIP/2.0
Call-ID: nist-sip-invite-callId 1
CSeq: 1 ACK
From: <sip:elisabetta@127.0.0.1:6102>;tag=8826
To: <sip:betty@127.0.0.1:4102>;tag=3437
Via: SIP/2.0/UDP 127.0.0.1:6102;branch=z9hG4bK39e96f59bce66ec8178f3f7b23c282ab
Max-Forwards: 10
Contact: <sip:127.0.0.1:6102;transport=udp>
Content-Length: 0
```

Preleva dal ContactHeader l'indirizzo della nuova entità da contattare, ossia l'indirizzo del B2BUA (ad esempio alice), che è stato creato per tale client, e invia lo stesso messaggio di INVITE precedente, modificando il RequestURI, il FromHeader, il SequenceNumber del CSeqHeader, a tale indirizzo.

```
INVITE sip:alice@127.0.0.1:3111;transport=udp SIP/2.0
Call-ID: nist-sip-invite-callId 1
CSeq: 2 INVITE
From: <sip:elisabetta@127.0.0.1:6102>;tag=2368
To: <sip:alice@127.0.0.1:3111>
Via: SIP/2.0/UDP 127.0.0.1:6102;branch=z9hG4bK0a767f349511d71a943ab4ecbcc06c7e
Max-Forwards: 70
Content-Type: application/sdp
Contact: <sip:127.0.0.1:6102;transport=udp>
Content-Length: 44

m=audio 3456 RTP/AVP 0
c=IN IP4 127.0.0.1
```

Alla ricezione della risposta **200 OK** invia una richiesta di ACK e instaura così un dialogo con tale entità.

In questo ambito è stato creato un event package **client**, sulla base dell'event framework del protocollo SIP, che si occupa di gestire, come sarà evidenziato in seguito, le informazioni di profilo del client, la predizione dello spostamento e il timestamp dell'ultimo pacchetto ricevuto dal client.

Dopo aver instaurato la sessione con il B2BUA, quest'ultimo si sottoscrive al client per l'EventHeader client.

Il client, dopo aver effettuato dei controlli sull'esistenza del package specificato nell'EventHeader, crea e invia un messaggio **200 OK** o **202 ACCEPTED** se deve ancora verificare l'autorizzazione per tale B2BUA (abbiamo simulato un controllo per l' autorizzazione del B2BUA per questo è stata inviata la risposta 202).

```
SIP/2.0 202 Accepted
Call-ID: nist-sip-invite-callId 1
CSeq: 3 SUBSCRIBE
From: <sip:alice@127.0.0.1:3111>;tag=3945
To: <sip:elisabetta@127.0.0.1:6102>;tag=2368
Via: SIP/2.0/UDP 127.0.0.1:3111;branch=z9hG4bK89fb77ca11cb66be219fe98b90ae85d0
Max-Forwards: 70
Content-Length: 0
```

Tale messaggio è stato creato con il metodo *createResponse(Response.ACCEPTED,request);* e inviato attraverso una server transaction con *sendResponse(response);* .

Dopo aver inoltrato tale risposta, invia una **NOTIFY** contenente il proprio profilo nel body del messaggio. Il body ora è scritto nel linguaggio xml. Tale richiesta viene creata e inviata con il seguente metodo:

public void sendNotify(String localSipURL, String remoteSipURL, String body,int cseq, CallIdHeader callid, String remoteTag) {… }

```
NOTIFY sip:alice@127.0.0.1:3111;transport=udp SIP/2.0
Call-ID: nist-sip-invite-callId 1
CSeq: 4 NOTIFY
From: <sip:elisabetta@127.0.0.1:6102>;tag=7064
To: <sip:alice@127.0.0.1:3111>;tag=3945
Via: SIP/2.0/UDP 127.0.0.1:6102;branch=z9hG4bK7fe91218127b3ca6a3c25c799a03b6cd
Max-Forwards: 70
Content-Type: application/clientinfo+xml
Subscription-State: active
Event: client
Content-Length: 275

<?xml version="1.0"?>
<!DOCTYPE clientinfo SYSTEM "clientinfo.dtd">
<clientinfo xmlns="urn:params:xml:ns:clientinfo">
 <profile aor="sip:elisabetta@127.0.0.1">
  <card type="Compaq"/>
  <frame rate="14.9"/>
  <levelPrediction>basso</levelPrediction>
 </profile>
</clientinfo>
```

Dopo queste operazioni il client inizia a ricevere il flusso RTP dal B2BUA (questo aspetto non è stato analizzato da tale tesi).

Nel momento in cui l'handoff predictor, presente sul nodo client, predice un cambio del livello di predizione (ad es. da basso ad alto), ossia il client sta per eseguire uno spostamento, l'user agent client crea e invia una NOTIFY con tale predizione al suo pari (*public void sendNotify(String localSipURL, String remoteSipURL, String body,int cseq, CallIdHeader callid, String localTag,String remoteTag)*), cosicché il B2BUA possa aumentare la dimensione del suo buffer e immagazzinare il flusso proveniente dal correspondent host (noi non ci siamo occupati di ciò).

```
NOTIFY sip:alice@127.0.0.1:3111;transport=udp SIP/2.0
Call-ID: nist-sip-invite-callId 1
CSeq: 5 NOTIFY
From: <sip:elisabetta@127.0.0.1:6102>;tag=7064
To: <sip:alice@127.0.0.1:3111>;tag=3945
Via: SIP/2.0/UDP 127.0.0.1:6102;branch=z9hG4bK9ad3c15d5320316727a8298c61727ebf
Max-Forwards: 70
Content-Type: application/clientinfo+xml
Subscription-State: active
Event: client
Content-Length: 247

<?xml version="1.0"?>
<!DOCTYPE clientinfo SYSTEM "clientinfo.dtd">
<clientinfo xmlns="urn:params:xml:ns:clientinfo">
<prediction aor="sip:elisabetta@127.0.0.1">
<level>alto</level>
<subnetDest>null</subnetDest>
</prediction>
</clientinfo>
```

L'utilizzatore del terminale client non si accorge che si sta spostando da un AP ad un altro in quanto continua a ricevere il flusso senza interruzione, questo grazie al buffer presente sul nodo client che riesce a bufferizzare il flusso in maniera tale da fornirlo all'utilizzatore quando il terminale si sta spostando.
Nel momento in cui si riconnette, il client invia una **NOTIFY** al B2BUA con l'indicazione del timestamp dell'ultimo pacchetto ricevuto, iniziando così a ricevere il flusso di pacchetti RTP dal B2BUA.

```
NOTIFY sip:alice@127.0.0.1:3111;transport=udp SIP/2.0
Call-ID: nist-sip-invite-callId 1
CSeq: 6 NOTIFY
From: <sip:elisabetta@127.0.0.1:6102>;tag=7064
To: <sip:alice@127.0.0.1:3111>;tag=3945
Via: SIP/2.0/UDP 127.0.0.1:6102;branch=z9hG4bK1b6b100659393796ac7868164425c32d
Max-Forwards: 70
Content-Type: application/clientinfo+xml
Subscription-State: active
Event:  client
Content-Length: 209

<?xml version="1.0"?>
<!DOCTYPE clientinfo SYSTEM "clientinfo.dtd">
<clientinfo xmlns="urn:params:xml:ns:clientinfo">
<flow aor="sip:elisabetta@127.0.0.1">
<timestamp></timestamp>
</flow>
</clientinfo>
```

Nel momento in cui vuole terminare il dialogo invia al suo pari un messaggio BYE (*createRequest (requestURI, "BYE", callIdHeader, cseqHeader, fromHeader, toHeader, viaList, maxForwardsHeader);*) ed aspetta la risposta **200 OK**, terminando così la sessione.

```
BYE sip:alice@127.0.0.1:3111;transport=udp SIP/2.0
Call-ID: nist-sip-invite-callId 1
CSeq: 7 BYE
From: <sip:elisabetta@127.0.0.1:6102>;tag=4198
To: <sip:alice@127.0.0.1:3111>
Via: SIP/2.0/UDP 127.0.0.1:6102;branch=z9hG4bKfadaf0bb2fef433fee0655278cf5fc66
Max-Forwards: 10
Contact: <sip:127.0.0.1:6102;transport=udp>
Content-Length: 0
```

6.7.2 Messaggi lato Redirect Server.

Nel momento in cui riceve un messaggio INVITE da un client, si occupa, prima di rispondere, di contattare attraverso una socket Stream un Creator, che crea un nodo B2BUA e invia, attraverso la socket, l'indirizzo di tale nodo creato. A questo punto il Redirect crea e invia al client il messaggio **301 MOVED_PERMANENTLY** con l'indirizzo del nuovo B2BUA creato nel

ContactHeader (*create Response(Response.MOVED_PERMANENTLY,request);* , per l'invio utilizza una server transaction).

```
SIP/2.0 301 Moved permanently
Call-ID: nist-sip-invite-callId 1
CSeq: 1 INVITE
From: <sip:elisabetta@127.0.0.1:6102>;tag=8826
To: <sip:betty@127.0.0.1:4102>;tag=3437
Via: SIP/2.0/UDP 127.0.0.1:6102;branch=z9hG4bK3e8c90855d371fdf602e6bb1cb218eb5
Max-Forwards: 70
Contact: <sip:alice@127.0.0.1:3111>
Content-Length: 0
```

Il Redirect aspetta una richiesta **ACK** e non verrà più contattato da tale client.

6.7.3 Messaggi lato B2BUA.

Nel momento in cui riceve un INVITE da un client risponde con un 200 OK, ossia accetta di instaurare una sessione con tale entità. Dopo aver ricevuto l'ACK invia una SUSCRIBE per l'evento **client** attraverso il metodo *sendSubscribe(String localSipURL,String remoteSipURL,CallIdHeader callid, int cseq,String localTag, String remoteTag);*.

```
SUBSCRIBE sip:elisabetta@127.0.0.1:6102;transport=udp SIP/2.0
Call-ID: nist-sip-invite-callId 1
CSeq: 3 SUBSCRIBE
From: <sip:alice@127.0.0.1:3111>;tag=3945
To: <sip:elisabetta@127.0.0.1:6102>;tag=2368
Via: SIP/2.0/UDP 127.0.0.1:3111;branch=z9hG4bK89fb77ca11cb66be219fe98b90ae85d0
Max-Forwards: 70
Contact: <sip:127.0.0.1:3111;transport=udp>
Expires: 9900
Event:  client
Accept: application/clientinfo+xml
Content-Length: 0
```

Alla ricezione della prima NOTIFY proveniente dal client, estrae le informazioni di profilo contenute attraverso il parser xml SAX e dimensiona il proprio buffer

(l'operazione di creazione e modifica buffer non è stata presa in considerazione da questa tesi). Invia la risposta 200 OK.
C'è così l'invio dei pacchetti RTP (anche questo scambio non è stato considerato) provenienti dal correspondent host per il client, che sono prima inviati al nodo B2BUA, il quale si occupa poi di inviarli al client.
Riceverà una seconda NOTIFY nel momento in cui il client sta per effettuare uno spostamento. Invierà una response 200 OK.
La terza NOTIFY sarà inviata dal client nel momento in cui si riconnetterà alla subnet attraverso un altro AP. Nel momento in cui riceverà la terza NOTIFY inizierà ad inviare il flusso che ha immagazzinato (il lavoro di tesi non si è occupato dello svuotamento del buffer).
Nel momento in cui riceverà la richiesta BYE invierà una risposta 200 OK, utilizzando il metodo *processBye(requestCloned,serverTransaction);*.

6.8 Event Package Client.

Sulla base dell'event framework, è stato sviluppato un nuovo package che si occupa della gestione: delle informazioni di profilo del client (tipo di card Wi-Fi installata, frame rate richiesto, dimensione del buffer client), della predizione degli spostamenti del client (predizione avuta attraverso la tecnica RSSI-GM), del timestamp dell'ultimo pacchetto ricevuto dal client prima della disconnessione dall'AP di origine. Il nome di questo package è "**client**". Questo valore appare nell'**Event** header presente nelle richieste SUBSCRIBE e NOTIFY (*Event: client).*
Tutti i sottoscrittori e i notificatori devono supportare, per il body presente nella NOTIFY, il formato "*application/clientinfo+xml"*. La richiesta SUBSCRIBE può contenere un header **Accept,** se non presente allora il suo valore di default sarà "*application/clientinfo+xml"*, altrimenti deve inserire "*application/clientinfo+xml"*

e può includere altri tipi capaci di rappresentare le informazioni di profilo, di predizione e di timestamp del client. Le notifiche generate dal client devono, quindi, essere in uno dei formati specificati nel campo Accept della richiesta SUBSCRIBE. E' possibile anche specificare la durata della sottoscrizione con l'header **Exipres** nella SUBSCRIBE. Il body del messaggio SUBSCRIBE è opzionale, nella nostra implementazione non è stato introdotto, ma è possibile inserirlo estendendo tale package.

6.8.1 Struttura documento client.

Il documento che racchiude le informazioni, riguardanti il client, utilizza XML 1.0 ed è codificato usando UTF-8. E' stato creato un documento DTD (Document Type Definition) in cui viene definita una grammatica di linguaggio markup per tale documento [22].

L'elemento radice è il tag "clientinfo", seguono altri elementi:

- profile;
- prediction;
- flow.

Prendendo in considerazione "profile", ha come attributo "aor", che contiene un URI che è l'address-of-record del client a cui il B2BUA è interessato. Contiene anche diversi elementi:

- card con attributo "type", indica il tipo di card Wi-Fi del client;
- frame con attribute "rate", specifica la velocità di trasmissione dei frame.

Per quanto riguarda il tag "prediction", esso ha un attributo "aor", che contiene l'URI del client di interesse, e due elementi:

- level, indica il livello di probabilità dello spostamento del client, può assumere tre valori: alto, medio e basso;

- subnetDest, fornisce la sottorete di destinazione del client.

L’elemento “flow” ha un attributo “aor” e un elemento “timestamp” che indica il timestamp dell’ultimo pacchetto ricevuto, quindi quello che si aspetta di ricevere dal B2BUA dopo essersi connesso al nuovo AP di destinazione.

CAPITOLO 7

Fase di test per la soluzione proposta.

Questo capitolo è dedicato alla rilevazione dei tempi di invio e ricezione dei messaggi SIP lato client e della percentuale di utilizzo di CPU per il Client, il Redirect Server, il Creator e il B2BUA. Le macchine utilizzate per i nostri test sonocaratterizzate dalle seguenti caratteristiche: Pentium IV, processore 2 GHz, memoria 512 MB, sistema operativo Linux, versione java virtual machine 1.4.2-01.

7.1 Tempi rilevati.

Sono stati rilevati sul lato client:

- Intervallo tra l'invio del messaggio INVITE e l'invio del messaggio ACK da parte del Client verso il Redirect Server.

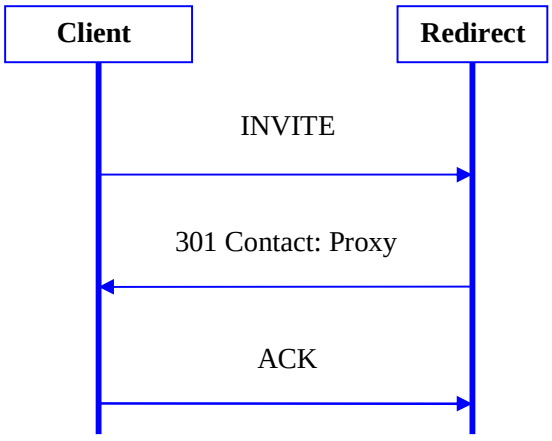

Figura 7.1 Primo intervallo rilevato.

- Intervallo tra l'invio del messaggio INVITE e l'invio del messaggio ACK da parte del Client verso il B2BUA.

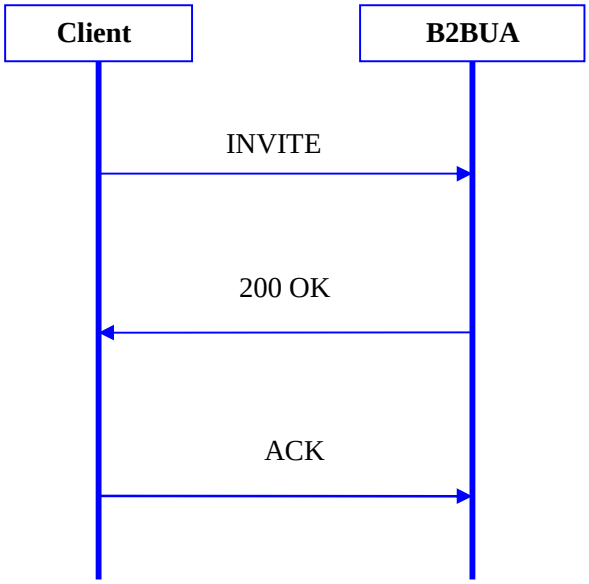

Figura 7.2 Secondo intervallo.

- Intervallo tra l'invio della prima NOTIFY da parte del Client al B2BUA e la ricezione del messaggio di risposta corrispondente (200 OK).

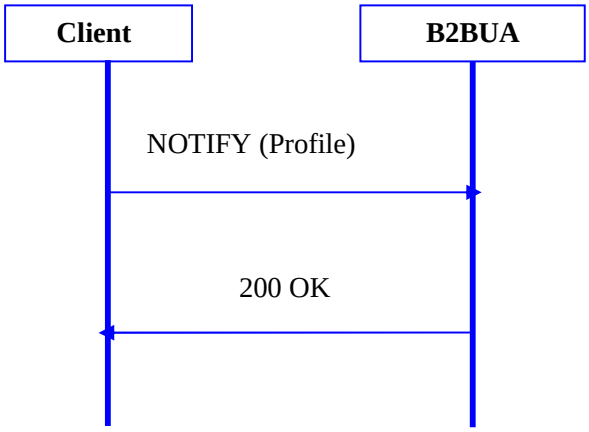

Figura 7.3 Terzo intervallo.

- Intervallo tra l'invio della seconda NOTIFY da parte del Client al B2BUA e la ricezione del messaggio di risposta corrispondente (200 OK).

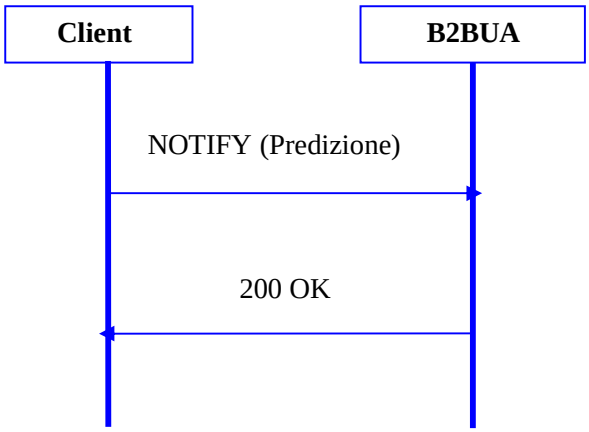

Figura 7.4 Quarto intervallo.

- Intervallo tra l'invio della terza NOTIFY da parte del Client al B2BUA e la ricezione del messaggio di risposta corrispondente (200 OK).

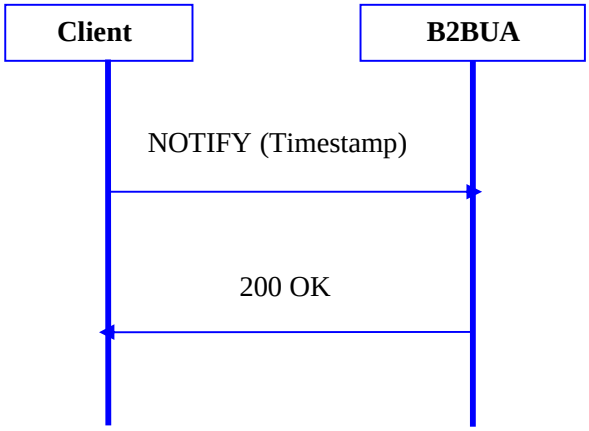

Figura 7.5 Quinto intervallo.

- Intervallo tra il messaggio di BYE inviato dal Client al B2BUA e la corrispondente risposta 200 OK.

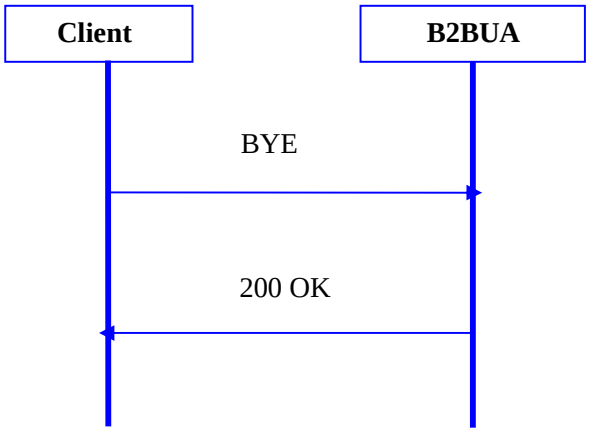

Figura 7.6 Sesto intervallo.

7.2 Utilizzo CPU.

Abbiamo deciso di rilevare anche la percentuale di utilizzo di CPU per:

- Il Redirect Server nel momento in cui viene messo in esecuzione, quindi nella fase di inizializzazione, e durante la ricezione di richieste provenienti da diversi Client.
- Il Creator nel momento della messa in esecuzione e durante la ricezione di richieste.
- Il B2BUA nel momento in cui riceve richieste e risposte.
- Il Client nel momento in cui viene messo in esecuzione e nell'invio dei messaggi di richiesta e risposta.

7.3 Dati rilevati.

Poiché la nostra soluzione si concentra principalmente sul micro handoff, quindi avviene solo un cambio di AP all'interno della stessa sottorete, senza che ci siano problemi di cambiamento di indirizzo IP e di autenticazione, autorizzazione e accounting (AAA), abbiamo deciso di simulare il tempo necessario per il cambio di AP solo mettendo il client nello stato di sleep(); per un certo intervallo di tempo, poiché il nostro obiettivo è quello di rilevare i tempi di invio e ricezione dei messaggi SIP e l'utilizzo di CPU durante lo scambio di tali messaggi. In laboratorio sono state effettuate diverse prove, abbiamo rilevato:

- I tempi tra l'invio e la ricezione dei messaggi SIP inserendo il client su una macchina e i server su un'altra macchina.
- I tempi tra l'invio e la ricezione dei messaggi SIP inserendo il client su una macchina e ogni server su una macchina diversa.

- I tempi tra l'invio e la ricezione dei messaggi SIP inserendo un client per macchina e ogni server su una macchina diversa.

I tempi sono stati rilevati utilizzando come unità di misura il millisecondo.

7.3.1 Tempi rilevati per i messaggi.

Analizziamo in questo paragrafo i tempi rilevati per le tre tipologie di test eseguite in laboratorio e descritte nel precedente paragrafo. I dati seguenti sono stati ottenuti dopo aver eseguito circa 50 prove per ogni tipologia di test, quindi sono dei valori medi.

I tempi rilevati nella prima tipologia di test:

INTERVALLO DI INTERESSE	MILLISECONDI
INVITE REDIRECT / ACK	169
INVITE B2BUA / ACK	30
Prima NOTIFY / 200 OK	27
Seconda NOTIFY / 200 OK	23
Terza NOTIFY / 200 OK	17
BYE / 200 OK	13

Figura 7.7 Tabella degli intervalli per la prima tipologia di test.

Per la seconda tipologia si hanno:

INTERVALLO DI INTERESSE	MILLISECONDI
INVITE REDIRECT / ACK	187
INVITE B2BUA / ACK	44
Prima NOTIFY / 200 OK	27
Seconda NOTIFY / 200 OK	23
Terza NOTIFY / 200 OK	17
BYE / 200 OK	13

Figura 7.8 Tabella degli intervalli per la seconda tipologia di test.

Per la terza tipologia:

INTERVALLO DI INTERESSE	MILLISECONDI
INVITE REDIRECT / ACK	196
INVITE B2BUA / ACK	43
Prima NOTIFY / 200 OK	32
Seconda NOTIFY / 200 OK	28
Terza NOTIFY / 200 OK	17
BYE / 200 OK	13

7.9 Tabella degli intervalli per la terza tipologia di test.

7.3.2 Percentuale di utilizzo di CPU.

La tabella sottostante racchiude le percentuali di utilizzo di CPU dell'applicazione Client, Server Redirect, Creator e B2BUA quando eseguono ognuna su una macchina diversa.

	REDIRECT SERVER	CREATOR	B2BUA	CLIENT
AVVIO DELL'APPLICAZIONE	18.9%	16.8%		25.4%
DURANTE IL DIALOGO	0.9%	0.3%	0.8%	5.3%

Figura 7.10 Tabella delle percentuali di utilizzo di CPU.

All'avvio dell'applicazione si ha un maggiore utilizzo di CPU in quanto devono essere caricate le librerie utilizzate e devono essere creati per ogni applicazione diversi elementi. Nel momento in cui le entità sono state avviate e si procede all'invio e alla ricezione dei messaggi l'utilizzo di CPU diminuisce sensibilmente ed è uno dei fattori positivi che devono essere notati, in quanto è importante nel momento in cui sarà integrato con il livello di trasporto.

7.3.3 Valori di picco.

Analizziamo in questo paragrafo le variazioni degli intervalli rilevati per le tre prove eseguite in laboratorio quando il numero di client messi in esecuzione contemporaneamente è uguale a 30.

Nel primo caso abbiamo notato che l'intervallo tra l'invio del messaggio INVITE e l'invio del messaggio ACK e tra l'invio dei messaggi di NOTIFY (ricordiamo che abbiamo ipotizzato almeno tre NOTIFY per ogni dialogo) e la ricezione dei messaggi di risposta non cambia aumentando il numero di client che contemporaneamente vengono messi in esecuzione, questo è un risultato che ci aspettavamo poiché ogni client ha un proprio B2BUA, quindi non deve condividerlo con altri. Così come l'intervallo tra l'invio del messaggio BYE e la ricezione della risposta. L'intervallo che cambia con il variare del numero di Client messi contemporaneamente in esecuzione è tra l'INVITE e l'ACK inviati dal Client al Redirect Server. Abbiamo notato che l'intervallo varia dai 169 millisecondi ai 189 millisecondi.

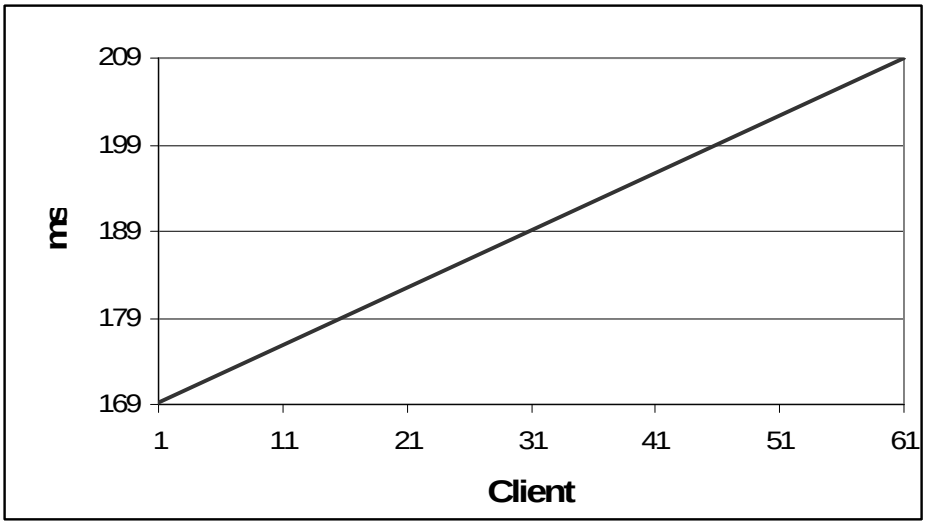

Figura 7.11 Variazione intervallo.

Nella seconda prova, mettendo in esecuzione contemporaneamente 30 client, valgono per i messaggi tra Client e B2BUA le stesse considerazioni fatte precedentemente. Per l'intervallo tra INVITE / ACK inviati dal Client al Redirect si ha una variazione di circa 20 millisecondi.

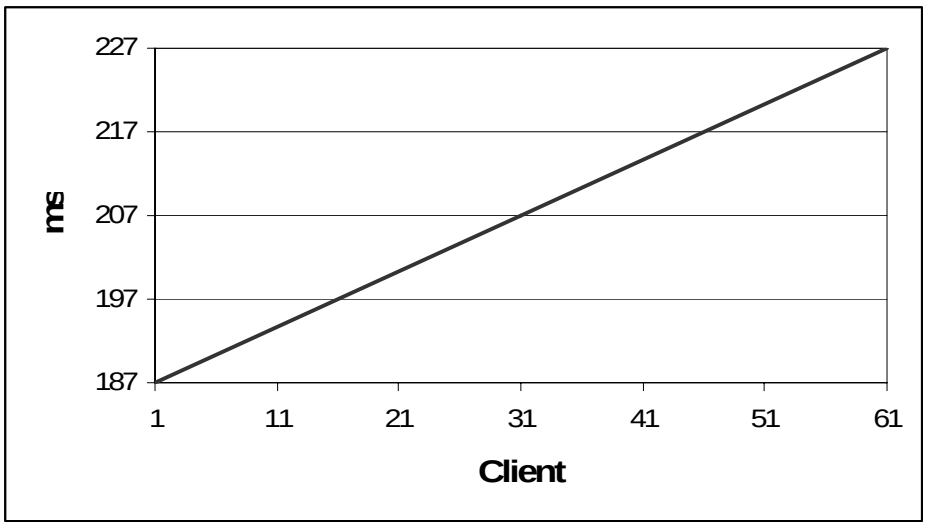

Figura 7.12 Variazione intervallo.

Nella terza prova per quanto riguarda l'intervallo tra l'invio dell'INVITE e l'invio dell'ACK dal Client al Server Redirect si hanno delle variazioni di circa 20 millisecondi, variazione che non accade per gli intervalli tra le richieste e le risposte spedite dal Client al B2BUA.

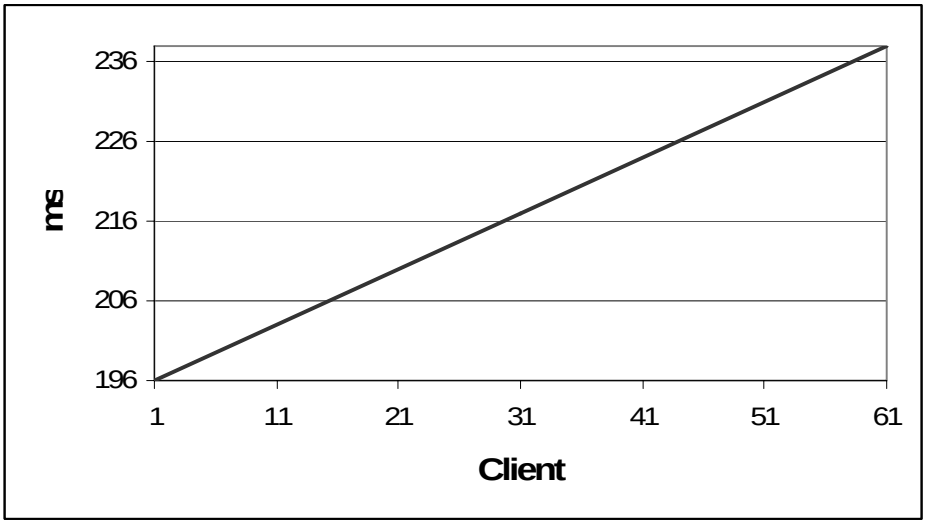

Figura 7.13 Variazione intervallo.

Dopo aver osservato i tempi e le variazioni abbiamo deciso di capire quale potesse essere il collo di bottiglia, ossia se il problema delle variazioni, conseguenti al crescere del numero di client messi in esecuzione contemporaneamente, è imputabile al Server Redirect o al Creator.

In un primo momento abbiamo pensato che il problema potesse essere il Redirect Server, quindi abbiamo inserito due Redirect, nonostante ciò la variazione non diminuiva.

In un secondo momento è stata introdotta una modifica sul Creator: nel momento in cui viene messo in esecuzione crea un pool di circa 30 B2BUA, thread messi in esecuzione con il metodo start();, poiché non hanno ancora ricevuto richieste o risposte da un client entrano nello stato di wait(); per uscirvi nel momento in cui sopraggiungono messaggi attraverso il metodo notify();. Abbiamo notato che la variazione diminuisce, infatti si ha una diminuzione di circa 14 millisecondi. Possiamo concludere che una possibile soluzione è quella di introdurre più Creator capaci di creare inizialmente ciascuno un pool di 30 Thread. Nel momento in cui il

Creator non ha più thread a disposizione potrebbe avvisare il Redirect e quest'ultimo potrebbe inviare le richieste ad un altro Creator.
Di seguito riportiamo le variazioni che si hanno apportando tale modifica, ossia inserendo due Creator con un pool di 30 Thread e ipotizzando che contemporaneamente vengano messi in esecuzione 60 Client:

- I tempi tra l'invio e la ricezione dei messaggi SIP inserendo tutti i client su una macchina e tutti i server su un'altra macchina.

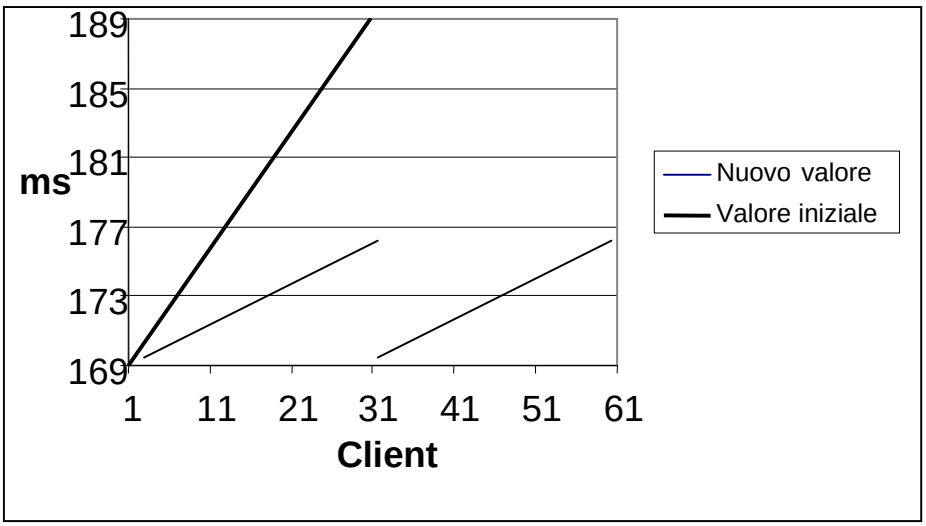

Figura 7.14 Variazione intervallo.

- I tempi tra l'invio e la ricezione dei messaggi SIP inserendo tutti i client su una stessa macchina e ogni server su una macchina diversa.

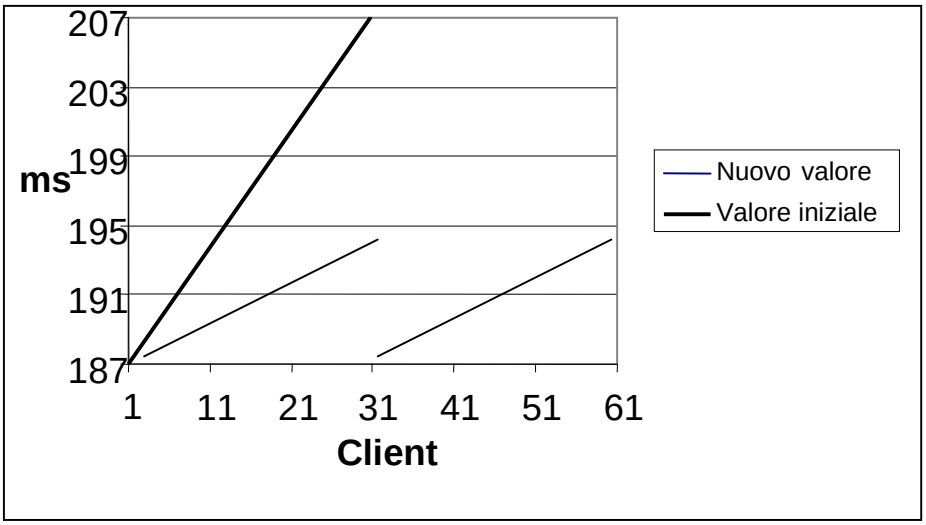

Figura 7.15 Variazione intervallo.

- I tempi tra l'invio e la ricezione dei messaggi SIP inserendo un client per macchina e ogni server su una macchina diversa.

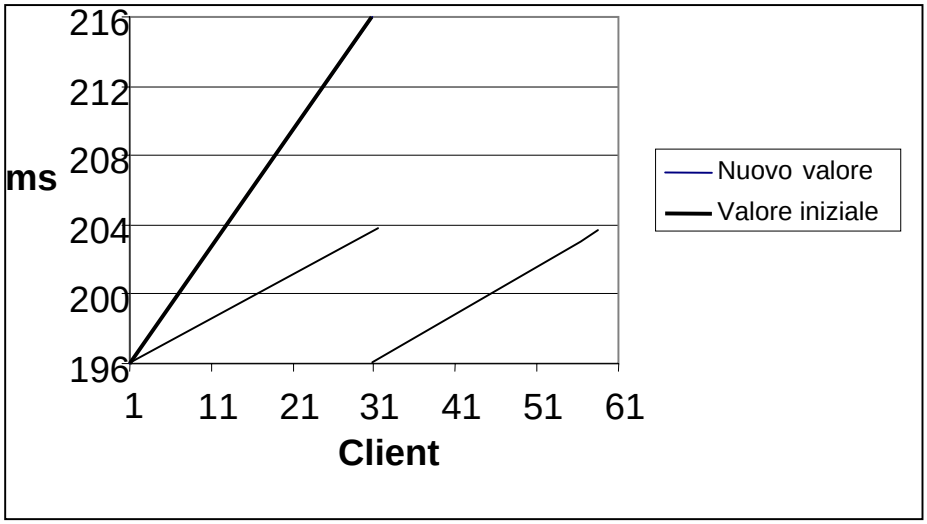

Figura 7.16 Variazione intervallo.

Conclusioni

In questa tesi ci siamo occupati di gestire il processo di handoff, la conseguente perdita di dati transienti e la continuità di sessione durante gli spostamenti del terminale mobile utilizzando il protocollo di sessione SIP e le sue estensioni. In particolare, abbiamo individuato tre possibili tipologie di handoff, i.e., micro, macro e global, e proposto un protocollo per gestirle. Ci siamo poi concentrati sul micro-handoff proponendo una possibile implementazione.

Per la gestione dell'handoff abbiamo deciso di utilizzare e creare un'architettura proxy-based, ossia un'architettura che prevedesse un'entità intermedia tra Client e Server, affinché i dati inviati dal Server al Client, durante lo spostamento del terminale, non andassero persi. Tale proxy funge da intermediario tra le due entità e prevede un buffer in cui bufferizzare i dati transienti in arrivo dal Server e diretti al Client.

Per poter anticipare alcune operazioni necessarie a velocizzare l'handoff e quindi anticipare la gestione della riconfigurazione della sessione dopo lo spostamento del terminale, abbiamo introdotto sul nodo Client una tecnica di predizione capace di predire gli spostamenti del terminale da un access point ad un altro access point.

Abbiamo proposto un'estensione dell'event framework SIP, in particolare un nuovo event package che coordina diversi eventi, necessari al proxy per gestire il processo di handoff.

L'implementazione è stata seguita da una fase di testing volta a rilevare i tempi di esecuzione del protocollo proposto e l'efficienza del prototipo realizzato. I risultati ottenuti sono stati soddisfacenti: i tempi tra l'invio di una richiesta e la ricezione della corrispondente risposta sono dell'ordine dei 200 millisecondi e

l’overhead introdotto è limitato, infatti la percentuale di utilizzo di CPU durante l’invio e la ricezione dei messaggi sia lato client sia lato proxy è dello 0.7%.

Il protocollo creato è utilizzabile da qualsiasi applicazione e può essere impiegato indipendentemente dalla tecnologia con cui sono stati realizzati gli strati sottostanti. Un altro elemento positivo è che può essere adottato sia per un handoff verticale sia per un handoff orizzontale.

Possibili sviluppi futuri potrebbero essere la realizzazione prototipale anche dei protocolli proposti per il macro e il global handoff e l’integrazione del livello di sessione con il livello di trasporto dati.

Attualmente i messaggi SIP non prevedono il trasporto, nel body, dei dati multimediali. Abbiamo notato comunque che è possibile utilizzare il protocollo Internet **Multipurpose Internet Mail Extensions** (MIME) per “imbustare” i dati multimediali e inserirli nel body dei messaggi, infatti il protocollo SDP è solo un possibile protocollo da utilizzare nei messaggi SIP. Sarebbe, dunque, interessante, in sviluppi futuri approfondire ed estendere questo aspetto.

Bibliografia

[1]. J. Kurose, K. Ross, “Internet e reti di calcolatori”, McGraw-Hill, 2003.

[2]. R. Flickenger, “Costruire reti wireless”, Milano: Hops tecniche nuove, 2003.

[3]. W. Stallings, “Comunicazioni e reti wireless”, McGrawHill, Milano 2003.

[4]. IEEE 802.11: www.ieee.org/

[5]. J. Rosenberg et al., “SIP: Session Initiation Protocol”, IETF RFC 3261, Giugno 2002.

[6]. R.Bolla, “Session Initiation Protocol”, Corso di Telematica 2, 2004, Facoltà di Ingegneria, Università di Genova.

[7]. SIP: http://www.sipcenter.com.

[8]. M. Handley, V.Jacobson, ISI/LBNL, “Session Description Protocol (SDP)”, RFC 2327, 1998.

[9]. SIP Server: http://www.voip-info.org/.

[10]. A. B. Roach, “SIP-Specific Event Notification”, IETF RFC 3265, 2002.

[11]. J. Rosenberg, “An INVITE Initiated Dialog Event Package for the Session Initiation Protocol (SIP)”, Internet-Draft, Aprile 2005.

[12]. J. Rosenberg, “A Session Initiation Protocol (SIP) Event Package for Registrations”, IETF RFC 3680, Marzo 2004.

[13]. H. Schulzrinne, E. Wedlund, “Application-Layer Mobility Using SIP”, Columbia University, Marzo 2003.

[14]. Ian F. Akyildiz, J. Xie, S. Mohanty, ”A Survey of Mobility Management in Next-Generation All-IP-Based Wireless Systems”, Georgia Institute of Technology, 2004.

[15]. A. Dutta et al., “Fast-handoff Schemes for Application Layer Mobility Management”, IEEE PIMRC, Spagna, 2004.

[16]. H. Schulzrinne et al., “RTP: A Transport Protocol for Real-Time Applications”, IETF, RFC 3550, Luglio 2003.

[17]. Real Time Protocol: http://www.cs.columbia.edu/hgs/rtp

[18]. D.Saha et al., “Mobility Support in IP: a Survey of Related Protocols”, IEEE Network, Vol. 18, No. 6, 2004.

[19]. JAIN SIP Developer Tools: https://jain-sip.dev.java.net

[20]. NIST: http://www-x.antd.nist.gov/proj/iptel

[21]. Java Development Kit – J2SE 5.0, http://java.sun.com

[22]. C. Kelley, B. Stanko, “Guida a XML”, Mc Graw Hill, 2004.