Introduzione

Questi ultimi anni hanno visto la rapida crescita e diffusione delle tecnologie wireless, ovvero quelle tecnologie che ci permettono di utilizzare normali strumenti come il computer oppure il telefono senza la scomodità di avere un cavo che ci vincola ad una posizione fissa. Gli utenti richiedono sempre più spesso di accedere con i propri dispositivi, solitamente assai limitati, a servizi tradizionalmente acceduti da postazioni fisse generalmente più potenti. Non sempre però si riesce a far fronte a questo tipo di richiesta senza difficoltà; l'accesso senza fili introduce infatti alcuni problemi ancora senza soluzione legati al fatto che gli utenti si possono muovere durante l'erogazione del servizio.

Alcune applicazioni risentono degli spostamenti dell'utente in maniera percepibile dall'utente stesso: tra queste applicazioni focalizzeremo l'attenzione su quelle multimediali, come ad esempio player video e audio. Infatti perdite temporanee della connessione, ad esempio dovute al movimento dell'utente tra zone servite da diversi punti di accesso alla rete fissa, possono causare interruzioni nella riproduzione video o audio. Questo avviene perché nessuno strumento è stato predisposto per la risoluzione di questo particolare problema. Le applicazioni multimediali non sono le uniche ad essere soggette a queste interruzioni, ma quelle in cui più facilmente ci si può accorgere di questo problema. Per garantire la completa mobilità dell'utente nasce la necessità di una infrastruttura che garantisca continuità di servizio mentre questo si sposta.

Lo scopo di questa tesi vuole quindi essere la proposta di una infrastruttura che offra il necessario supporto agli utenti mobili e si adatti alle loro esigenze facendogli percepire l'erogazione del servizio come continua anche durante i loro spostamenti. In particolare, ci concentreremo sulle applicazioni multimediali per la richiesta di video, progettando un'applicazione che non risenta delle interruzioni durante la riproduzione del filmato ed una infrastruttura che riesca a seguire l'utente nel suo spostamento. Parte del progetto è mirata a raccogliere informazioni sui dispositivi di rete in dotazione all'utente in modo da poter adattare l'applicazione stessa alle esigenze specifiche del dispositivo da cui l'utente richiede il servizio. Le attuali realizzazioni degli standard wireless maggiormente diffusi, come 802.11, si differenziano infatti molto le une dalle altre e questo aspetto influenza le prestazioni di una infrastruttura come quella che si vuole progettare.
Il nostro obiettivo è quello di far fronte alle esigenze degli utenti senza fare ipotesi troppo stringenti sulle caratteristiche dei dispositivi wireless, proprio per andare incontro alla naturale evoluzione di questa tecnologia.

Nasce quindi la necessità di verificare la validità dell'infrastruttura progettata all'interno di un numero elevato di contesti diversi tra loro. E' però difficile pensare di avere a disposizione una così ampia varietà di contesti dove sperimentare le soluzioni progettate. E' nata così l'esigenza di uno strumento che permettesse di simulare contesti wireless variegati ed effettuare test sulle applicazioni progettate senza provarle in un contesto reale. Nel corso della tesi si svilupperà un componente software che offra alle

applicazioni multimediali la stessa visione del mondo che avrebbero su un normale dispositivo.

Era necessario non solo offrire l'astrazione di un contesto wireless reale, ma anche emulare il comportamento dei dispositivi che permettono l'accesso a queste reti poiché il loro comportamento ha una forte influenza sulle prestazioni delle applicazioni multimediali. L'emulatore è stato pertanto progettato come composto da due livelli distinti: il primo ha il compito di simulare più contesti wireless reali e di offrire al livello soprastante una visione del tutto simile a quella che si avrebbe in una rete vera.
Il secondo livello svolge invece la funzione di emulare il comportamento dei dispositivi di rete che offrono agli utenti la possibilità di connettersi alle reti in modo che l'applicazione multimediale sia circondata da un ambiente caratterizzato dagli stessi problemi offerti da una reale connessione wireless. L'emulazione dovrà inoltre essere invisibile all'infrastruttura progettata: questo significa che sarà possibile sperimentare la validità dell'infrastruttura ideata per garantire all'utente la continuità di servizio senza apportare alcuna modifica all'infrastruttura stessa come normalmente avviene negli ambienti simulati.

La tesi verrà spiegata lungo i seguenti capitoli. Nel primo si darà un'idea del contesto in cui il progetto si va ad inserire, spiegando le problematiche ancora non risolte e lo stato dell'arte su cui il progetto si basa. Nel capitolo 2 verrà spiegato dettagliatamente cos'è una rete wireless, da cosa è composta e lo standard 802.11, più altri concetti fondamentali per questo tipo di tecnologie.
Il capitolo 3 ha invece lo scopo di illustrare l'analisi del progetto, ovvero di caratterizzare la struttura dell'applicazione mettendo in evidenza unicamente le sue funzionalità senza scendere nel dettaglio.
I capitoli 4 e 5 saranno invece dedicati all'implementazione delle applicazioni e risulteranno quindi molto più tecnici rispetto ai precedenti.
L'ultimo capitolo sarà invece dedicato alla presentazione dei risultati ottenuti e agli sviluppi futuri che potrebbero essere realizzati seguendo la linee guida di questa tesi.

Capitolo 1 – Scenario Applicativo

In questa prima parte introduttiva illustrerò brevemente l'argomento di questa tesi e darò una sorta di traccia su cui poi si svilupperà il progetto. In primo luogo è necessario fornire al lettore un'idea dell'ambiente in cui si inserisce il progetto e dello stato dell'arte del settore prima di questa tesi, per elencarne le problematiche e soprattutto per guidare alla comprensione dei motivi che hanno spinto a ricercare soluzioni in questo particolare ambito.
Prima di tutto è utile dire che tutto il lavoro riguarda le reti wireless e si sofferma su un aspetto particolare di quest'ultime; il supporto a utenti che si muovano fra celle diverse e vogliano mantenere attive la proprio sessione di lavoro.
Farò quindi prima una panoramica dei problemi legati all'utilizzo di queste nuove tecnologie per evidenziarne alcuni tratti fondamentali che ne hanno fatto un tema di interesse centrale nel panorama moderno e come hanno cambiato le nostre abitudini e le nostre esigenze

Questa tesi si può inoltre dividere in due parti: la prima riguarda la soluzione proposta garantire continuità nell'erogazione di servizi multimediali durante gli spostamenti degli utenti in un ambiente wireless.
La seconda parte riguarderà invece lo sviluppo di un emulatore di una rete wireless che consenta di provare la validità delle applicazioni sviluppate in diversi scenari difficilmente attuabili in una rete wireless reale.

1.1 - Reti Wireless

Per reti wireless si intende in generale, un serie di dispositivi che vanno dai PC ai cellulari, collegati tra loro senza mezzi fisici come ad esempio cavi o fili di qualsiasi genere.
La comunicazione tra le varie macchine avviene tramite apposite antenne che funzionano sia da trasmittente che da ricevente e fanno le veci dei collegamenti fisici più comunemente usati.
Il primo vantaggio evidente è che non c'è bisogno di una postazione fissa per partecipare ad una rete ovvero questo tipo di tecnologia ci consente una certa mobilità e ci svincola dalla presenza e disponibilità di connettori e postazioni adatte ad ospitare, ad esempio, un PC. L'utente ora può semplicemente munirsi di un portatile e connettersi alla rete da ovunque egli ritenga più comodo, a patto che si trovi all'interno di un'area coperta da una rete Wireless: sebbene non necessiti di cavi la rete wireless ha comunque un'estensione massima che è data dalla portate delle antenne che trasmettono e ricevono il segnale dall'utente. Infatti quando ci connettiamo ad una rete di questo tipo in realtà lo facciamo comunicando con antenne che fungono da punti di accesso alla rete stessa e che vengono chiamati Access Point (AP). Oltre alla mobilità le reti Wireless hanno anche portato ad una più ampia estensione dell'accesso ad Internet: prima dell'avvento di questa tecnologia per accedere ai servizi legati ad Internet era necessario possedere una

connessione privata oppure poter accedere a postazioni pubbliche che avevano comunque il limite di avere di poter offrire un numero limitato di accessi.
In un mondo che ci spinge a postazioni di lavoro sempre meno fisse e che sta facendo degli spostamenti rapidi un uso sempre più quotidiano, è facile pensare come questo tipo di tecnologie abbiano trovato terreno fertile per svilupparsi. Basti pensare ad esempio allo sviluppo delle reti cellulari per farsi un'idea del cambiamento avvenuto negli ultimi anni.

1.1.1 - Confronto con le reti fisse

Oltre a questi indubbi vantaggi che hanno guidato ad un'espansione sempre maggiore di queste reti, queste tecnologie comportano anche alcuni svantaggi non trascurabili rispetto alle reti fisse.
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.
Garantire un accesso più esteso ad Internet vuol dire anche avere un maggior numero di utenti da servire ma non implica avere più banda disponibile. 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 di loro.
Questo è un problema che in generale affligge tutte le reti ma in particolare le reti Wireless poiché, come detto, danno modo ad un maggior numero di utenti di collegarsi.
Questo avviene se non si predispone un controllo degli accessi al servizio ovvero se si consente a chiunque voglia di accedere alla rete, cosa che può essere limitata accettando solo un numero di utenti tale da poter garantire una certa qualità di servizio.
Tradizionalmente l'accesso alla rete Internet veniva effettuato da macchine con determinate caratteristiche che permettevano di accomunarle ed in generale si parlava di computer dalle diverse potenze e prestazioni, ma caratterizzati da una struttura comune.
Questi nuovi tipi di reti invece prevedono l'accesso di dispositivi che presentano una maggiore varietà e vanno da potenti laptop a dispositivi più limitati come i Personal Digital Assistans (PDA).Questo fa si che difficilmente si possano fare ipotesi sul dispositivo con cui ci si trova a dialogare e quindi più difficilmente si riesce ad adattarsi alle sue esigenze specifiche. Inoltre reti di questo tipo sono più inaffidabili rispetto alle fisse e la perdita di informazione può essere sensibilmente maggiorepoiché 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 e sensibile anche alle interferenza con altri tipi di strumentazione che comunicano utilizzando lo stesso mezzo trasmissivo.
Altro problema da considerare per completare il panorama è quello della sicurezza che, sebbene non centrale in questo elaborato, è invece piuttosto rilevante nella diffusione dell'uso di queste reti ad esempio in uffici o banche: questo problema è legata 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.

1.1.2 - Handoff

Abbiamo visto che per poter accedere ad una rete Wireless dobbiamo comunque instaurare una comunicazione con un punto di accesso chiamato Access Point. Ognuna di queste antenne ha però una portata limitata, il che vuol dire che il suo segnale non può raggiungerci ovunque ma ha dei limiti ben precisi e che variano a seconda della tecnologia utilizzata. La qualità del segnale aumenta con la vicinanza a tali AP e degrada più ci allontaniamo fin quando non è più sufficiente a mantenere una comunicazione. Per questo motivo si dispongono più AP in modo da coprire l'area desiderata.
Accade così che mentre ci allontaniamo da un AP ci avviciniamo al successivo e ad un certo punto perdiamo la comunicazione con l'AP precedente per instaurarla con quello successivo.
La rete cellulare per la telefonia funziona allo stesso modo, facendoci comunicare in ogni istante con l'antenna più vicina a noi.
Questo processo di distaccamento e successiva riconnessione prende il nome di handoff ed in generale può essere diverso a seconda della tecnologia wireless utilizzata

1.1.3 – Handoff in reti 802.11

Finora abbiamo parlato di reti wireless senza scendere nel dettagli di una specifica tecnologia ma prendendo come unica caratteristica quella di non necessitare di mezzi fisici di comunicazione ed utilizzare antenne per la comunicazione.
Sotto questo aspetto anche la rete di telefonia mobile cellulare rientra sotto la definizione di rete Wireless.
Questo progetto tratta di una particolare tecnologia Wireless chiamata 802.11 che oggi viene spesso identificata col nome di Wi-Fi.
La diffusione su larga scala di questa tecnologia ha fatto si che , quando si parla di reti wireless, ci si riferisca implicitamente a questo tipo di tecnologia.
Le principali caratteristiche di 802.11 verranno descritte in dettaglio in un capitolo dedicato e per ora vogliamo focalizzare l'attenzione su un unico particolare.
Una delle principali differenze, ad esempio con la telefonia mobile, di questa particolare tecnologia wireless riguarda la gestione dell'Handoff: è esperienza comune che durante una telefonata non avvertiamo interruzioni nemmeno andando in macchina oppure in treno passando diverse volte da un AP all'altro. Questo perché la tecnologia cellulare gestisce il passaggio da un AP all'altro in modo da non perdere dati durante gli handoff.
802.11 invece non gestisce l'handoff in questo modo e durante questo processo alcuni dei dati trasmessi possono essere persi.
L'utente avverte quindi generalmente un momento di buio nella comunicazione che può essere più o meno evidente da seconda delle applicazioni usate in quel momento.
Senza scendere nel dettaglio risulta intuitivo come applicazioni multimediali come quelle elencate in precedenza possano essere particolarmente sensibili a questo tipo di evento.

1.2 – Erogazione di servizi continui in rete Wireless

1.2.1 - Servizi multimediali

Attraverso una rete wireless è possibile offrire praticamente gli stessi servizi che in 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. In particolare noi ci occuperemo dell'erogazione di servizi multimediali.
Con servizio multimediale intendiamo generalmente la trasmissione di video audio oppure entrambe le cose insieme.
Esempi tipici di questi servizi sono:

- *Video conferenza o Voice On IP*: in entrambi i casi due o più utenti vengono fatti collegare tra loro per comunicare simulando una conferenza all'interno di una stessa stanza. E' quindi possibile trasmettere l'immagine e la voce di chi parla a tutti i partecipanti. 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.
- *Video on Demand* (Vod): in questo caso l'utente richiede di visualizzare un filmato che risiede in qualche archivio remoto. In questo caso il flusso dei dati è praticamente monodirezionale e va dalla macchina che possiede il filmato all'utente che lo richiede.
- *Live*: anche in questo caso il flusso di informazioni è monodireziole 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.

Nell'ambito di questo progetto considereremo soprattutto degli ultimi due tipi di servizio, ovvero Video on demand e Live. Ciò che principalmente li differenzia dalla video conferenza è che quest'ultima prevede che la comunicazione avvenga tra pari, ovvero ogni utente ha lo stesso identico ruolo ti trasmettitore e di ricevente e quindi prevede l'interazione tra le varie parti e quindi impone vincoli molto più stringenti di comunicazioni real-time poiché l'impressione deve essere di non avere alcun ritardo nella comunicazione con gli altri partecipanti.

1.2.2 - Continuità di servizio.

Il servizio di streaming video è l'argomento intorno al quale ruota questo progetto e che quindi ha portato allo sviluppo delle sue parti ma non dev'essere considerato a se stante se si vuole inquadrare l'idea del contesto in cui si va ad inserire: la richiesta di video è senza dubbio una delle applicazioni che più mette in evidenza le problematiche legate alla mobilità all'interno della rete wireless ma non certo l'unica.
Questo progetto si inserisce in un contesto che vuole offrire ad ogni utente una continuità di servizio dal momento in cui richiede di essere accolto nel sistema al momento in cui

decide di lasciarlo senza fargli percepire la perdita di connessione che avviene normalmente quando si muove da un AP all'altro.
Lo scopo di questa tesi sarà quindi quello di risolvere questo tipo di situazioni in modo da non fare avvertire all'utente il momento di handoff.
Visto che 802.11 non propone una soluzione a questo tipo di problema e ad altri legati a questa tecnologia si rende necessario lo sviluppo di uno strato intermedio tra le applicazioni e la rete stessa che tenga in considerazione le problematiche relative a questo particolare tipo di tecnologia.
Uno strato di questo tipo viene generalmente indicato come middleware cioè qualcosa che si colloca a metà tra diverse applicazione e che ha appunto l'utilità di far comunicare le due parti offendo soluzioni ad un certo tipo di problemi.
Si mira quindi alla costruzione di una infrastruttura che sia dedicata alla gestione degli utenti mobili e che possa offrire una serie di servizi pensati per i problemi specifici che questi utenti comportano.

Il sistema non vuole inoltre fare ipotesi troppo restrittive sulle risorse a disposizione dall'utente come la potenza del suo processore o la quantità di memoria a lui disponibile.
E' chiaro che esiste un minimo di risorse che l'utente deve possedere se vuole accedere a determinati servizi, ma la direzione è quella di adattarsi proprio alle risorse che l'utente mette a disposizione.
Volendo trovare soluzione ad un problema al momento non risolto è chiaro che l'infrastruttura dovrà mettere a disposizione alcuni strumenti propri e che eventualmente dovrà fornire alcuni strumenti anche all'utente stesso.

1.2.3 - Il servizio di streaming

Dopo aver dato un'idea generale del contesto focalizziamo ora l'attenzione sul problema della continuità di servizio durante lo streaming video.
L'utente, una volta connesso al sistema globale, verrà informato di una serie di servizi a cui ha accesso ed uno dei quali sarà appunto la richiesta di poter vedere una serie di filmati oppure di conferenze in corso.
La configurazione più semplice in questo caso è quella tipica Client – Server in cui il processo Client, che risiede nella macchina dell'utente, invia una richiesta ad un Server in ascolto. Una volta che il Server risponde affermativamente alle richiesta del Client quest'ultimo si mette in ricezione dei dati mentre il Server provvede a trasmetterli.
In questo tipo di applicazioni è previsto che il Client visualizzi lo stream video mentre lo riceve e non lo richieda nel suo complesso prima di visualizzarlo. Questo perché il filmato potrebbe essere troppo lungo oppure perché potrebbe essere ripreso in diretta come nei casi di Video on Demand e Live.
In questo tipo di applicazioni non è previsto che il Client confermi continuamente al Server la ricezione dei dati: la principale caratteristica degli stream video attraverso reti di PC è la quantità considerevole dei dati da trasmettere e le tempistiche da rispettare affinché il filmato risulti continuo. Visti i vincoli che impongono i software real-time si

preferisce lasciare all'applicazione la gestione dei dati persi, se chiedere la ritrasmissione oppure decidere di rimediare altrimenti.
Un protocollo che preveda la conferma di tutti i dati spediti verso il Client risulta notevolmente più pesante e rallenta la trasmissione in quanto ogni dato trasmesso deve essere confermato e perché anche le conferme stesse potrebbero essere perdute. Inoltre non lascia il controllo all'applicazione che invece è desiderabile per poter meglio decidere come porre rimedio ad una situazione di perdita di informazioni.
In questo tipo di applicazioni si utilizzano quindi protocolli senza connessione proprio perché si predilige la prestazione all'affidabilità
Il Server che trasmette il filmato potrebbe risiedere in una località remota ed inoltre dover servire una quantità massiccia di richieste di stream video.
Quando l'utente muovendosi cambia AP, ovvero la scheda della sua macchina decide di connettersi ad un'altra antenna perché il segnale dell'antenna precedente sta venendo meno per un intervallo di durata variabile il Client non riceve i dati spediti dal Server con il risultato che parte del filmato viene perso.
Utilizzando protocolli senza connessione non c'è una riparazione automatica a questo tipo di perdite e diventa quindi compito dell'applicazione stessa riconoscerle ed eventualmente rimediare.
Spesso in questi casi i dati vengono persi ed il Client ricomincia a visualizzare il filmato solo quando nuovi dati vengono ricevuti.

1.2.4 - Il concetto di sessione

Sarebbe auspicabile poter dare all'utente l'idea di essere sempre connesso e di poter caratterizzare la sua partecipazioni non come il tempo tra una connessione ed il termine della stessa ma come il periodo in cui egli richiede i servizi alla rete.
In generale una sessione può essere identificata come il periodo compreso tra la richiesta da parte dell'utente di essere accolto dal sistema globale al momento il cui l'utente decide di allontanarsi e di uscire dal sistema.
Quando utilizziamo una rete wireless è piuttosto frequente un numero elevato di handoff. Questo fa si che l'effetto a livello applicativo sia simile al distacco del cavo Ethernet in modo intermittente.

Quello che si vuole offrire all'utente è l'astrazione di essere collegato al sistema globale che offra un'interfaccia all'utente per poter accedere alle sue risorse e che renda trasparente all'utente le continue connessioni ad AP diversi mentre egli si muove.
Per far questo si rende necessaria la creazione di un'entità che segua l'utente nei suoi spostamenti, che tenga memoria delle sue scelte e delle sue caratteristiche.
Una delle ipotesi che ha condotto a queste scelte è che l'utente sia riconosciuto in base alla sua identità, che possa aver espresso delle preferenze se è un utente conosciuto e che possa ad esempio connettersi ogni volta con dispositivi diversi pur mantenendo attiva la stessa sessione.
La stessa persona può connettersi una volta con un PC portatile e la volta seguente con un palmare e magari voler proseguire nel proprio lavoro da dove lo aveva lasciato

L’idea potrebbe essere di aver memorizzato il filmato in diversi formati con qualità diverse e quindi dimensioni diverse, in modo da poter sfruttare al meglio le potenzialità delle macchine di ogni utente. Quindi si cerca di creare un’infrastruttura che realizzi un adattamento dinamico alle necessità degli utenti.

1.2.5 - Architetture Proxy-based

Come visto la struttura più semplice per la realizzazione di applicazioni di streaming è quella tipica Client-Server in cui il Client, che risiede nella macchina utente, richiede un servizio al Server, che risiede in una macchina remota, il quale provvede all’erogazione del servizio, in questo caso la trasmissione di dati multimediali.
Abbiamo sottolineato come la conferma della ricezione di ogni dato sia di per se una strada difficilmente percorribile e che quindi la soluzione vada ricercata in un’altra direzione che consenta una minore comunicazione tra Client e Server e soprattutto che non carichi eccessivamente il Server di lavoro: le applicazioni multimediali in se richiedono spesso un uso massiccio della CPU ed è quindi consigliabile non caricare il Server di lavoro aggiuntivo.
Una soluzione ipotizzabile potrebbe essere la comunicazione da parte del Client del momento di buio, ovvero prima di perdere il segnale da un AP il Client dovrebbe comunicare la propria decisione al Server che interromperà la trasmissione dei dati per poi riprenderla da dove si era fermato una volta che il Client, connesso nuovamente, inoltri la richiesta di riprendere il servizio.
Questa soluzione molto intuitiva porta però con se una serie di problemi non irrilevanti: le schede wireless infatti non comunicano la decisione di cambiare AP ma semplicemente attuano il processo quando lo ritengono opportuno.
Inoltre nascerebbe il problema di calcolare l’anticipo con cui è necessario segnalare la richiesta di interruzione per far si che questa avvenga prima della reale perdita di segnale: nel tempo impiegato dalla segnalazione del Client a raggiungere il Server quest’ultimo avrà certamente spedito alcuni dati che devono avere tempo a sufficienza per raggiungere il Client se non vogliamo che vadano comunque persi.
Una tale operazione non consente di avere la certezza di non perdere pacchetti a meno di non prendere un ampio margine di anticipo che quindi avrebbe l’inevitabile conseguenza di allungare il periodo in cui il Client non riceve dati.
La possibilità che il Server sia in una località distante o che debba servire numerose richieste potrebbe inoltre aumentare il tempo di ripresa dello flusso video ed è quindi auspicabile un sistema che non carichi il Server, oltre della trasmissione del video, della gestione delle perdite dati del Client.

Un’altra possibile soluzione è quella di inserire un’ulteriore entità tra Client e Server chiamato Proxy[VoDSup].
Compito di questo Proxy è quello di fare da tramite tra le richieste del Client e i servizi del Server agendo da intermediario in modo da interfacciarsi col Server come un Client senza particolari richieste che non lo stream video, ed invece offrire al Client l’astrazione di un Server che possa seguirlo in maniera dedicata per venire incontro alle sue esigenze.

L’ipotesi è che Proxy e Server comunichino tra loro attraverso una rete fissa o comunque caratterizzata da un’elevata affidabilità e velocità di trasmissione.
Si da quindi per assodato, ed è un’ipotesi più che plausibile, che non ci siano perdite di informazione nel trasferimento tra Server e Proxy anche in assenza di una conferma costante dei dati ricevuti dal Proxy.
Le uniche comunicazioni che avverranno tra Proxy e Server saranno quindi la richiesta di servizio, il flusso dati, e la comunicazione di interruzione del servizio stesso.
Sarà quindi il Proxy ad occuparsi delle esigenze particolari del Client, cioè a realizzare localmente la ritrasmissione dei dati persi durante gli handoff

1. 3 - Simulazione

1.3.1 – Perché la Simulazione

Quando si cercano soluzioni a problemi relativi a questi contesti nascono esigenze specifiche che talvolta può essere difficile superare.
Abbiamo sottolineato la grande varietà di dispositivi che possono essere connessi alle reti ma più avanti, nel capitolo dedicato alle reti 802.11, sarà evidente come anche l’effettiva implementazioni di tali reti siano in realtà molto diverse l’una dall’altra: lo standard 802.11 infatti pone alcuni vincoli anche stringenti su certi aspetti della realizzazione del protocollo di comunicazione, ma lascia completa libertà per molti aspetti implementativi che verranno dettagliati nel capito dedicato a tale standard.
Per fare un esempio che chiarisca da subito la diversità a cui si deve fare fronte, diciamo da subito che non esiste una procedura standard per la gestione dell’handoff e che quindi ogni costruttore di schede può adottare la strategia che ritiene vincente.
Sviluppare una soluzione ad un problema che la tecnologia stessa non risolve significa dover aggiungere uno strato ulteriore che si faccia carico della gestione delle situazioni critiche e di conseguenza ha un costo che può essere non irrilevante.
La grande variabilità delle situazioni che si possono presentare rende evidente come qualsiasi soluzione pensabile non potrà essere al tempo stesso le migliore e la più economica per tutte le situazioni.
Inoltre le applicazioni che si prenderanno in considerazione hanno la caratteristica di dipendere fortemente dal tempo poiché real-time: questo comporta che la soluzione non può essere trovata solo a livello logico o di algoritmo, ovvero la soluzione non deve essere solo logicamente corretta ma deve rispettare determinati tempi se vuol essere efficace.
Purtroppo questi tempi possono essere molto diversi a seconda delle situazioni e molto spesso possono variare a seconda delle condizioni e fattori non facilmente prevedibili.
Oltre al numero delle prove, che dovrebbe essere molto elevato, si rendono necessarie anche numerose situazioni in cui provare la validità della soluzione cercata.
Si rende quindi necessario, durante lo sviluppo dell’infrastruttura, effettuare numerose prove per poi analizzare i risultati e stabilire se la strada intrapresa è quella giusta oppure va rivista.

In pratica potrebbero essere necessarie centinaia di prove fatte con decine di device differenti in diversi contesti e con diverse modalità.
Infine, è evidente l'impossibilità di avere a disposizione, a costi contenuti, una larga gamma di device e ancora più evidente è l'impossibilità di avere più reti Wireless.
Nasce quindi l'esigenza di creare un ambiente simulato in cui poter variare le condizioni secondo i nostri desideri per poter mettere alla prova le nostre soluzioni.
Questa simulazione sarà mirata a dare la possibilità di poter provare le soluzioni come se davvero si trovassero in un ambiente Wireless con le relative problematiche nascondendo all'applicazione la reale sede in cui sta girando.
Altra caratteristica desiderabile è quella di poter effettuare un grande numero di prove contemporaneamente in modo da risparmiare tempo.
Il simulatore dovrà inoltre consentire di svolgere più prove contemporaneamente e di poter provare le applicazioni con una grande varietà di situazioni.

1.3.2 – Emulazione

Lo scopo di questa tesi sarà quindi sviluppare un ambiente simulato che consenta lo sviluppo dell'applicazione senza tener conto della simulazione stessa: normalmente le applicazioni devono essere modificate per poter essere simulate e questo causa un lavoro extra per il progettista. Si vuole invece dare la possibilità di progettare l'applicazione come se dovesse essere lanciata si una device reale ma poi poterla provare in un ambiente simulato. La parola "Simulatore" diviene quindi non sufficiente per descrivere ciò che vogliamo realizzare: infatti le caratteristiche ottenute permetteranno di poter portare le applicazione sviluppate su questo ambiente direttamente ad interfacciarsi con device wireless reali senza necessità di modifiche. La capacità di offrire le stesse funzioni e le procedure di una vera device wireless tale che all'applicazione stessa sia trasparente il fatto che si trovi in un contesto wireless reale oppure in un ambiente simulato prende il nome di "emulazione".

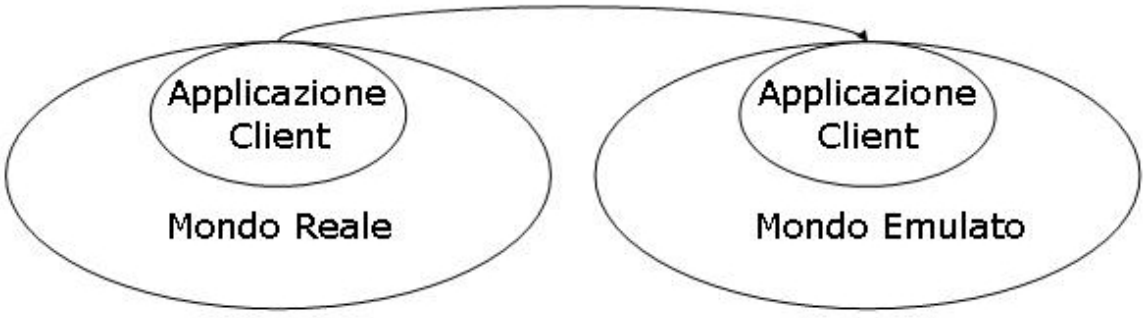


Fig 1.1 – Mondo reale e simulato

Capitolo 2 - Reti Wireless

Verso la fine degli anni ’90 il nostro stile di vita è notevolmente cambiato grazie all’introduzione di alcune tecnologie che permettono ad ognuno di noi di svincolarsi dalla necessità di essere fisicamente connessi ad una rete.
Ognuno di noi ha conosciuto i vantaggi della telefonia mobile che permette di essere rintracciati e di chiamare da praticamente qualsiasi locazione.
Questo ha portato (e sta portando) ad una necessità sempre maggiore di poter accedere a determinati servizi indipendentemente dal luogo in cui ci si trova.
Si pensi ad esempio ad uno sviluppatore in viaggio per lavoro: poter essere connesso alla rete Internet e quindi anche alla propria azienda da una biblioteca, una sala conferenza o persino da una coffe house costituisce un notevole vantaggio.
I motivi fondamentali che hanno portato allo sviluppo di reti Wireless sono quindi essenzialmente la mobilità, intesa come la possibilità di connettersi senza un cavo fisico, e la flessibilità della rete stessa.
Ogni rete Wireless si deve comunque appoggiare ad una rete fisica esistente e portare il segnale attraverso antenne garantendo così agli utenti riconosciuti i servizi che si avrebbero connessi ad una normale rete Ethernet.
La possibilità di connettere un numero variabile di utenti utilizzando la stessa struttura ha costituito una notevole spinta nella diffusione di questo tipo di rete: si pensi ad esempio ad un aeroporto o un campus universitario dove garantire ad ogni possibile utente un accesso Ethernet sarebbe dispendioso oltre che difficilmente realizzabile.
Le reti fisiche vanno anche incontro a quello che è il normale degrado di ogni componenti hardware.
Questo non significa che le reti wireless possano sostituire le reti fisiche in quanto, insieme agli indubbi vantaggi, portano con se anche una serie di limiti che vedremo in seguito.

2.1.1 – Standard 802.11

Quando si parla di reti Wireless si da quasi sempre per assodato che si stia parlando di una rete che si basa su componenti che seguono lo standard 802.11.
In realtà la definizione di rete Wireless non implica affatto uno standard: letteralmente infatti significa semplicemente “rete senza fili”
Lo standard 802.11 è semplicemente stato lo standard che più rapidamente si è imposto e seguito dai produttori di Hardware Wireless e quindi attualmente il più diffuso.
Quando descriveremo le caratteristiche tecniche delle Reti Wireless faremo quindi riferimento allo standard 802.11.
Come detto sopra qui parleremo di reti Wireless riferendoci allo standard 802.11 ma in realtà la definizione implica semplicemente una rete di macchine non connesse tramite cavo.
Un altro esempio potrebbe essere una rete che sfrutta le tecnologie Bluetooth.
Alla fine degli anni ’90 era però lo standard 802.11 ad essersi imposto e ben presto assunse altri nome come “Wireless Ethernet” .

Il marchio "*Wi-Fi*" (Wireless Fidelity) nacque quando la WECA (Wireless Ethernet Compatibility Alliance) introdusse il suo programma di certificazione: ogni strumenti che passi i test studiati per garantire il rispetto degli standard può utilizzare il marchio *Wi-Fi*.
In seguito usci anche il marchio *Wi-Fi5* per indicare quegli strumenti che utilizzando lo standard 802.11a che sfrutta una banda intorno ai 5 GHz.

2.2.2 – Il mezzo di trasmissione

Pur essendo senza fili anche queste reti utilizzano comunque un mezzo di trasmissione che in questo caso sono le onde elettromagnetiche.
Utilizzare le onde elettromagnetiche come mezzo per trasmettere dati significa anche utilizzare determinate frequenze che sono soggette ad un rigido controllo per evitare fenomeni di sovrapposizione.
Come già detto però le reti Wireless hanno anche i loro limiti intrinseci nell'essere sconnesse fisicamente e nell'usare onde elettromagnetiche come mezzo di trasmissione.
In primo luogo possiamo considerare come la velocità di trasmissione dei dati sia limitata dalla larghezza di banda disponibile.
E' possibile calcolare il limite superiore di tale velocità avendo a disposizione la larghezza di banda che, a meno che le autorità non decidano di aumentare, ha un limite.
Infatti attualmente gli strumenti Wireless utilizzano una banda "senza licenza" denominata S-Band ISM che ha disponibile la banda di frequenza dai 2.4 GHz ai 2.5 GHz.
Gli strumenti Wireless tendono ad essere più lenti di quelli connessi. Il mezzo stesso è meno affidabile rispetto alle reti Ethernet il che rende l'accettazione dei pacchetti in arrivo più pesante.
Le onde inoltre sono soggette ad altri tipi di problemi come ad esempio l'interruzione del collegamento, cammini multipli oppure "ombre".
Un problema critico per le reti Wireless è inoltre quello della sicurezza: i dati trasmessi infatti sono a disposizione di chiunque sia nel limite di portata dell'antenna.
Mentre nei collegamenti via cavo è possibile controllare l'instradamento del pacchetto e la protezione dello stesso, in una rete Wireless lo "sniffing" dei pacchetti (letteralmente "fiutare i pacchetti": consiste nel leggere anche i pacchetti non indirizzati alla propria macchina) è quantomeno facilitato in quanto le antenne trasmettono a chiunque riesca a ricevere il loro segnale.
Per le normali reti "con filo" i limiti dell'area di copertura sono chiari e sono determinati dal numero degli accessi che si predispone, al contrario una rete Wireless non può essere esattamente confinata: è probabile che se un ufficio o un negozio possiede una rete di questo tipo, il suo segnale possa essere captato anche al di fuori dell'edificio permettendo di entrare nella rete anche ad una persona parcheggiata in una zona vicina.

2.2.3 – Storia dell'802.11

Lo standard 802.11 uscì per la prima volta nel 1997 e fu pubblicato dall'IEEE.
Riporto in breve qui sotto la storia degli standard 802.11

802.11 – Nel 1997 esce il primo standard 802.11 caratterizzato da una velocità di 2Mbps e dalla frequenza di 2.4 GHz, troppo lenta per molte applicazioni.

802.11b – Nel 1999 la IEEE espande lo standard precedente arrivando a supportare una banda di 11 Mbps. Questo standard utilizza sempre la frequenza di 2.4 GHz poiché senza licenza e quindi non grava sul costo del prodotto. La vicinanza con altri strumenti che utilizzino la stessa banda può provocare interferenze ma può essere facilmente evitando disponendo accuratamente gli strumenti.
Pro: basso costo, migliore portata del segnale e miglior tolleranza agli ostacoli.
Contro: bassa velocità di trasferimento, limite basso al numero di utenti contemporaneamente collegati, possibili interferenze per l'utilizzo di una banda non regolamentata.

802.11a – Esce contemporaneamente alla 802.11b anche se poi non si diffuse quanto la precedente. Nasce come standard ad alto costo ma che garantisce migliori prestazioni.
La sua frequenza è 5GHz e supporta una velocità di 54Mbps. Utilizzando una frequenza più alta è meno tollerante ad ostacoli sul cammino ed inoltre è incompatibile con lo standard 802.11.
Pro: alta velocità di trasferimento, maggior numero di utenti contemporaneamente e utilizzo di una banda regolamentata (assenza di interferenze)
Contro: Alto costo e bassa portata del segnale che può essere facilmente ostacolato.

802.11g – Tra il 2002 ed il 2003 esce uno standard ulteriore che ha prende le migliori caratteristiche dei due precedenti: utilizza la frequenza 2.4 GHz garantendo così una migliore portata e una migliore tolleranza agli ostacoli. Inoltre supporta una velocità di 54 Mbps. Costa più dell'802.11b e mantiene gli stessi problemi dovuti all'utilizzo di una banda non regolamentata.

2.2 - Struttura delle reti 802.11

In questo capitolo farò una breve panoramica sulla struttura delle reti wireless basate sullo standard 802.11.
Una tale introduzione ha lo scopo di presentare alcuni meccanismi chiave su cui si base il trasferimento di dati, l'accettazione degli utenti e lo spostamento di questi.
Inoltre introdurrò alcune parole chiave che fanno parte del mondo wireless e che in seguito userò ripetutamente per spiegare lo scopo di questo lavoro.
E' utile inoltre spiegare a grandi linee ciò che queste reti hanno in comune con la tipologia di rete comunemente chiamata Ethernet e quali le differenze poiché alcune analogie possono fuorviare mentre altre sono state volute dagli ideatori dello standard 802.11 proprio per rendere, almeno superficialmente, simili i due standard.
Ad esempio, sia le schede Ethernet sia ogni scheda Wireless ottengono un indirizzo MAC di 48 bit che li rende, per molti aspetti, indistinguibili da chi li osserva esternamente mentre le principali differenze sono ovviamente legate al fatto che una scheda Wireless può muoversi e soprattutto il mezzo usato per la trasmissione.

2.2.1 - 802.11 e IEEE 802

Lo standard 802.11 fa parte della famiglia IEEE 802 che raccoglie una serie di specifiche per le tecnologie LAN (Local Area Network).
Qui di seguito una figura che riporta i componenti della famiglia 802 relazionata al modello OSI.

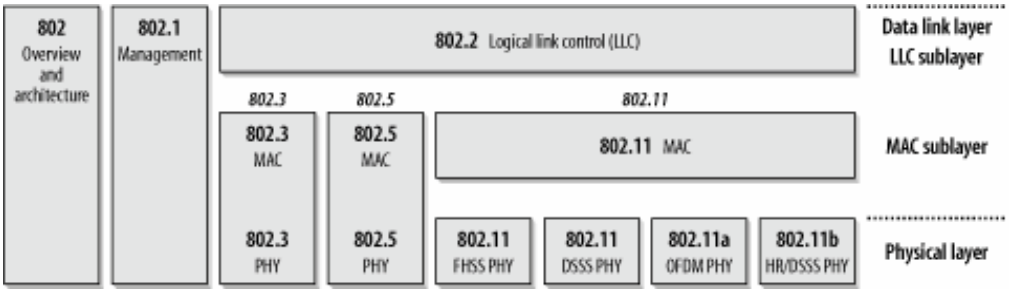


Fig 2.1 – Famiglia 802.11

Come si può vedere in figura 2.1 la famiglia 802 si colloca su i primi due livelli dello strato OSI
Ovvero il livello Fisico e quello di Data link.
MAC (Medium Access Controll) è un'insieme di regole che controllano l'accesso al mezzo di trasmissione, la trasmissione e la ricezione dei dati.
I dettagli implementativi su come effettivamente effettuare tali operazione sono però lasciati al livello Fisico poiché per ogni mezzo di trasmissione la tecnologia è diversa:

questo implica che laddove lo standard non specifica il costruttore è libero di procedere secondo le proprie scelte.
Ogni specifica è inoltre identificata da un secondo numero dopo il punto; ad esempio 802.3 è la specifica per il CSMA/CD (Carrier Sense Multiple Access/Collision Detection), ciò che è comunemente noto come Ethernet.
Come si può notare tutte le specifiche hanno uno strato in comune chiamato 802.2 (LLC) il che consente che tutte le specifiche offrano la stessa interfaccia al livello successivo e da questo si può capire come a tutti gli effetti una scheda Wireless 802.11 connessa ad una rete possa essere trattata come una qualsiasi scheda Ethernet.
Se si guardassero i dettagli dei vari protocolli ci si accorgerebbe in fretta di come lo standard 802.11 sia notevolmente più complesso ed arricchito di numerose regole: questo dipende unicamente dall'adozione delle onde elettromagnetiche come mezzo di trasmissione dati e dalla mobilità che deve essere garantita ad ogni utente.

2.2.2 - Elementi delle reti 802.11

I principali componenti di una rete Wireless sono effettivamente pochi:

Stazioni: tipicamente sono Computer Portatile o Palmari, ma in generale lo è qualsiasi macchina che sia dotata di una interfaccia Wireless. Non esiste un motivo per cui debbano essere macchine "portatili": si pensi ad esempio ad un ufficio in cui per comodità si è deciso di usare questo tipo di tecnologia per usare fili ed avere un sistema molto più flessibile: aggiungere o togliere un PC non comporta alcuna modifica alla rete esistente purché non superi il limite di utenti degli Access Point esistenti.

Access Point: fanno da ponte tra il mondo "senza fili" e quello connesso, traducendo i pacchetti che, viaggiando su mezzi diversi, non possono avere la stessa forma. In realtà hanno anche altre funzioni, ma questa rimane la più importante. Sono fisicamente identificabili come le antenne che trasmettono e ricevono il segnali degli utenti connessi.

Sistema di distribuzione: quando si connettono fra loro più Access Point per formare una rete con una copertura vasta, si necessita di un sistema di comunicazione tra gli Access Point stessi per tenere traccia degli spostamenti delle Stazioni mobili e per instradare i pacchetti provenienti dalle schede Wireless verso Access Point diversi o verso la rete Internet.

2.3 - Tipologie di Reti

2.3.1 - Reti IBSS o BSS ad hoc:

La più semplice delle reti è costituita da due stazioni dotate di interfaccia Wireless che decidono di comunicare tra loro. In generale sono composte da due o più stazioni che decidono di instaurare una rete momentanea fra di loro.
Immaginiamo, ad esempio, un gruppo di quattro sviluppatori che vogliano condividere le risorse per lavorare parallelamente ad un progetto o semplicemente mettere a disposizione degli altri alcune informazioni immagazzinate sulle loro macchine; il luogo in questo caso non ha molta importanza, può essere una biblioteca, una mensa o un parco.
Questa rete avrà ovviamente una copertura poco estesa e sarà di durata relativa alla necessità del momento.
Queste reti sono chiamate IBSS (Indipendent Basic Service Set) oppure reti ad hoc, proprio per il fatto che sono create per uno specifico motivo e momento.
Ovviamente, se tutte le stazioni vogliono comunicare l'una con l'altra, l'area di tale rete è limitata a quella che è la portata degli strumenti a disposizione.
Va inteso che la trasmissione del segnale avviene tridimensionalmente e quindi non occorre essere tutti allo stesso piano o livello; dato che comunque mediamente gli edifici sono costruiti a piani e che i muri costituiscono un notevole ostacolo alla propagazione delle onde, il termine Area viene comunemente accettato viste le applicazioni pratiche.

2.3.2 - Reti BSS o Infrastructured Basic Service Set

Sebbene il nome non vengono mai chiamate IBSS.
Sono reti in cui tutte le stazioni presenti, invece di comunicare direttamente tra loro, dialogano con un unico Access Point che collega tra di loro tutte le stazioni in un'unica rete facendo quindi da ponte tra una e l'altra. Per trasmettere un frame da una stazione ad un'altra sono necessari quindi due passi, ovvero prima trasmetterlo all'Access Point che poi provvederà a ritrasmetterlo.
Sebbene questo meccanismo sia indubbiamente più dispendioso per quanto riguarda l'occupazione di banda offre però altri vantaggi notevoli:

- L'area della BSS è definita dalla posizione dell'Access Point e la condizione sufficiente per poter accedere al suo servizio è trovarsi dentro la portata dello stesso: non ha quindi importanza la posizione relativa delle varie stazioni. Inoltre, se da un lato costa maggiormente sulla trasmissione di frame, elimina quella parte di comunicazione di una IBSS dovuta al fatto che le varie stazione devono mantenersi in comunicazione l'una con l'altra.
- La posizione dell'Access Point è in genere studiata per garantire una miglior gestione dei consumi: accorgendosi di quando una stazione entra in uno stato di basso consumo può memorizzare i frame a lei diretti in un buffer e trasmetterli solo quando la stazione stessa lo richiede. Quando una stazione entra in uno stato di basso

consumo disattiva il ricevitore per attivarlo unicamente per trasmettere e ricevere buffer di frame, in modo da utilizzare al meglio la carica residua della batteria.

In una rete di questo tipo ogni stazione, prima di usufruire del servizio, deve associarsi alla rete. E' la stazione a richiedere all'Access Pont di associarsi alla rete il quale può decidere di accettare la richiesta o rifiutarla a secondo della sua politica. La cosa più frequente è che sia richiesta un ID ed una Password di autenticazione.

Inoltre una stazione può essere connessa ad un solo Access Point per volta, cosa che, ovviamente, non vale per gli Access Point.

Lo standard non pone un limite al numero delle stazioni mobili che possono associarsi ad Access Point il che non vuole affatto dire che questo limite non ci sia: infatti maggiore è il numero di utenti minore sarà la banda riservata ad ogni utente, pertanto ogni Access Point ha un limite di utenti che può gestire contemporaneamente.

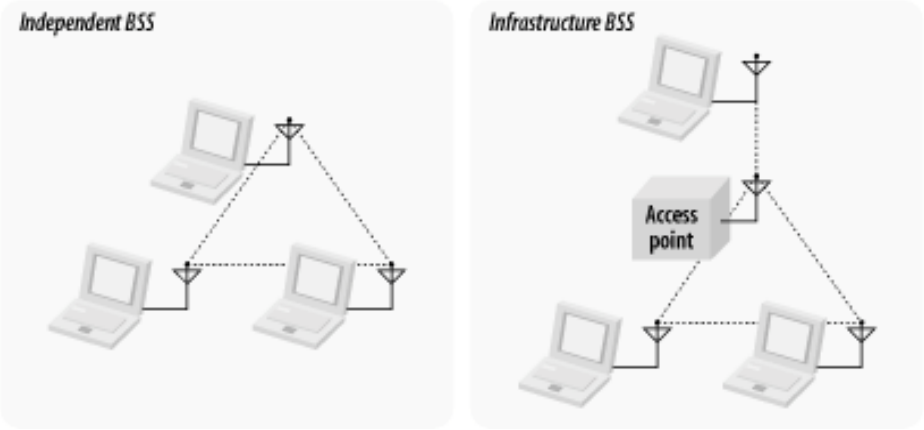


Fig 2.2 – Reti BSS

2.3.3 - ESS o Estended Service Set

Le reti BSS hanno il limite di poter coprire comunque un'area limitata, come un piccolo ufficio, una casa. Lo standard 802.11 permette di formare reti con una qualsivoglia area di copertura connettendo opportunamente tra di loro più BSS formando reti che vengono chiamate ESS (Extended Service Set).

L'unico requisito è che le BSS siano collegate tra loro: non è specificato in che modo ma sono specificati un numero di servizi minimi.

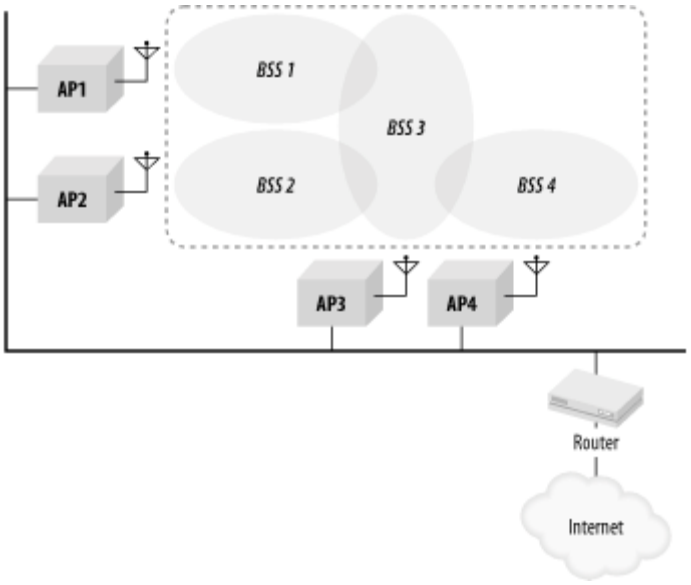


Fig 2.3 - ESS

Nella figura 2.3 possiamo osservare quattro Access Point collegati tra loro per formare una rete con una copertura più ampia di quanto potrebbe fare un unico Access Point.
In realtà non sono desiderabili spazi vuoti tra un'area e l'altra e si tende spesso a sovrapporre più aree per garantire ovunque il servizio.
Ogni stazione mobile all'interno di una ESS può comunicare con un'altra all'interno della stessa Rete: ovviamente, nel caso i due comunicanti siano su BSS diverse, l'Access Point che riceve il pacchetto dal trasmittente lo instraderà attraverso il Sistema di distribuzione verso un altro Access Point che provvederà a ritrasmetterlo
L'ESS rappresentano il più alto livello di astrazione di una rete offerta dallo Standard. 802.11 permette di poter comunicare con una Stazione mobile unicamente usando il suo MAC indipendentemente da dove questa si trovi realmente.

2.3.4 - Il Sistema di distribuzione

Dopo aver illustrato le varie tipologie di reti è bene capire su cosa debba necessariamente appoggiarsi una rete ESS, ovvero un sistema che connetta tra di loro i vari Access Point.
Come già detto questo sistema viene descritto unicamente nei termini dei servizi minimi che deve garantire e si lascia la massima libertà all'implementazione pratica dello stesso.
Il principale compito di questo sistema è portare ogni pacchetto all'Access Point al quale è collegata la Stazione destinataria del pacchetto stesso.
Deve quindi anche tenere traccia di dove siano realmente le stazioni o meglio sapere a quale Access Point sono legate.
Un esempio è quello di Router che si affaccia ad una rete Ethernet: egli conosce semplicemente l'indirizzo MAC del pacchetto e non può certo farlo pervenire all'Access Point corretto.
E' quindi chiaro che mediamente la rete Ethernet fa parte del sistema di distribuzione e viene utilizzata come mezzo di trasmissione, ma non è il sistema di distribuzione stesso.

La maggior parte degli Access Point operano come bridge ed hanno quindi almeno un'interfaccia Ethernet ed una Wireless per poter far da tramite tra i due mezzi di trasmissione.

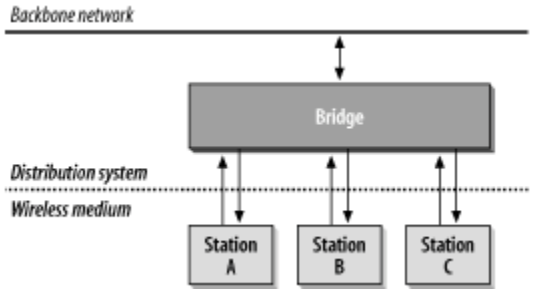


Fig 2.4 – Backbone network

Ogni stazione connessa ad un Access Point deve quindi utilizzare questo ponte per poter comunicare con un'altra stazione.
E' evidente che ogni Access Point dev'essere informato su ogni altri suoi pari in modo da poter mandare il frame al giusto destinatario.
Questo meccanismo fa sicuramente parte del sistema di distribuzione anche se non esistono metodi obbligatori per implementare questo servizio.
Nella figura si da per scontato che la rete di collegamento tra gli Access Point sia una rete fissa, ma nessuno vieta che anch'essa possa essere Wireless.

2.3.5 - Aree sovrapposte

Molto spesso le aree di due o più BSS si sovrappongono: questo è spesso voluto per garantire un più semplice passaggio da una all'altra durante il movimento o semplicemente per potenziare l'accesso in una singola area in sono previsti molti utenti ed un unico Access Point risulterebbe sovraccaricato.

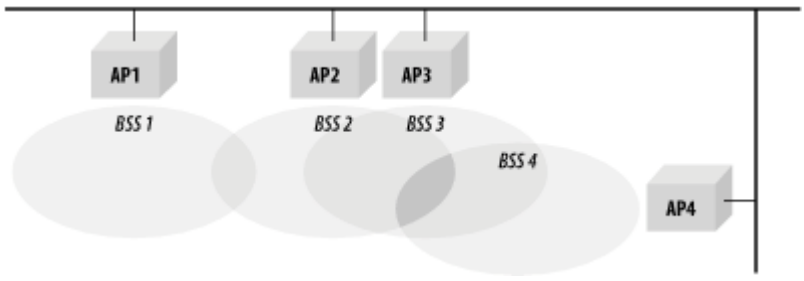


Fig 2.5 – Sovrapposizione di aree di AP

Un altro caso può essere invece quello in cui due o più stazioni decidano di formare una IBSS laddove esiste una BSS: in questo caso le aree delle due reti si sovrappongono ma esse sono slegate ed indipendenti.

Il protocollo che regola l'accesso al mezzo di trasmissione consente questo tipo di situazione permettendo quindi a più reti indipendenti di coesistere nella stessa area.

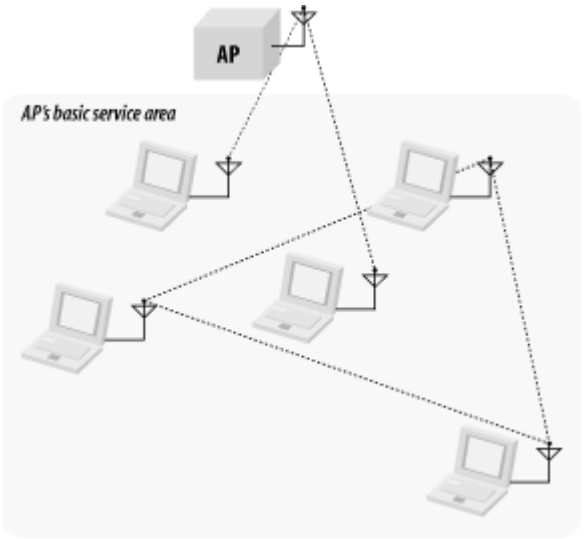


Fig 2.6 – Reti sovrapposte

2.4 - Mobilità nelle reti 802.11

Ora che abbiamo dato un'idea di come le reti Wireless sono composte e funzionano, passiamo a trattare un argomento centrale nei problemi legati a queste reti e che costituisce l'argomento centrale di questa tesi.
Finora abbiamo preso in considerazione a grandi linee il funzionamento in situazione "statiche" ovvero dove, una volta associata con un Access Point, la Stazione mobile non usciva dalla copertura di questo.
Ma uno dei grandi vantaggi delle reti Wireless è appunto la possibilità di muoversi portando con se il proprio portatile.
Ma cosa succede alla connessione o ad una applicazione durante questo processo che prende il nome di "*roaming*"?

2.4.1 – Roaming

Esistono due tipi di roaming:
Seamless roaming : letteralmente "migrazione senza giunture" e sta ad indicare che la connessione con la rete non viene mai persa durante lo spostamento. Il caso esemplare di questo tipo di roaming è quello della telefonia mobile: mentre telefoniamo infatti possiamo anche attraversare più celle e non accorgerci mai del passaggio dall'una all'altra. La tipologia stessa dell'applicazione rende necessario il fatto che la connessione non venga mai persa.

Nomadic roaming : l'esempio più calzante di questo tipo di roaming è proprio quello di una rete 802.11: quando ci spostiamo da una locazione ad un'altra con il nostro portatile generalmente noi interrompiamo il nostro lavoro per poi riprenderlo una volta giunti alla nuova locazione. Mentre questo avviene la nostra scheda Wireless si sconnette dal primo Access Point per riconnettersi a quello che copre la nuova area. Questo tipo di roaming è quindi caratterizzato dal fatto che non è previsto che l'utente utilizzi la rete durante lo spostamento.

Ma cosa accade alle nostre applicazioni durante questo spostamento?
Per rispondere a questa domanda è necessario prendere in considerazione vari aspetti quali:

- Di che natura è il roaming?
- Quanto dura il roaming?
- Il roaming avviene entro una singola sottorete o attraverso più sottoreti?
- Di che natura è l'applicazione? Connection-Oriented o Connectionless?

2.4.2 - Natura del Roaming

La natura del Roaming 802.11 può essere identificata come "break before make", ovvero interrompe la connessione prima di procedere al cambio di Access Point: questa soluzione a prima vista sembra sconveniente in quanto ammette la possibilità di perdere alcuni pacchetti poiché esiste un intervallo di tempo in cui non siamo connessi alla rete.
Ogni interfaccia wireless dovrebbe essere capace di essere connessa contemporaneamente a due Access Point aumentando la complessità delle schede stesse rendendole inevitabilmente più costose.
Inoltre se si tentasse di legarsi al nuovo Access Point prima di disconnettersi dal vecchio si avrebbero notevoli difficoltà di implementazione.
Per prima cosa esisterebbe la possibilità di essere legati simultaneamente a due Access Point con lo stesso MAC-address. Questo potrebbe dare origine a "loop" e bisognerebbe dotare le sottoreti di opportuni algoritmi per riconoscere queste situazioni appesantendo notevolmente il protocollo.

2.4.3 - Natura dell'applicazione

Un'applicazione Connection-Orientend che utilizzi connessioni di tipo TCP è sicuramente più resistente a questo tipo di operazioni in quanto il protocollo stesso incapsula una misura di sicurezza contro la perdita di pacchetti tramite un sistema di acknoledgement e ritrasmissione dei pacchetti perduti.
Altre applicazioni invece sono basate sul protocollo UDP che risulta meno pesante per la rete ed è preferito in applicazioni come VoIP (Voice over IP) oppure applicazioni video.

Il protocollo UDP non prevede un sistema di ritrasmissione dei pacchetti persi e quindi le applicazioni che lo utilizzano sono sicuramente più soggette al roaming tra un Access Point ed un altro.

2.4.4 - Durata del Roaming

La durata del processo di Roaming è influenzata da vari fattori quali:
- Il tempo di ricerca
- Il tempo di autenticazione 802.11
- Il tempo di connessione 802.11
- Il tempo di autenticazione 802.1X

Per ora tralascio maggiori dettagli riguardo le modalità di roaming perché esulano dallo scopo di questo capitolo ma vale la pena dare alcune informazioni che possono essere chiarificatrici per le situazioni che verranno esposte in seguito.
Va subito detto che lo standard 802.11 non regola l'handoff. Lascia cioè libertà ad ogni costruttore di implementare la propria politica. In ogni algoritmo sicuramente avranno importanza alcuni fattori come la potenza del segnale o la perdita di segnali di sincronizzazione, ma se ci spostiamo ad un livello superiore tutto questo si traduce, mediamente, in un maggiore o minore tempo di disconnessione percepito a livello applicativo.

2.5 – Politiche di Roaming

Passiamo ora a vedere le principali strategie che vengono seguite quando si effettua il roaming da un AP all'altro. Un handoff è in realtà un serie di eventi e decisioni che hanno una precisa sequenza: la scheda wireless deve infatti decidere quando migrare, dove migrare, iniziare il processo di roaming ed in seguito le applicazioni dovranno riprendere la loro sessione di lavoro. Come abbiamo detto 802.11 non regola le strategie di handoff e quindi ogni costuttore ha la propria.
Potenza del segnale, perdita dei segnali di controllo sono cose che vengono tenute in considerazione in ogni strategia. Inoltre ogni costruttore dovrà cercare un equilibrio tra durata degli handoff e stabilità delle applicazioni dell'utente poichè strategie troppo reattive possono causare un numero eccessivo di handoff mentre strategie meno reattive disconnessioni molto lunghe.

2.5.1 – Decidere dove migrare

Possiamo in generale identificare due tipi di strategie principali[80211WL]:
- Preemptive AP discovery
- Reactive AP discovery

Ognuna di queste due strategie può utilizzare due tipi di meccanismi:
Active scanning: in questo caso la scheda wireless manda alcuni segnali di "probe" e rimane in ascolto da parte degli AP. Si tratta quindi di un tipo di ricerca attiva in cui si interrogano direttamente tutti gli AP in ascolto su tutti i possibili canali, attendo poi i messaggi di risposta.
Passive scanning:con questo meccanismo la scheda rimane in ascolto dei messaggi di "segnalazione" spediti autonomamente dagli AP. Ad ogni intervallo prefissato di tempo cambia canale di ascolto ma non effettua mai una richiesta specifica.

Usando il meccanismo active scanning la scheda deve rimanere in attesa di una risposta su ogni canale per un tempo prefissato che mediamente varia tra i 10 e i 20ms.
Invece usando il meccanismo passive scanning la scheda deve rimanere in ascolto dei messaggi di segnalazione da parte degli AP che mediamente vengono spediti ogni 10 ms.
Ovviamente questo comporta da una parte una maggiore velocità per il primo meccanismo riducendo la possibilità di non vedere un AP in quanto le richieste sono effettuate dalla scheda stessa, dall'altra parte la tecnica passiva è sicuramente più lenta ed esiste la possibilità di non vedere qualche AP (cioè perdere i suoi messaggi di segnalazione) ma non c'è la necessità di effettuare richieste attivamente.

Preemptive AP discovery:con questa politica la scheda wireless decide dove migrare "prima" di effettuare la disconnessione. Questo processo permette di avere handoff più breve e quindi ha un minor impatto sulle applicazioni.
Questo tipo di discovery ha comunque le sue controindicazioni: per consentire questo tipo di operazione la scheda dell'utente deve effettuare l'operazione di scanning anche durante il normale svolgimento della connessione. Per fare questo deve cambiare canale di ascolto per effettuare la ricerca della connessione. Ricordo che una scheda wireless può ascoltare su un solo canale per volta.
Durante questa operazione la scheda non può quindi rimanere in ascolto dei dati in arrivo dall'AP corrente con la conseguente possibilità di perderli e di chiedere la ritrasmissione.
Simmetricamente le applicazioni sulla macchina utente non possono trasmettere informazioni verso l'AP.
Si può sfruttare l'opzione del power-safe per eliminare questo problema: molte schede possono entrare in una modalità a basso consumo per preservare le batterie dei dispositivi portatili. In questo stato la scheda si disattiva fintanto che non ha un numero prefissato di dati da spedire. Comunicando questa sua modalità all'AP anch'esso bufferizza i dati destinati all'utente. Quando l'utente si risveglia trasmette i suoi dati e riceve quelli conservati per lui dall'AP. La soluzione consiste nell'effettuare lo scanning prima di disattivare la ricezione ma dopo aver comunicato il cambio di modalità all'AP in modo tale che la scheda e l'AP conservino i dati da inviare e nel frattempo la scheda possa effettuare lo scanning.
Questo tipo di roaming è supportato fintanto che l'utente si muove ad una velocità limitata e la scheda ha il tempo di effettuare le sue operazioni. Se l'utente di muove troppo velocemente c'è il rischio di una frequenza di handoff elevata con un conseguente degrado per le prestazioni delle applicazioni

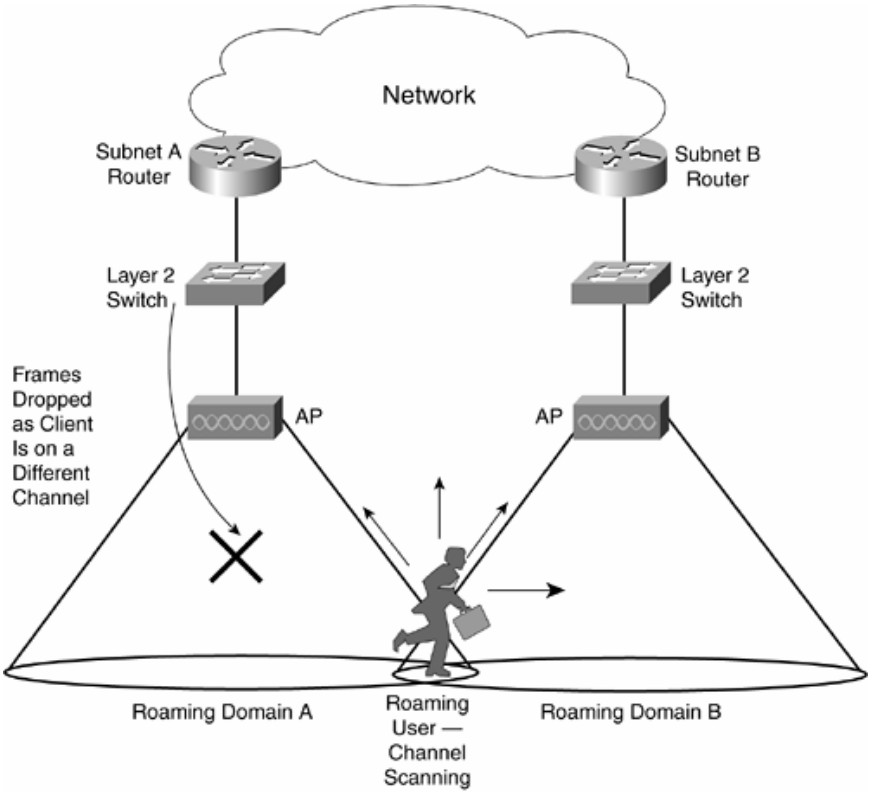


Fig 2.7 – Preemtive AP discovery

Reactive AP discovery:l'altra opzione è che la ricerca dell'AP avvenga dopo la decisione di migrare. In questo caso il protocollo è simile a quello con cui la scheda chiede l'associazione tranne per il fatto che il messaggio sarà di ri-associazione.
Facilmente intuibile è che in questo modo l'utente non ha l'overhead introdotto dalla ricerca preventiva ma paga questo vantaggio avendo durate di disconnessione maggiori.

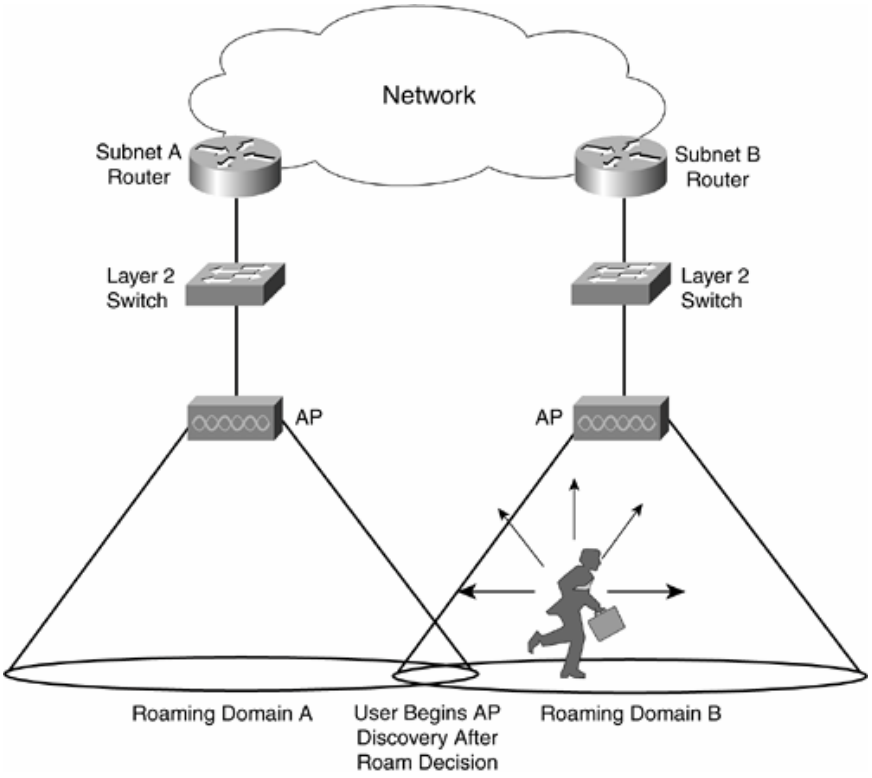


Fig 2.8 – Reactive AP discovery

2.5.2 - Strategie di Handoff

Come abbiamo detto lo standard 802.11 lascia al costruttore la libertà di gestire l'handoff cosa che se da una parte aumenta la varietà delle soluzioni adottate, dall'altra rende più difficile costruire una soluzione sui comportamenti delle schede.
Non potendo avere una conoscenza a priori dell'esatto meccanismo della scheda si è cercato di individuare almeno le linee guida che vengono seguite nella decisione del cambio AP.
E' chiaro che alcuni fattori, come la potenza del segnale, sono comuni per tutte le politiche di gestione dell'handoff, ma resta il fatto che noi non possiamo essere a conoscenza di queste politiche ma dobbiamo basarci sul comportamento osservabile a livello più alto.
Si sono così individuate due principali di categorie o meglio politiche di gestione, una proattiva e l'altra reattiva.

Reattive (Reactive) sono le strategie che cercano di minimizzare il numero degli handoff conservando l'associazione con l'attuale AP fintanto che il segnale non degrada. Così facendo la procedura di handoff parte solamente quando il segnale con l'AP viene perso dando luogo a tempi più lunghi in quanto ad la ricerca di nuovi AP e la richiesta di connessione fanno parte della procedura.

Proattive (Proactive) sono politiche che tendono ad avviare la procedura di handoff prima della perdita del segnale e cioè quando si accorgono dell'esistenza di un segnale più forte rispetto a quello dell'AP a cui la scheda è associata. Questo produce sicuramente un maggior numero di handoff inutili ma sono in genere più veloci in quanto la ricerca di nuovi AP è effettuata prima della perdita del segnale.

RSSI: (Received Signal Strength Indication) è la sigla che useremo per indicare la forza di un segnale proveniente da un AP.

2.5.3 - Hard Proactive e Soft Proactive

Tra le schede che adottano un strategia proattiva si possono distinguere due categorie:

Hard Proactive: sono strategie in cui l'handoff viene effettuato non appena l'RSSI di un AP supera quello del segnale dell'AP a cui siamo connessi di una certa soglia che chiameremo HHT (Hysteresis Handoff Threshhold): questo per evitare che l'utente possa cambiare continuamente AP se staziona in una zona in cui due segnali sono praticamente uguali.

Soft Proactive: sono strategie meno "proattive" rispetto alle precedenti in quanto includono, oltre all'HHT, un'altra condizione, e cioè che l'RSSI del segnale del AP attuale sia inferiore ad una certa soglia FHT (Fixed Handoff Threshold)

2.7 – Standard 802.11 correlati

Abbiamo visto come in realtà lo standard 802.11 si composto da una famiglia molto più ampia di protocolli che estendono le funzionalità base degli standard principali. L'elenco completo di questi standard già sviluppati o in corso di sviluppo fino ad oggi sono elencati in una tabella in Appendice A. Gli standard vengono differenziati tra di loro tramite una lettere posta dopo la sigla "802.11" come abbiamo già visto e non tutti sono interessanti ai fini di questa tesi. Alcuni di esse trattano della sicurezza nella tecnologia Wi-Fi, altri dell'aumento di prestazioni.
Alcuni di essi invece non possono essere ignorati poiché interessano l'ambito in cui si colloca questa tesi che è principalmente l'handoff durante la migrazione tra due AP.

2.7.1 – 802.11F

Questo protocollo [80211FS] ha lo scopo di formalizzare una comunicazione tra AP per permettere di risolvere alcune situazioni all'occorrenza di alcuni eventi come le associazioni da parte di utenti e la ri-associazione da parti di alcuni di essi: è il caso in cui un utente, già connesso, perde la connessione con un AP e la riottiene con uno successivo.
Questo standard riguarda quindi il progetto di un Inter-Access Point Protocol (IAPP).
Il risultato è un protocollo usato dall'entità che gestisce un AP di poter comunicare con gli altri AP per segnalare alcuni eventi.
E' chiaro quindi che è necessario avere alle spalle degli AP un'infrastruttura che si consapevole della presenza di tali AP e che riesca a controllarli.

Fanno parte di una tale infrastruttura i server di autenticazione (RADIUS:Remote Authentication Dial In User Service): il compito principale di questi server è tenere traccia di tutti gli AP presenti collegando i loro BSSID(Basic Service Set Identifiers) con i rispettivi IP e di fornire a tutti gli AP delle chiavi per la trasmissione criptata delle informazioni.
Ovviamente è previsto un protocollo di comunicazione anche tra AP ed i server di autenticazione che però riguarda strettamente la gestione degli AP.
Due eventi vengono principalmente gestiti tramite questo protocollo e sono l'associazione di una nuova stazione mobile e la ri-associazione di una stazione mobile.
All'atto dell'associazione, ovvero quando una stazione mobile richiede la connessione alla rete, semplicemente l'AP registra il nuovo utente comunicando il MAC della suo dispositivo wireless in modo da permettere al sistema di tenere traccia del nuovo utente.
Quando un utente si muove cambiando AP egli chiede di essere ri-associato. In questo caso il nuovo AP può chiedere al sistema a quale AP questo utente fosse precedentemente associato. A questo punto può iniziare uno scambio di informazioni tra i due AP che ha lo scopo di velocizzare la procedura di ri-associazione. Questo da l'opportunità di richiedere il "contesto" della stazione mobile direttamente al vecchio AP. Un esempio delle informazioni che si possono ottenere sono le disposizione della stazione mobile sulla sicurezza.
Inoltre lo standard prevede anche di effettuare un "caching context" proattivo, ovvero la possibilità di trasmettere il contesto di una stazione mobile agli AP in cui potrebbe migrare prima che l'handoff avvenga.
Entrare nei dettagli tecnici di questo protocollo esula dai nostri attuali scopi ma è evidente che il tema trattato potrebbe essere simile in quando si introduce una infrastruttura che tenga traccia degli AP e degli utenti collegati ed alcune strategie che tendono a minimizzare la durata degli handoff.

2.7.2 - 802.11r

Questo standard non è, alla data di ultima revisione di questa tesi, ancora stato pubblicato. Sono però note le intenzioni e che ad un gruppo di ricercatori è stato assegnato il compito di sviluppare questo nuovo elemento della famiglia 802.11.
Lo scopo di questo standard è ridurre fortemente la durata degli handoff fino a 50 ms.
Uno standard di questo tipo cambierebbe sensibilmente il panorama da cui questa tesi è partita poiché i dati persi durante un handoff si ridurrebbero in maniera consistente.

Capitolo 3 – Analisi del Progetto

In questo capitolo illustrerò la struttura del progetto senza entrare nel dettaglio della implementazione ma dando un'idea precisa dell'organizzazione dello stesso e delle funzionalità delle varie parti. Verrà quindi analizzata la soluzione proposta ai problemi presentati nei capitoli precedenti dando una descrizione delle varie parti e dell'insieme che prenda in considerazione solo le caratteristiche desiderate e tralasci, per il momento, i dettagli implementativi che saranno esposti nei capitoli seguenti.
Il capitolo sarà diviso in due parti come due sono le parti principali in cui è diviso l'elaborato.
Nella prima parte descriverò l'applicazione lato Client e l'infrastruttura pensati per risolvere il problema dell'handoff e quindi descriverò l'idea che sta alla base delle soluzione e poi le varie parti.
Nella seconda invece illustrerò la struttura del Simulatore mettendo in evidenza la progettazione logica per evidenziare come si è pensato di dare risposta alle varie esigenze e come questo diventi emulatore per le applicazioni lato Client.

3.1 – Struttura della soluzione applicativa

Il problema a cui ci proponiamo di dare soluzione è quello che riguarda l'handoff in reti 802.11 durante un'applicazione multimediale come Video on Demand o Live.
La struttura tipica di questo tipo di applicazioni è quella di avere da una parte un Server che possiede il filmato (nel caso di VoD) o che sta riprendendo l'evento (Live) che invia un flusso di informazioni multimediali verso un Client che lo ha richiesto. Caratteristica da tenere in considerazione è che, video o audio che sia, la riproduzione avviene in tempo reale e cioè mentre si ricevono le informazioni. Questo comporta, come già visto, di poter avere perdite di informazioni durante un handoff.

3.1.1 – Architettura Proxy-based

Per risolvere il problema dell'handoff si è scelto di interporre una terza entità tra il Server ed il Client, ed a tale entità prende il nome di Proxy. Questa scelta è stata fatta tenendo conto delle particolari esigenze del Client che variano a secondo delle situazioni e del momento. Dovendo interagire in tempo reale con un Client molto eterogenei si è ritenuto opportuno decentrare la gestione delle richieste del Client stesso in modo da poter agire tempestivamente ed adattarsi dinamicamente alla variazioni dell'utente.
La struttura che assume infine lo scenario comprende quindi un Server che soddisfa la richiesta di VoD dialogando con il Proxy e che ignora la vera natura del Client; il Proxy, agendo da serbatoio di passaggio per i dati, provvede in seguito a spedire i frame video al Client e si prepara a soddisfare le sue richieste.

La comunicazione tra Proxy e Server si limiterà quindi ad una transizioni iniziale in cui il Proxy richiede il filmato ed un successivo stream di dati dal Server al Proxy.
Tra il Proxy ed il Client sono invece previste, oltre alla transizione iniziale e alla trasmissione del filmato, l'invio di richieste da parte del Client.

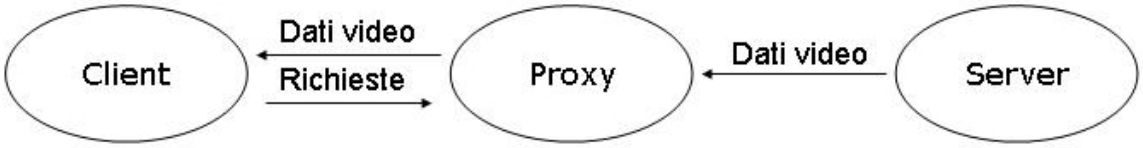


Fig 3.1 – Struttura Proxy-based

Non ha importanza dove sia si trovi il Server in questo modo, la gestione delle problematiche del Client avverrà sempre vicino ad esso e ciò garantisce risposte immediate, un minore overhead per il server e di poter prevedere Proxy dalle caratteristiche variabili che si adattino alle variazioni dinamiche del Client. Altra cosa che il Proxy può compiere è adattare i dati multimediali al Client: il Server potrebbe avere il file in un unico formato che il Client potrebbe non essere capace di visualizzare o che potrebbe essere troppo oneroso per le capacità limitate della macchina che ospita l'applicazione Client. Il proxy potrebbe farsi carico della conversione dei dati che invece costerebbe notevolmente in termini di utilizzo della CPU da parte del Server.

3.2.2 – Gestione dell'handoff

Quando avviene un handoff il Client perderà certamente una determinata quantità di dati che quindi potrà richiedere una volta riassociato ad un AP. Sarà compito del Proxy soddisfare questa richiesta conservando opportunamente i dati da poter reinviare. Il Client manderà quindi una richiesta non appena terminato l'handoff al Proxy che provvederà a ritrasmettere la quantità di dati richiesta. Il Client, dal suo lato, dovrà riuscire a gestire l'applicazione anche mentre non è connesso. L'insieme delle due procedure renderà invisibile all'utente la migrazione tra un AP ed un altro. Ovviamente il Proxy dovrà adattarsi alle caratteristiche sia del Client che del filmato da trasmettere: il Client può essere caratterizzato da handoff più o meno lunghi ed il filmato richiedere più o meno memoria a seconda del suo formato di compressione. Il fatto che il Client possa richiedere la ritrasmissione dei dati non deve influenzare la comunicazione tra Proxy e Server ovvero il Proxy non farà mai richieste di questo tipo al Server altrimenti perderebbe di validità la struttura adottata che ha il compito di decentralizzare la soluzione a questo tipo di situazioni.

3.2.3 – Gestione della memoria

Una soluzione di questo tipo risolve il problema della perdita di dati durante un handoff ma ha un costo aggiuntivo rispetto ad una architettura Client - Server in termini di occupazione di memoria. Un Client non è costantemente prossimo ad un handoff ed anzi, molto probabilmente passa la maggior parte del tempo fermo, specialmente se guarda un filmato. Il fatto che le tecnologie wireless offrano una maggiore mobilità non deve farci considerare ogni utente come un'entità in continuo movimento. Durante tutto il tempo in cui il Client non si muove questa occupazione di memoria aggiuntiva è inutile. Si vorrebbe quindi un modo di poter allocare la memoria aggiuntiva solo quando ce n'è effettivamente necessità, minimizzando l'occupazione dei Proxy durante i periodi in cui il Client è fermo.
Il Client potrebbe informare il Proxy della sua situazione, ovvero potrebbe comunicare se la sua posizione è al momento stabile o se si sta muovendo. Chiaramente questo prevede una maggior comunicazione tra i due e lavoro aggiuntivo per il Client.
Il sistema che ospita i Proxy non è ovviamente dedicato alla sola gestione di questi ma regola tutte le operazioni che avvengono all'interno della rete e tutte le altre richieste degli utenti diverse da quelle multimediali. Questo sistema potrebbe decidere che in un particolare momento le applicazioni multimediali sono le più importanti, come ad esempio una conferenza, oppure decidere che altre applicazioni hanno la precedenza e quindi limitare le risorse a questo tipo di applicazioni. Da qui sorge la necessità di un supervisore dei Proxy creati che tenga conto della memoria attualmente occupata e, ad esempio, possa decidere di non accettare più richieste fintanto che non sarà disponibile altra memoria per queste applicazioni.

3.2.4 – Previsione

Inserire un livello tra Client e Server significa ovviamente una maggiore occupazione di memoria e se pensiamo ad un Proxy (non inteso come il programma che serve un Client ma come quello che ospita l'intero sistema, ovvero che raccoglie le richieste di tutti i Client) che raccolga le richieste di decine di Client allora comincia a diventare un dispendio di risorse non accettabile nel caso queste risorse siano impegnate a prescindere.
La necessità di implementare un sistema che potesse "prevedere" un possibile spostamento nasce dalla constatazione che per provvedere alla situazione creata da un handoff sono necessarie più risorse rispetto ad una situazione stazionaria: prendendo in esame il sistema di stream video è evidente che non solo è necessario un sistema in grado di riconoscere i dati persi ed un meccanismo che li ritrasmetta, ma anche acquisire più risorse per poter conservare le informazioni. Dato che mediamente si lavora con un portatile o con un palmare stando fermi, o comunque muovendosi a velocità limitate, sarebbe utile avere un'informazione sul prossimo futuro dell'utente, ovvero sapere se è probabile che si sposti o meno.
La parola "previsione" può fuorviare poiché non si tratta esattamente di una predizione sullo spostamento, quanto più un sistema che, raccogliendo i dati ed osservandoli cronologicamente ci da una stima della probabilità esistente di un cambio di AP.

3.3 – Applicazione Server

3.3.1 - Funzionalità

Il Server, nello scenario descritto, deve soddisfare delle funzionalità piuttosto limitate in quanto la sua principale caratteristica dev'essere quella di poter soddisfare più richieste contemporanee possibili e di assicurare un flusso continuo di informazione.
Possiamo immaginare il Server come un grosso archivio di filmati che può risiedere praticamente ovunque, nel senso che la sua locazione precisa non ha particolare influenza ai fini delle prestazioni.
Facciamo l'assunzione che il Server sia raggiungibile attraverso la rete Internet e che tra il sistema che ospita il Proxy ed il Server sia garantita una velocità di trasmissione sufficientemente elevata.
Anche il Server può prevedere un sistema di riconoscimento perché ad esempio esterno al Sistema o perché comunque si vogliono controllare gli accessi a tali risorse.
La richiesta di servizio prevede una breve parte iniziale di scambio informazioni come ad esempio l'identità del richiedente ed il nome del filmato richiesto. Inoltre si ipotizza che durante una prima fase di negoziazione il server passi anche alcune informazioni sul filmato stesso per consentire al ricevente di predisporsi alla ricezione del filmato. Un'informazione essenziale, ad esempio, è la velocità con cui verranno trasmessi i dati. Ricordo che, nel caso di applicazioni real-time come la riproduzione di video, questo dato è equivalente alla frequenza del filmato stesso.
Si è detto che la distanza del Server non ha molta influenza sulle prestazioni dell'applicazione: in realtà i tempi di trasferimento dati possono aumentare, ma questo potrà avere ripercussioni solo sulla parte iniziale della comunicazione. Una volta partito il flusso di dati questo sarà comunque continuo anche se partito con qualche istante di ritardo.
La caratteristica più importante del Server è quella di garantire una trasmissione dati precisa e costante, ovvero che rispetti le tempistiche dettate dal filmato. Un Server che trasmetta i dati troppo velocemente oppure troppo lentamente può portare a diversi problemi nella visualizzazione del filmato. Il Server, prima di spedire il filmato, devo comunque elaborare l'informazione in modo da adattarle alle esigenze del protocollo di trasmissione dati. Tale elaborazione dati può risultare pesante se pensata moltiplicata per il numero di richieste pendenti sul Server.

3.4 – Applicazione Proxy

3.4.1 - Funzionalità generali

Per Proxy qui intenderemo quel modulo che si occuperà della gestione del filmato richiesto dal Client. In generale possiamo vedere il Proxy come l'entità che si occupa dell'utente durante la propria sessione di connessione e che quindi potrebbe predisporre una serie di servizi allo stesso utente.
In questo caso ci occuperemo nello specifico di quella parte predisposta appunto alla gestione dei servizi multimediali, ovvero che accoglie la richiesta di filmato da parte del Client e che provvede alla comunicazione con il Server.

3.4.2 - Il riconoscimento

Come abbiamo detto il Proxy dev'essere in qualche modo in grado di identificare chi richiede un servizio e decidere di conseguenza se servirlo o meno.
Il Proxy dovrà quindi prevedere un modulo incaricato di ricevere richieste da chiunque, vagliarle, e nel caso predisporre un'entità in grado di soddisfare le richieste dell'utente.
Possiamo pensare che, nel momento in cui l'utente viene accolto nel sistema globale, gli venga assegnato un ID di sessione che renderà più semplice identificarlo ed in modo da non dover ripetere tale procedura ogni volta ch'egli richieda un servizio.
Ogni volta che una richiesta viene accettata viene creato un modulo riservato per l'utente che avrà il compito di gestire la sua richiesta dall'inizio alla fine. Dato che questo modulo richiederà una certa quantità di risorse è opportuno controllare che il Client non faccia più richieste contemporanee: a questo scopo, e per poter riconoscere il Client, il modulo incaricato del riconoscimento dovrà servirsi di tabelle opportune da poter consultare per accettare le richieste e per poter registrare gli utenti già serviti.
Identificheremo il modulo incaricato della gestione del riconoscimento come Proxy Manager mentre le entità create appositamente alla richiesta dei Client saranno indicate semplicemente Proxy.
Compito del Proxy Manager sarà anche quello di tenere traccia dei Proxy creati e di monitorare il loro stato così che non ci siano mai Proxy che occupano risorse inutilmente.

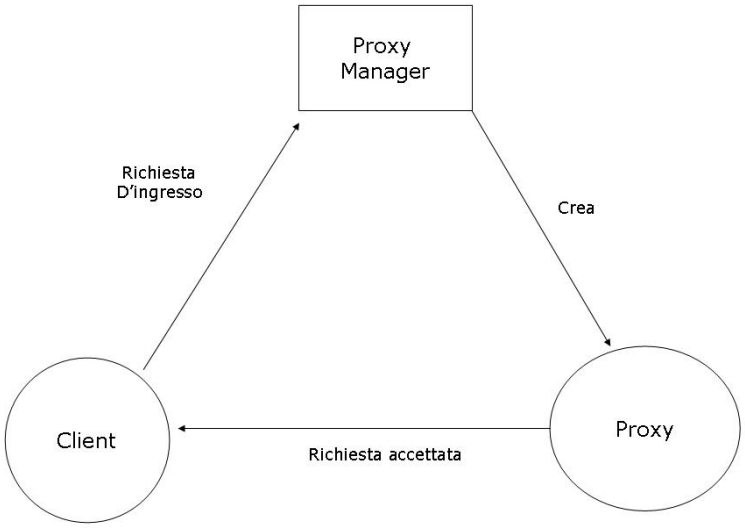


Fig 3.2 – Struttura Proxy

3.4.3 - Gestione dello stream video

Il modulo Proxy creato appositamente per il Client a questo punto apre un canale privato di comunicazione attraverso il quale accetterà le richieste da parte del Client. Ovviamente il Client dovrà essere messo al corrente di questo canale di comunicazione che utilizzerà in seguito per inoltrare richieste. La prima richiesta del Client sarà, limitatamente al caso trattato, quella richiedere la visualizzazione di un filmato, non ha importanza in questa sede se sia VoD oppure una ripresa Live. Il Proxy dovrà quindi comunicare con il Server e fare la stessa richiesta predisponendosi per la ricezione dei dati multimediali. Compito del Proxy non è unicamente passare i dati al Client ma conservarli per poter permettere di soddisfare le richieste di ritrasmissione da parte del Client. Il Proxy ha quindi la funzione di fornire una riserva dati aggiuntiva messa a disposizione per il Client. Quando il Client richiede la ritrasmissione dati il Proxy dovrà essere in grado di riconoscere i dati richiesti e provvedere alla trasmissione senza perdere lui stesso dati dal Server durante questa operazione.

3.4.4 – Adattamento dinamico al Client

I Proxy dovranno avere una struttura adattabile alle esigenze del Client e non una struttura fissa identica per tutti. Questo perché, come detto, le macchine che ospitano l'applicazione Client possono essere estremamente diverse tra loro. In prima battuta quindi il Proxy dovrà comprendere le caratteristiche del Client che sta servendo e configurare le sue parti nel modo migliore.
Le richieste del Client possono inoltre mutare durante l'esecuzione e quindi il Proxy dovrà prevedere di poter mutare lui stesso a seconda dello stato del Client. Inoltre non si può dare per ipotesi la conoscenza precisa di ogni Client: il Proxy dovrà essere predisposto per ricevere un qualunque Client ed eventualmente imparare le sue esigenze durante l'erogazione del servizio. Inoltre dovrà essere in grado di adattarsi alle previsioni che il Client potrà inviargli.

3.5 – Applicazione Client

3.5.1 – Gestione del contesto

Abbiamo sottolineato l'importanza di avere un Proxy che si adatti ad ogni tipo di Client in modo da poter gestire al meglio le sue risorse e le richieste del Client stesso.
L'applicazione Client non vuole essere così adattiva ma certamente dovrà favorire il compito del Proxy e quindi comunicargli quante più informazioni utili possibile sulle proprie esigenze. Quando l'applicazione Client viene fatta partire potrebbe comunicare con il sistema ospite per acquisire alcune informazioni sulle risorse della macchina: alcune di queste informazioni potrebbero essere i codec video presenti, la memoria disponibile, il processore ed altri dati tecnici.
La cosa più importante sarà ovviamente potersi interfacciare con la scheda wireless per poter acquisire informazioni su di essa: l'informazione principale sarà sicuramente la marca ed il modello della scheda poiché, se conosciuta, permetterebbe al Proxy un servizio più personalizzato. Inoltre la scheda wireless rappresenta l'unico "occhio" che ha la macchina sul mondo esterno per poter comprendere la situazione dell'utente, se ad esempio è fermo o si sta spostando.
E' chiaro quindi che una parte del Client dovrà occuparsi del dialogo con il dispositivo wireless montato sulla macchina.

3.5.2 - Client come Player

La prima funzione che deve possedere il Client è quella di visualizzatore di filmati in quanto questa è la richiesta espressa dall'utente. Un normale riproduttore di filmati ha a disposizione l'intero file, da leggere e quindi rendere a video, all'interno del sistema e quindi non deve curarsi di avere abbastanza dati per poter svolgere la propria funzione, ne di riceverli da un processo esterno.
Nel nostro caso invece il filmato si trova in una locazione esterna ed è trasmesso attraverso un flusso continuo di dati verso il Client.
Una opzione potrebbe essere quella di ricevere interamente il filmato e poi riprodurlo, ma questo costa tempo, risorse sulla macchina dell'utente e potrebbe non essere possibile (Es: una conferenza in diretta).
Il Client deve quindi riprodurre il filmato mentre lo riceve, eliminando i dati giù utilizzati per lasciare spazio a quelli entranti.
L'applicazione dovrà quindi essere predisposta alla ricezioni dei dati attraverso la scheda wireless e quindi prevedere un modulo che svolga questo compito.
Un altro modulo dell'applicazione sarà invece incaricato di leggere i dati ricevuti e riprodurli a video: dovrò pertanto riconoscere il formato dei dati passati per utilizzare il tipo di codifica giusta.
Come abbiamo sottolineato le schede Wireless prendono la decisione di cambiare AP in maniera autonoma e non comunicano questa decisione; prevedono unicamente di poter essere interrogate sulla situazione attuale.

Il Client quindi non può accorgersi di essere stato non connesso par alcuni secondi ma può, ad esempio, notare l'interruzione del flusso dati.
Il modulo incaricato della visualizzazione del filmato dovrò quindi essere in grado di riconoscere la perdita di elementi e di formulare una richiesta per il Proxy in modo da ottenere i dati perduti.
Durante il periodo in cui l'utente non è connesso il Client dovrà comunque continuare a riprodurre il filmato per non dare all'utente l'impressione di interruzione di servizio.

3.5.3 – Installazione del Client

Osservando lo scenario prima di questo progetto è chiaro che il problema della perdita di informazione durante un handoff non ha attualmente soluzione.
Le applicazioni che sono state pensate al fine di sopperire alle carenze delle reti Wireless non sono quindi di dominio pubblico e pertanto non possiamo pensare che siano già presenti nel sistema dell'utente.
L'utente potrebbe non avere affatto un'applicazione in grado di ricevere dati attraverso una scheda di rete per poi visualizzarle.
E' chiaro inoltre che l'applicazione Client dovrà comunicare con il Proxy e che quindi dovrà possedere una certa conoscenza intrinseca di come dialogare con esso.
Gli strumenti moderni ci danno la possibilità di trasportare codice[MoAgMi] [CodMob], ovvero di trasmettere al Client l'applicazione adatta a svolgere la funzione richiesta dal Client.
In questo caso Java ci offre l'enorme vantaggio di poter scrivere il codice una volta sola per tutti gli utenti per le sue proprietà di portabilità e mette a disposizione strumenti per "Muovere" il codice da una macchina all'altra. E' quindi ipotizzabile che la macchina utente possa effettuare il dowload del codice direttamente dal Proxy.

3.5.4 – Previsione

Un'altra parte fondamentale che il Client dovrà svolgere sarà quella di prevedere gli spostamenti dell'utente: per questo ovviamente dovrà sfruttare i dati provenienti dalla scheda ed elaborarli. In seguito dovrà comunicare la sua previsione al Proxy in modo che esso si possa adattare alla sue esigenze temporanee.

3.6 – Il Simulatore

Il Simulatore ha lo scopo di permettere lo sviluppo delle applicazioni senza avere la necessità di provarle in un ambiente wireless reale e su macchine con dispositivi wireless.
Il suo compito principale sarò quindi di riprodurre il più fedelmente possibile la realtà che circonda un dispositivo wireless in modo da poter effettuare prove significative.
L'altro scopo è quello di permettere la scrittura del codice senza prendere in considerazione l'ambiente simulato, ovvero permettere al progettista di scrivere il codice applicativo un'unica volta. Lo scopo finale è quindi fornire all'applicazione Client un

comportamento verosimile dell'ambiente che la ospita, dotato di tutte le funzionalità di una macchina vera e di tutte le difficoltà di un ambiente wireless.

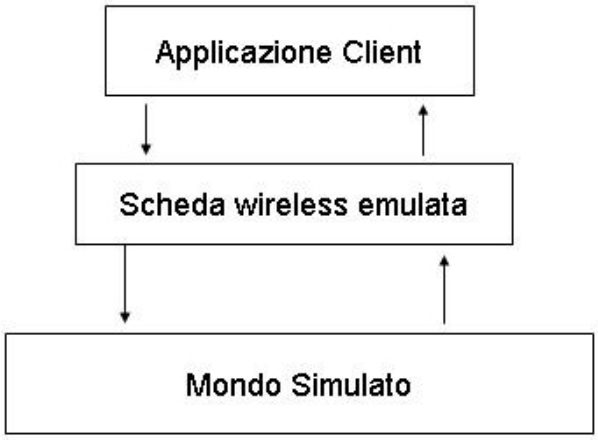


Fig 3.3 – Emulazione scheda wireless

3.6.1 – Architettura

La struttura del simulatore è stata pensata riprendendola direttamente dal contesto reale. L'unico strumento con cui dialoga l'applicazione per ottenere informazioni è la scheda wireless mentre quest'ultima ha come unico sensore la ricezione dei segnali da parte degli AP. La simulazione è quindi stata divisa in due strati: il primo, quello inferiore, dovrà occuparsi della simulazione dei segnali che vengono emessi dagli AP e che possono essere captati dalle schede wireless. Il secondo strato, che si appoggia sul primo, dovrà invece poter leggere i dati offerti dal mondo sottostante ed offrire all'applicazione lo stesso comportamento che avrebbe una scheda nel mondo reale.

3.6.2 – Il mondo simulato

Questo modulo ha il compito di presentare le informazioni che una scheda normalmente riceverebbe se si trovasse in un contesto wireless vero e proprio. La comunicazione tra AP e schede wireless prevede uno scambio di messaggi che seguono diversi protocolli, ad esempio l'associazioni iniziale, oppure i messaggi di segnalazione (chiamati beacon) da parte degli AP stessi. Inoltre le informazioni vengono passate seguendo un certo protocollo che è stato pensato sulle proprietà del mezzo di trasmissione. Non si deve pensare che sia necessario replicare tutti questi funzionamenti perché eccessivamente complesso e soprattutto non richiesto dagli scopi che ci siamo prefissi. Questo strato è giustamente uno strato di pura simulazione e non ha lo scopo di rappresentare fedelmente tutto ciò che avviene realmente ma solo di offrire una certa interfaccia al livello soprastante. Siamo cioè consci che lo strato che emula le schede wireless ci richiederà solo un certo tipo di informazioni.

La simulazione dovrà quindi prevedere un meccanismo che simuli la visione del mondo da parte della scheda che dipenda dalla sua posizioni attuale all'interno di un contesto.
Tale contesto non dovrà essere fisso per tutte le simulazione ma dovrà essere in grado di variare per presentare una pluralità di situazioni per poter sperimentare al megli ola validità dell'applicazione.

3.6.3 – Emulazione delle schede

Questo strato dell'applicazione ha il compito di dialogare con l'applicazione Client. Il suo scopo è offrire un'interfaccia che non sia diversa da quella di un contesto reale e presentare gli stessi comportamenti a livello di connessione di una scheda: quando una vera scheda wireless decide di effettuare l'handoff la macchina ospite perde momentaneamente la connessione con la conseguente perdita dei dati ed anche questo dovrà avvenire all'interno della simulazione. Questo livello ha inoltre il compito di leggere informazioni dallo strato sottostante e, in base a quelle, prendere le decisione come farebbe una normale scheda.
Le schede wireless prevedono di poter essere interrogate e di fornire determinate informazioni come ad esempio gli AP visibili e relative potenze ed il MAC dell'AP a cui è attualmente associata.
Tutte queste informazioni dovranno essere disponibili all'applicazione come se davvero dialogasse con una device wireless.
Inoltre questo livello non dovrà avere un comportamento fisso ma dovrà prevedere di emulare un diverso numero di schede come varie sono le schede con cui l'applicazione dovrà funzionare.
E' quindi essenziale che siano disponibili informazioni sulla scheda stessa e che i comportamenti delle varie schede emulate dovranno essere diversi l'uno dall'altra riproducendo il più possibile il comportamento delle reali schede.
Le strategie di ricerca, i parametri che ogni scheda prende in considerazione per prendere le proprie decisioni costituiscono il punto di forza della varie schede e possono quindi essere complessi e spesso non noti. Non è possibile quindi pensare di implementare davvero il meccanismo decisionale di ogni scheda. Non dobbiamo dimenticarci però che il nostro scopo è quello di fornire all'applicazione la stessa situazione e non di emulare davvero una scheda wireless: questo significa, ad esempio, che non è utile tentare di riprodurre lo scambio iniziale che porta l'associazione ad un AP poiché l'applicazione saprà unicamente se il dispositivo è connesso o meno.
L'applicazione non conosce cosa succede durante la gestione di un handoff, l'unico effetto a lei visibile è un periodo di buio in cui riceve dati. E' quindi essenziale guardare la scheda come la potrebbe vedere un'applicazione Client e riprodurre quella visuale senza scendere nel dettaglio laddove diventa superfluo per i nostri scopi.

Capitolo 4 – Progetto ed implementazione dell'infrastruttura

In questo capitolo entreremo nel dettaglio del progetto, mettendo alla luce alcuni dettagli ed alcune scelte specifiche introdotte dagli strumenti usati, come ad esempio il linguaggio Java.
Prima di iniziare la spiegazione dell'implementazione delle varie parti scenderò anche nel dettaglio di alcuni strumenti utilizzati come il protocollo RTP. Noi non gestiremo tale protocollo ma useremo delle librerie preesisteni scritte in Java facenti parte di un progetto della Sun chiamato JMF.
In seguito daremo una descrizione specifica di alcuni dettagli implementativi che possono aiutare alla comprensione del codice.
Escluderemo per leggibilità e per interesse il codice stesso.

4.1 – Real Transfer Protocol (RTP)

E' utile, prima di proseguire nella presentazione del lavoro, esporre qualche concetto chiave legato al Real Transport Protocol poiché offre alcuni strumenti per meglio comprendere diverse scelte operate all'interno della tesi.
Inoltre probabilmente allarga gli orizzonti sulle applicazioni che utilizzano questo tipo di protocollo e del perché si è deciso di utilizzarlo nelle nostre applicazioni.
Premetto che la presentazione di tale protocollo non vuole essere esaustiva ma solo fornire alcune principali indicazioni per migliorare la comprensione del panorama in cui si inserisce il mio lavoro, tralasciando invece alcuni dettagli che, sebbene importanti per il protocollo stesso, rischiano solo di creare più confusione per un lettore non esperto.

4.1.1 - Presentazione

Il protocollo RTP [RFC1889] fornisce una serie di funzioni per il trasporto punto a punto adatte ad applicazioni che trasmettano dati real-time come video, audio, o simulazioni, in maniera unicast (i.e. trasmissione di tipo uno a uno) o multicast (i.e. trasmissione di tipo uno a molti facenti parti di un gruppo).
RTP non riserva risorse e non garantisce alcuno qualità di servizio.
Accanto all'RTP è stato ideato l'RTCP (Real Time Controll Protocol) che permette un monitoraggio del trasferimento dati.
Entrambi i protocolli sono stati pensati per essere indipendenti dai livelli di trasporto sottostanti
Le applicazioni generalmente usano RTP sopra il protocollo UDP utilizzando le caratteristiche proprie di tale protocollo per soddisfare le condizioni richieste anche se RTP non richiede tale protocollo ma è stato pensato per poter essere implementato su qualsiasi livello di trasporto sottostante.
RTP non ha predisposto alcun meccanismo per assicurare tempi di trasferimento dati ma affida queste caratteristiche al livello di trasporto sottostante. Inoltre non garantisce

l’arrivo dei pacchetti ne l’arrivo nell’ordine di spedizione e non si basa su livelli sottostanti che garantiscano tali caratteristiche.
I numeri di sequenza introdotti in RTP consentono di ricostruire l’ordine di spedizione ed anche di computare pacchetti con un preciso ordine anche non in sequenza.

4.1.2 - Definizioni

Mixer: il mixer è un sistema intermedio che riceve dati da una o più fonti, può cambiare il formato di tali dati e disporli nella maniera desiderata, e poi li rispedisce. Può anche cambiare la temporizzazione dei pacchetti. Il sorgente dei pacchetti RTP diventa a questo punto il Mixer stesso.

Translator: è un sistema intermedio che inoltra i pacchetti RTP senza cambiare il loro SSRC. Il loro compito potrebbe essere quello di cambiare la codifica dei dati, ad esempio video.

RTP Payload: i dati trasportato dal pacchetto RTP, che può essere audio, o dati video compressi. La trattazione di questi formati esula dallo scopo di questo paragrafo.

RTP Packet: è un pacchetto di dati contenente un header RTP e l’RTP Payload. Alcuni livelli sottostanti potrebbero richiedere un metodo per incapsulare il pacchetto RTP. Tipicamente ogni pacchetto del livello di trasporto sottostante contiene un pacchetto RTP ma se il metodo di incapsulamento lo permette possono essere più d’uno.

RTCP Packet: è un pacchetto di controllo che consiste in un header fisso seguito da elementi strutturati che dipendono dal tipo di pacchetto RTCP. Generalmente più pacchetti RTCP sono incapsulati in un pacchetto del livello sottostante e spediti insieme.

Port (porta): E’ l’astrazione che i protocolli di comunicazione usano per distinguere diversi destinatari all’interno di una singola macchina ospite. TCP/IP identifica tali porte con un intero positivo. Anche se mediamente corrispondono alle porte UDP il protocollo non prevede un protocollo specifico sottostante.

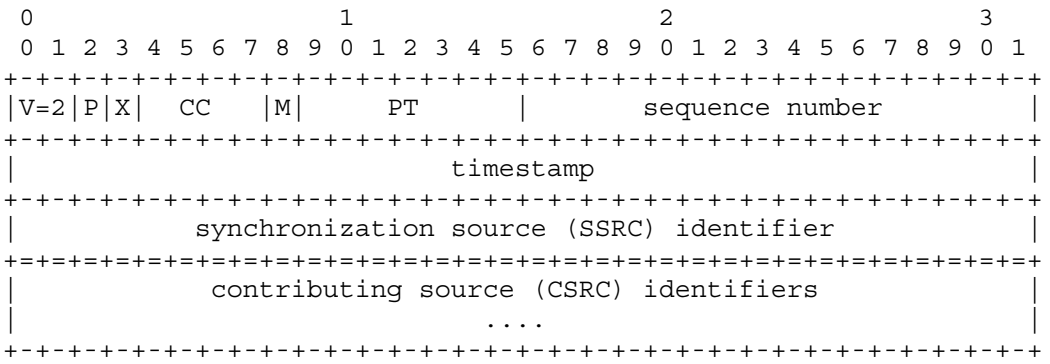
Transport address: la combinazione di un indirizzo di rete ed di una porta. Tipicamente un indirizzo IP ed una porta.

RTP Session: l’associazione tra un gruppo di partecipanti che comunicano utilizzando RTP. Per ogni partecipante la Sessione è definita da una sorgente ed un destinatario in caso unicast mentre nel caso multicast il destinatario è un Transport address di gruppo.
Ogni partecipante mette a disposizione una coppia di transport address.Tale coppia è generalmente formata da un indirizzo di rete e due porte, una per l’RTP ed una per l’RTCP. Nel caso di dati multimediali ogni canale necessita della propria Sessione RTP

Synchronization source (SSRC): la sorgente di uno stream di pacchetti RTP identificata da un numero a 32 bit chiamato SSRC ed inserito in ogni header RTP. Il ricevente può così raggruppare i pacchetti di una stessa sorgente utilizzando tale parametro. Un esempio di sorgente può essere il segnale di un microfono e di una videocamera. Questo numero è scelto casualmente in modo da essere unico all'interno di una sessione.

Non ho incluso tutte le definizioni possibili poiché alcune riguardano dettagli che difficilmente verranno messi in evidenza. Queste definizioni dovrebbero comunque essere sufficienti a garantire la comprensione sostanziale di questo protocollo.

RTP Header



Questo è il formato dei campi fissi in un header RTP.
Particolarmente importanti per i nostri scopi sono:

PT (Payload type): identifica il tipo di Payload trasportato dal pacchetto in modo da predisporre un sistema di lettura adeguato (se è audio o video, ad esempio)

Timestamp: un numero a 32 bit che identifica il pacchetto nella sequenza ma soprattutto da un'indicazione "temporale" della collocazione del pacchetto in quanto dovrebbe rappresentare l'istante di campionamento e, quindi, di riproduzione. Il valore iniziale è generato casualmente. A differenza del numero di sequenza, che è diverso per ogni pacchetto, il timestamp può essere uguale per più pacchetti successivi: si pensi ad un frame video scomposto in più pacchetti che hanno quindi la caratteristica di rappresentare lo stesso istante di campionamento.

Esempio

Un gruppo di persone, magari dirigenti di una impresa, decidono di discutere una decisione comunicando attraverso il servizio multicast di IP. Utilizzando alcuni meccanismi di allocazione ottengono un indirizzo multicast ed una coppia di porte, una per i dati audio e l'altra per il sistema di controllo. Ogni partecipante spedisce pacchetti audio tramite RTP in cui ogni volta specifica il tipo di dato (Payload Type); questo

consente di cambiare tipo di compressione per adattarsi, ad esempio, ad un nuovo utente inseritosi che magari non dispone di banda sufficiente.
Ogni pacchetto contiene inoltre il numero di sequenza ed il timestamp in modo da poter riordinare pacchetti arrivati non ordinatamente
Questa ricostruzione avviene indipendentemente per ogni sorgente audio durante la sessione che, ricordo, può essere identificata dal SSRC

E' spesso utile conoscere i partecipanti ad una conferenza di questo tipo, per cui periodicamente ogni partecipante spedisce un report RTCP utilizzando il multicast e quindi indirizzandolo a tutti i presenti. Questo consente una conoscenza su chi attualmente sta partecipando alla conferenza.

4.2 – Applicazioni multimediali

Questo breve capitolo introdurrà alcuni concetti legati alle applicazioni multimediali per fornire una migliore comprensione della struttura dell'applicazione.

4.2.1 – Riproduzione

Quando parliamo di applicazioni multimediali in generale parliamo di poter riprodurre filmati oppure file audio, per guardare un film oppure ascoltare musica o un telefonata.
E' conoscenza diffusa che i filmati siano ad esempio costituiti da una sequenza di immagini fisse che, mostrate in veloce successione, danno al nostro occhio l'idea di movimento. In generale ognuna di queste immagini viene chiamata "frame" ed occupano una certa dimensione in memoria. L'occhio umano ad esempio ha bisogno che l'immagine cambi almeno 25 volte al secondo per avere l'impressione di un'immagine fluida e non a scatti. Queste applicazioni dovranno quindi cambiare immagine con tempi prefissati e scadenze stingenti per ottenere un buon risultato. E' inoltre noto che i file video hanno dimensioni difficilmente trascurabili. Per venire incontro a questo problema si sono inventate delle codifiche che riducano le dimensioni di tali filmati. Questo significa che ogni frame video non viene conservato in un filmato direttamente riproducibile ma che deve prima essere decodificato e poi messo a video. Questo fa si che le applicazioni multimediali possano risultare pesanti anche come occupazione della CPU ovvero che richiedano molti calcoli per decodificare un frame video. I frame video non hanno tutti la stessa dimensione, a secondo del tipo di codifica, ed anzi spesso possono essere diversi tra loro. Discorsi analoghi possono essere fatti per i file audio.

4.2.2 – Stream video

Quando un filmato non risiede nella macchina su cui gira l'applicazione i dati devono essere trasmessi. E' possibile che il file non possa essere spedito per intero prima di essere visualizzato ma che debba essere visualizzato durante la ricezione dei dati. Anche se l'applicazione ragiona a frame non è detto che così sia per chi si occupa della

spedizione dei pacchetti stessi. Questo significa che spesso un frame non viene spedito interamente ma diviso in più parti più piccole e poi spedito. Questo perché generalmente un frame è più grande dell'unità di trasmissione dei dati attraverso internet.
Quando parliamo di pacchetti persi durante un handoff ci riferiamo alle unità di spedizione utilizzate dai livelli più bassi e non ai frame. Perdere un pacchetto non vuol quindi dire perdere l'intero frame ma ad esempio solo una parte. Da una parte questo comporta il vantaggio che la perdita di più pacchetti possa significa la perdita di un solo frame e quindi un minor impatto sull'utente. Dall'altra parte invece è possibile che la perdita di un solo pacchetto dati vanifichi la ricezione del resto del frame.
L'applicazione dovrà quindi gestire in qualche modo questa differenza, se ad esempio richiedere solo frame interi oppure solamente i pacchetti persi.
Le tempistiche stringenti per questo tipo di applicazioni fanno si che la trasmissione dati debba prediligere la velocità al controllo sui dati spediti: è spesso meglio perdere un frame ma non fermare il video, ottenendo come unico effetto un piccolo scatto nel filmato, che non fermare la riproduzione per attendere la ritrasmissione dei dati.

4.3 – Client

Vediamo ora più da vicino i meccanismi che sono stati studiati per risolvere il problema dell'handoff.
Per ora ci occuperemo del problema considerando unicamente il lato del Client e tralasciando momentaneamente il lato Proxy ed il lato Server in quanto le cose possono essere viste in maniera non correlata.
Per ora assumiamo unicamente che sia la richiesta che la comunicazione avvenga Connectionless, ossia appoggiandosi su RTP over UDP.

4.3.1 – Struttura del Client

Il programma lato Client si dovrà occupare di:
- Inviare al Server la richiesta del filmato.
- Predisporre le porte per la ricezione dei pacchetti
- Decodificare e rendere a video i frame
- Interrogare la scheda Wireless in modo da poter inviare al Proxy la possibilità di handoff

Appare quindi chiaro che in realtà il programma Client sarà diviso in più parti, alcune indipendenti tra loro ed unite da un unico componente che chiameremo Controller. Altre supposizioni che facciamo è che al momento del lancio del programma l'utente abbia già effettuato la connessione alla rete; qualora questo non fosse vero il programma semplicemente si metterà in attesa.
Come ho già sottolineato, si fa l'ipotesi di essere accolti da un sistema che ci riconosce e che vuole garantirci alcuni servizi a livello di sessione. Questo implica che anche se l' indirizzo IP dell'utente dovesse cambiare tra un handoff egli dovrebbe essere riconosciuto da altri identificativi e la sua sessione di lavoro non dovrebbe essere

interrotta: in pratica, una volta che l'utente effettua il suo primo ingresso in rete viene riconosciuto e la sua sessione mantenuta attiva finché questi non decide di terminare la sessione o si allontana dall'area coperta dalla rete stessa.
L'ingresso alla rete può avere più modalità ma molto semplicemente assumiamo che ad ogni Log ci venga assegnato un numero identificativo che useremo per tutta la sessione per venire identificati.
Il client è in realtà piuttosto semplice ed è costituito da tre parti principali:

- Player video
- Client Stub
- Controller

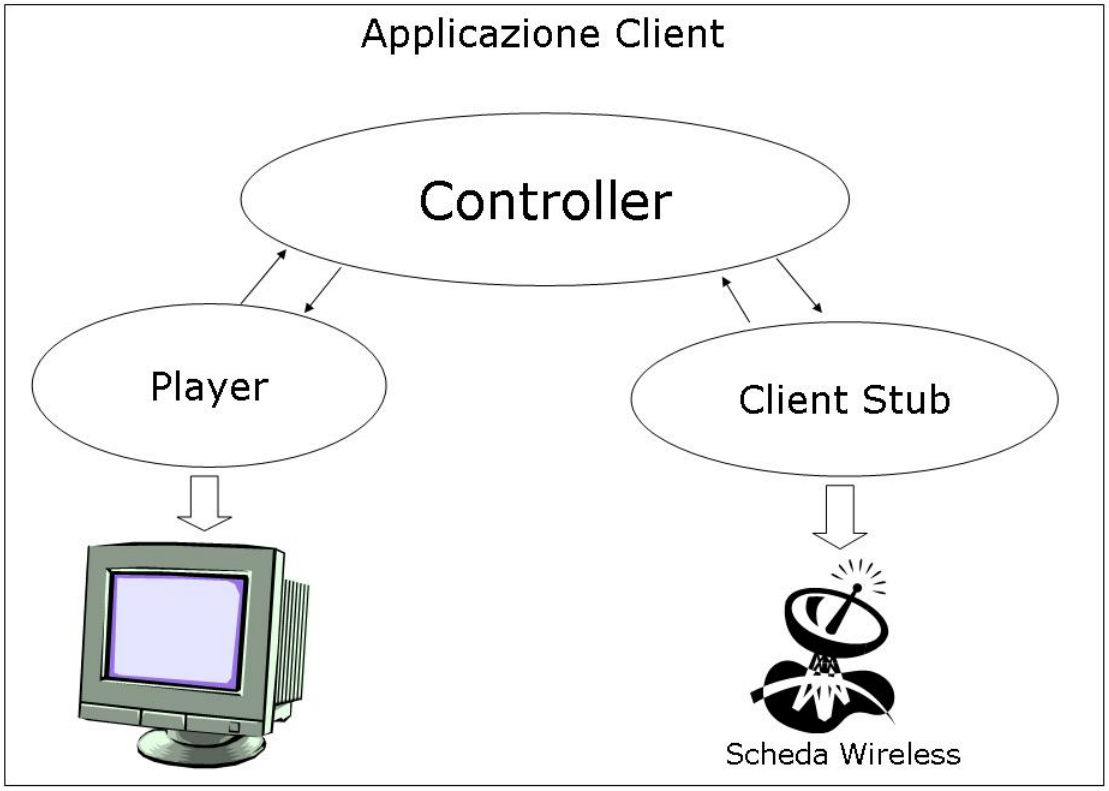


Fig 4.1 – Struttura Client

Il Client dovrà inoltre dialogare con il Proxy per ottenere il servizio ed effettuare le sue richieste. Illustriamo brevemente i principali passaggi della comunicazione tra Client e Proxy:

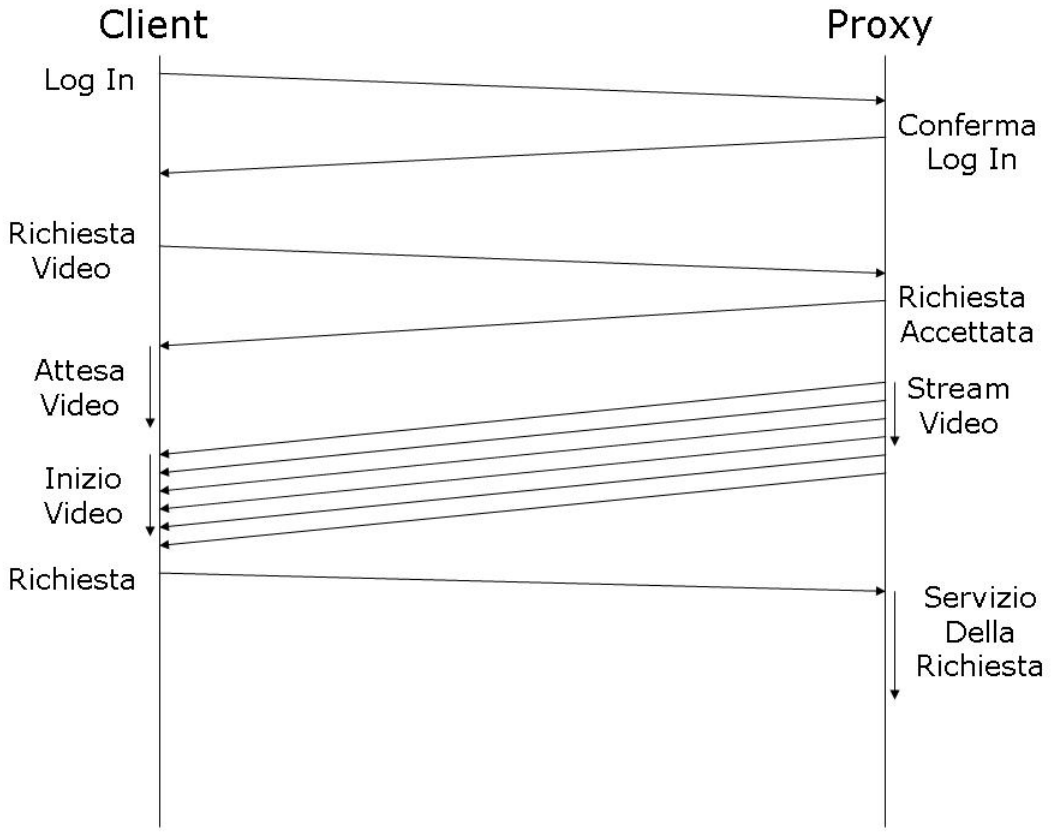


Fig 4.2 – Protocollo comunicazione Client- Proxy

Come possiamo notare c'è una prima fase in cui il Client richiede di essere riconosciuto. Una volta effettuato il Log In il Client può effettuare le sue richieste. In questo caso la prima richiesta sarà quella di ricezione del video. Quando tale richiesta viene confermata il Client si mette in ascolto per ricevere i dati. Durante tale ricezione il Client potrà inviare alcune richieste al Proxy.
Analizziamo ora le singole componenti e la comunicazione li riguarda.

4.3.2 - Player

Il Player video ha in realtà delle funzioni che vanno oltre alla riproduzione, ma possiamo dire in generale che è l'unico componente che si occupa della gestione dello stream video e per questo lo possiamo trattare come una parte che, una volta messa in esecuzione, procede in maniera totalmente parallela ed indipendente dalle altre parti del Client.
Le sue funzioni sono:

- Ricevere lo stream video
- Riprodurlo a video
- Comunicare con il Proxy per coordinare lo stream

Come detto sopra, l'Utente viene identificato tramite un ID. Il Player dovrà essere pertanto informato su quale sia l'ID dell'utente con il quale potrà identificarsi presso il Proxy.
Le altre informazioni richieste dal Player sono quelle per utilizzare il protocollo RTP su cui si basa la ricezione dei dati da parte del Proxy.

Il player è quindi composto da:
- Listener
- Receiver
- CircularBuffer
- Renderer

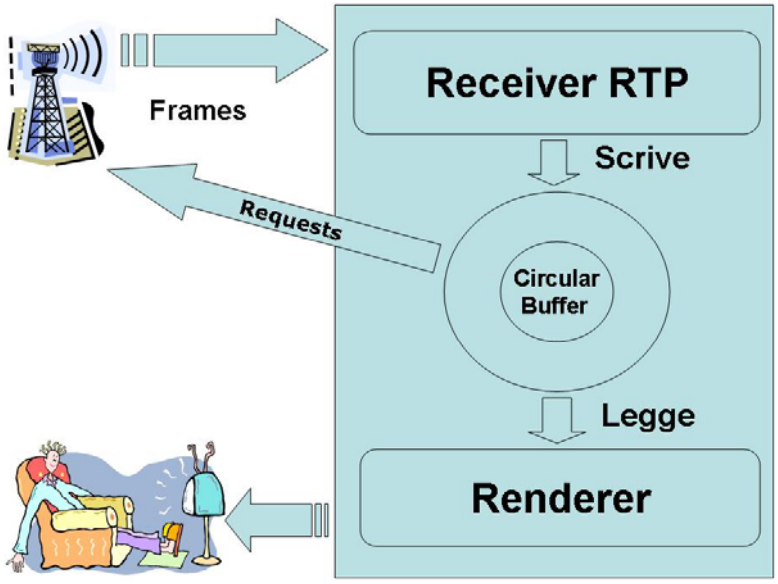


Fig 4.3 – Il Player

Il Listener è un'entità che si mette in attesa dell'arrivo dello stream video utilizzando le librerie del JMF[JMFH].
Egli è a conoscenza della porta su cui mettersi in ascolto e anche della fonte prevista dello stream video. Va detto che può accettare una connessione diretta tra due endpoint ma anche una trasmissione broadcast. In quest'ultimo caso il meccanismo salta in quanto il proxy non si cura minimamente dei destinatari e pertanto esula da questo trattato.
Per gestire il flusso multimediale in arrivo il Listener si appoggia al Receiver. Questi due elementi, insieme al CircularBuffer ed al Rendere, realizzano un architettura tipica produttore consumatore. In questo caso è il Receiver ad agire come produttore inserendo i dati multimediali nel buffer, mentre il Rendere li consuma per visualizzare il video.

Il Receiver è un'entità creata dal Listener ed ha il compito di estrarre i frame dalla traccia dati per inserirli nel CircularBuffer. Il suo unico compito è quindi quello di controllare lo stato del buffer e, se non è pieno, scrivere il frame successivo.
L'altro compito di questo elemento è quello di attivare il Render video. Per far questo attende la prima occorrenza di buffer pieno in modo di attivare l'attività di produzione-

consumo a regime: infatti Receiver e Renderer hanno in ultimo il compito di riempire e svuotare il buffer.
In linea di principio se la velocità di trasferimento dati e la frequenza del video sono ben calibrate, una volta riempito il buffer ed attivato il Render il livello di riempimento del buffer stesso non dovrebbe più cambiare.
Il Receiver, dopo aver eventualmente attivate il Render, continua a catturare frame dalla traccia finché non incorre in un frame che indica la fine dello stream.

Il Renderer, come detto, è un componente che preleva i frame dal CircularBuffer e, una volta individuato il codec adatto, decodifica le informazioni per riprodurre l'immagine a video.
A livello di computazione è senza dubbio l'attività che più influenza le prestazioni della macchina specie in palmari .
Il Renderer è l'unico componente lato client che debba rispettare i tempi dati dal filmato e quindi dovrebbe avere la priorità sugli altri componenti per non subire ritardi nel cambio di frame importante per una buona resa a video del filmato.

Finora appare evidente come nessuno di questi elementi sia coinvolto nella comunicazione col Proxy o meglio, nessuno degli elementi è incaricato di avvertire il proxy di un salto nei frame.
La comunicazione col Proxy deve avvenire unicamente quando c'è una richiesta. Per quanto concerne il Player le richieste possono essere quelle di chiedere la ritrasmissione dati oppure un cambiamento nella velocità della trasmissione.
Questo tipo di osservazioni vengono fatte osservando lo stato del Buffer ed è quindi chiaro che saranno il produttore ed il consumatore ad occuparsi di questo monitoraggio.
I controlli sono fatti invocando delle funzioni apposite sul buffer che ne monitorano lo stato. Il che vuol dire che è il buffer ad inglobare le politiche di gestione ma esse sono attivate da chi usa il buffer, proprio perché questa non è un'entità attiva.
Durante il periodo di connessione normale, ovvero quando la connessione con l'AP è stabile, il ciclo di produzione e consumo sul buffer avviene regolarmente senza interruzioni. Quando avviene un handoff il Client rimane senza connessione per qualche istante e quindi il processo di produzione non può più continuare mentre quello di consumo non è interrotto dall'handoff.
Il controllo più importante effettuato all'interno del CircularBuffer è quello sui dati persi: ogni frame è "marchiato" da un timestamp che definisce una certa successione tra i frame stessi. In seguito ad un handoff alcuni frame vengono perduti ed alla successiva connessione ci si può così accorgere di questa perdita confrontando i timestamp dell'ultimo frame ricevuto e del penultimo.
A questo punto viene inoltrata la richiesta di ricevere nuovamente alcuni frame. I timestamp dei frame successi hanno una certa regolarità ma non si può prevedere con esattezza il timestamp del frame successivo e quindi si spedisce quello dell'ultimo ricevuto correttamente.

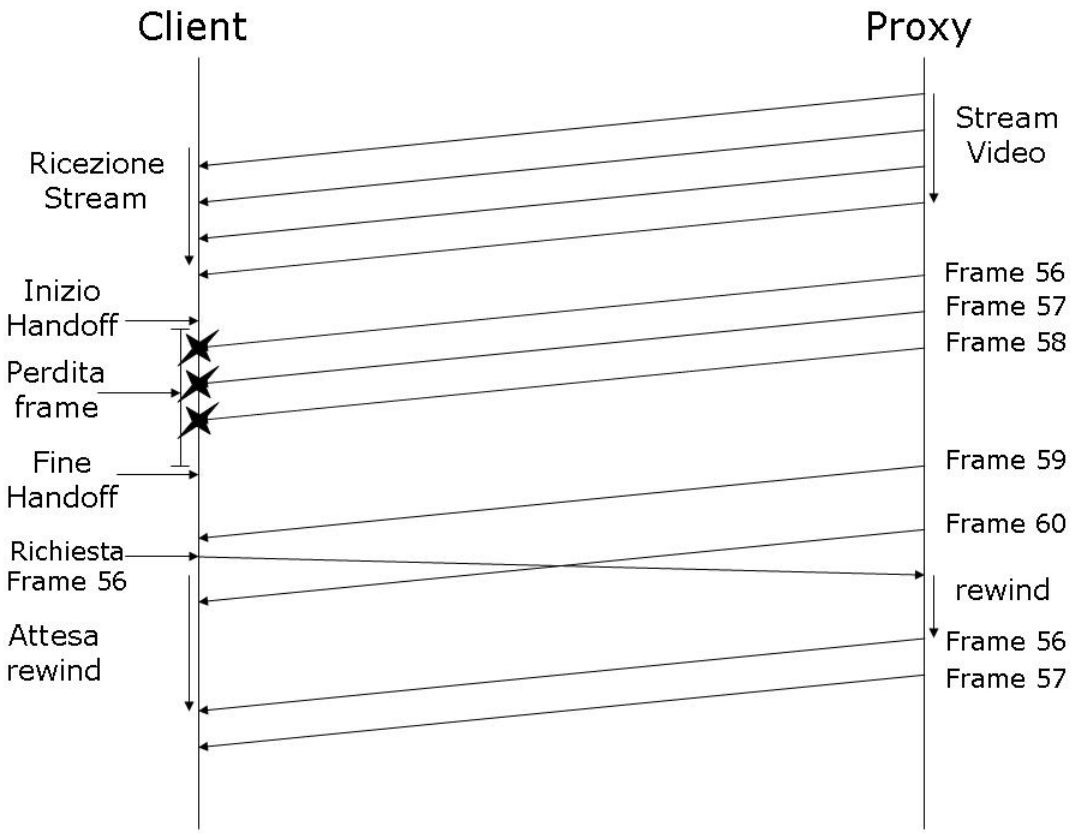


Fig 4.4 – Protocollo rewind

Una volta effettuata la richiesta è necessario mettersi in uno stato di attesa particolare: dato che si sta aspettando un riavvolgimento del filmato non ha senso effettuare altre richieste mentre ancora non sappiamo se la prima avrà esito positivo. Il Proxy continuerà a spedire dati normalmente finché non arriva una richiesta e solo a quel punto tornerà indietro. Nel frattempo egli trasmetterà altri dati. Si potrebbe pensare di non ricevere più dati finché la richiesta non è soddisfatta ma nel caso il Proxy non ricevesse il messaggio oppure non potesse eseguire la richiesta, non ci sarebbe alcun riavvolgimento ed i dati spediti mentre si attendeva la ritrasmissione sarebbero perduti, provocando un ulteriore salto nel filmato. Mentre si attende la ritrasmissione è quindi necessario conservare comunque i dati ricevuti. Nel caso la richiesta venga eseguita, questi dati verranno eliminati e sostituiti da quelli ritrasmessi. Nel caso invece la ritrasmissione non avvenga quei dati saranno utili per proseguire la visualizzazione del filmato.
Il periodo di attesa deve, ovviamente, avere una durata precisa oltre la quale fare ritorno ad uno stato normale di controllo.
Esiste una certa tolleranza nel considerare la perdita di frame: vengono inoltrate richieste solo se il numero di frame persi è superiore ad una soglia determinata sperimentalmente. Questo perché naturalmente avvengono perdite nella rete o perché alcuni frame possono giungere malformati. Inoltre, mentre in condizioni di ottimo segnale queste perdite sono praticamente nulle, in alcune situazioni potrebbero diventare troppo frequenti ed appesantire il sistema provocando più danni che miglioramenti, visto che la perdita di uno o due frame risulta spesso difficilmente avvertibile dall'occhio umano.

Rimane comunque un parametro che può essere cambiato.

Il secondo controllo che viene effettuato sul CircularBuffer è sul livello di riempimento dello stesso: infatti il numero di frame presente all'interno del buffer dovrebbe essere mantenuto entro certi livelli per evitare malfunzionamenti.
Nel caso il livello dovesse abbassarsi per qualche motivo si rischia lo svuotamento e quindi l'interruzione da parte del filmato.
Quindi, quando il livello di riempimento scende sotto una certa soglia, viene spedito un messaggio al Proxy chiedendo di aumentare un poco la velocità.

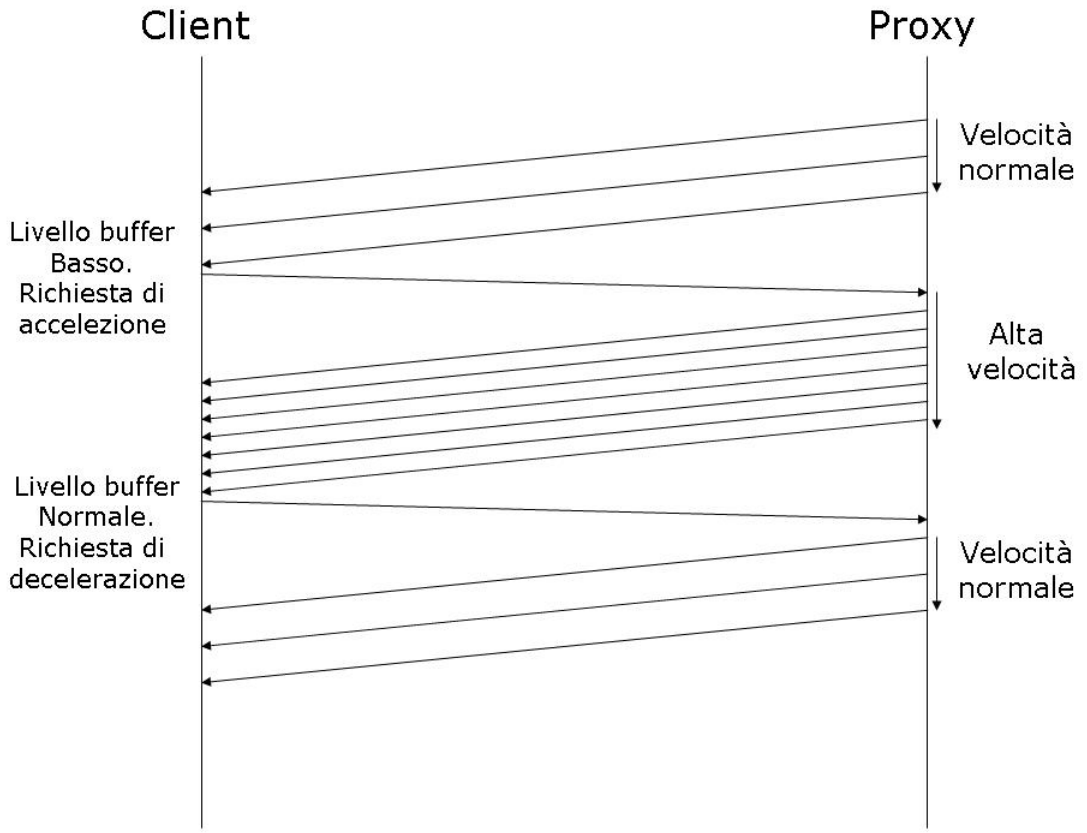


Fig 4.5 – Protocollo Accelerate

In realtà questo meccanismo è particolarmente utile per risolvere la condizione dopo un handoff: mentre siamo nel momento di buio i frame continuano ad essere consumati ma nessuno li rimpiazza causando uno svuotamento temporaneo. Quando, alla ripresa della connessione, chiediamo di ricevere nuovamente alcuni frame, il buffer è comunque in una situazione di parziale riempimento e quindi deve richiedere al Proxy di accelerare la trasmissione dei dati. Oltre ad una richiesta di accelerazione è prevista anche una richiesta di ripristino della normale velocità ed anche una di rallentamento: in seguito ad alcune accelerazioni successive si può venire a creare la situazione in cui il buffer RTP è eccessivamente pieno. In questi casi le librerie del JMF [JMFH] scartano automaticamente i pacchetti entranti poiché non possono eccedere una certa dimensione del buffer interno (che non è il CircularBuffer).

Visto che c'è un certo ritardo tra quando il Buffer può accorgersi del riempimento ed il momento in cui il Proxy può ricevere la richiesta di ripristino dello stato di normalità, non si aspetta il riempimento totale del Buffer ma si anticipa la richiesta di uno o due frame.
Va inoltre sottolineato come questa verifica deve essere richiesta dal Receiver, ovvero dal produttore, se non si vuole che la richiesta vada persa: come abbiamo detto il Renderer, che altro non fa che consumare i pacchetti, continua ad operare anche durante un handoff. Su fosse il Renderer a richiedere il controllo sul livello del buffer le condizioni di svuotamento sarebbero verificate durante il momento di buio. A livello di applicazione non ci accorgiamo dell'handoff se non dalla perdita dei dati. Il Rendere quindi spedirebbe la richiesta non appena il livello scende sotto un certo livello ma possibilmente all'interno di un handoff, rendendo così vana la trasmissione del messaggio. La scrittura invece avviene solo quando siamo connessi e riceviamo dati. Quindi è opportuno che sia il Receiver ad invocare il controllo sul buffer.

4.3.3 - Client Stub

Questo componente ha il principale compito di interfacciarsi con la scheda Wireless ed interrogarla per stabilire la situazione della connessione.
Esso incorpora in i meccanismi della previsione che però vedremo in dettaglio nella prossima sezione Appena partita la Stub chiede al WirelessManager un'istanza di un WirelessCardController da interrogare.
Si è scelto di svincolare la Stub dalla conoscenza della Card Wireless per rendere più estendibile il sistema: così facendo infatti si potranno aggiungere vari tipi di controller magari fatti su misura per certi tipi di schede ma che rispettino la stessa interfaccia. Come vedremo più avanti, questo tornerà utile anche nella simulazione.
In primo luogo la Stub attende che la connessione sia effettuata e cioè che il WirelessCardController sia pronto. Dopodichè interroga il Controller per sapere che tipo di scheda essa sia.
Attualmente sono previsti tre tipi di schede Wireless: Orinoco, Cisco ed una di Default.
In base a questa informazione la Stub predispone alcuni parametri per la previsione.
Come già sottolineato esiste una grande varietà di schede Wireless e soprattutto ognuna segue una diversa politica per l'handoff.
Queste rende assai complicato interfacciarsi con le schede e trovare una soluzione ottimale per tutte.
La Stub interroga il WirelessCardManager ogni secondo: questi fornisce l'AP a cui si è attualmente collegati ed una lista degli AP visibili con le relative potenze di segnale.
In basi a questi dati il sistema di previsione è in grado di darci una probabilità di spostamento.
Sono previsti due status:
LOW_PROBABILITY
HIGH_PROBABILITY

Nel momento in cui la Stub si accorge di cambiamento nello status del Client "sveglia" il Controller che nel frattempo si era messo in sospensione il quale provvede ad avvertire il Proxy.

4.3.4 - Controller

Il componente che realizza il Controller è l'entry point dell'applicazione.
Suo compito è avviare le altre entità come il Player e la Stub e controllare il loro operato.
Per prima cosa il Controller crea un'istanza della ClientStub ed attende che essa sia pronta.
Una volta stabilito che esiste una connessione invia la richiesta di Log In al Proxy.
In seguito il Client rimane in attesa di una risposta dal Proxy dove vengono passati i dettagli relativi alla sessione RTP per poter trasmettere il filmato ed il framerate del filmato stesso
Quindi il Controller provvede a creare un CircularBuffer ed un Listener a cui passarlo in modo da far partire l'applicazione video.
Non appena inoltra la richiesta al Proxy il Client attiva anche un'altra entità che ha il compito di inviare un segnale di Acknoledgement ogni sette secondi al Proxy: questo perché utilizzando socket di tipo UDP serve un controllo a livello applicativo per verificare la presenza dell'altro.
In seguito il Controllore rimane in uno stato di attesa di eventi e può essere risvegliato dalla Stub quando lo status del Client cambia. Il Controllore a quel punto controlla il cambiamento dello stato segnalato dalla Stub e, se necessario, provvede a segnalare tale cambiamento al Proxy.

4.3.5 – Definizione dei messaggi

Diamo ora alcuni dettagli relativi all'implementazione di questi protocolli.
Innanzi tutto abbiamo detto che la comunicazione avviene utilizzando RTP over UDP.
Insieme ad un canale RTP ne viene aperto uno di controllo per l'RTCP. Il sottostrato utilizzato per la gestione del protocollo RTP non ricerca le porte UDP in maniera autonoma ma richiede che sia passata la porta per l'RTP. La porta per l'RTCP non viene chiesta perché si considera che sia sempre quella successiva a quella usata per RTP.
Dobbiamo quindi assicurarci che la scelta ricada su due porte contigue entrambe libere.
Questo viene fatto richiedendo le porte ad una classe chiamata PortManager che si occupa di controllare lo stato delle porte e di passarne una coppia libera.
Diciamo inoltre che, ad eccezione del CircularBuffer, quasi tutte le entità sono oggetti attivi chiamati thread(processi leggeri)per rendere la loro esecuzione parallela.
Le richieste vengono effettuate utilizzando Socket UDP e pertanto inviate all'interno di pacchetti UDP.
Le richieste seguono una formattazione precisa che ora illustriamo.

Ogni richiesta ha un formato di questo tipo:

Fig 4.6 – Struttura richiesta generica

Client ID: è un intero lungo che identifica l'utente per quella sessione.
Request number: è un intero che identifica la richiesta.
Request fields: sono in numero e tipo variabile e dipendono dalla richiesta.
Richiesta di stream video:

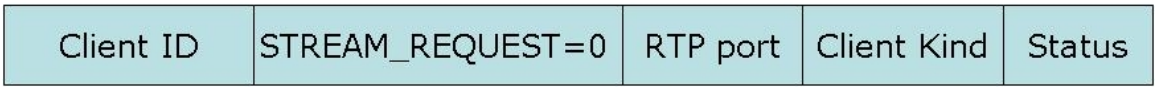


Fig 4.7 – Richiesta di Stream

RTP port: prima di inviare la richiesta al Proxy il Controller chiede al PortManager una porta disponibile per ricevere utilizzando il protocollo RTP ed in seguito la invia al Proxy per indicargli l'endpoint a cui spedire i pacchetti.

Client Kind: è in intero per definire che tipo di scheda Wireless monti l'utente ed in base a questo decidere la politica migliore. In base a questa informazione il Proxy potrà adattarsi per trovare il miglior compromesso tra prestazioni ed utilizzo delle risorse.

Status: indica lo status iniziale del Client.

Richiesta di cambiamento stato:

Fig 4.8 – Richiesta di cambio di stato

Status è un intero
0→ LOW PROBABILITY
1→MEDIUM PROBABILITY
2→HIGH PROBABILITY

Attualmente il sistema di previsione utilizza solo gli stati 0 e 2 poiché la tempistica non rende necessario transitare prima per uno stato intermedio come vedremo più avanti nei risultati sperimentali.

Richiesta di rewind

Fig 4.9 – Richiesta di Rewind

Timestamp è l'indicatore utilizzato per identificare il frame.

Richiesta di modifica della velocità:

Fig 4.10 – Richiesta di cambio velocità

Velocity è un intero:
0→LOW VELOCITY
1→NORMAL VELOCITY
2→HIGH VELOCITY

4.4 – Previsione.

Nel capitolo 3 abbiamo sottolineato l'importanza di poter prevedere gli spostamenti del Client per poterli comunicare al Proxy in modo da gestire al meglio gli handoff senza eccessiva occupazione di memoria. Ora vedremo in dettaglio come si agisce per poter prevedere questi spostamenti. Prima di tutto daremo un'idea di come agiscono le schede per prendere le loro decisioni. Quindi, su questa base, illustreremo nel dettaglio l'idea che sta dietro la previsione.

4.4.1 – Strategie di handoff

Nel capitolo 2.5.2 abbiamo racchiuso le strategie di gestione dell'handoff seguendo alcune principali caratteristiche individuabili.
Ci concentriamo in particolare su schede che utilizzano strategie Hard Proactive e Soft Proactive[PreRSSI]. Ricordo brevemente che le prime tendono a considerare solo il confronto tra gli RSSI degli AP visibili per decidere se migrare o meno mentre le seconde osservano anche la bontà del segnale dell'AP a cui sono associate: fintanto che il segnale è considerato buono, anche se esiste un AP visibile con un segnale migliore, non si effettua un handoff. In generale possiamo dire che le strategie Hard Proactive cercano sempre di offrire l'associazione con l'AP con il miglior segnale causando spesso un numero elevato di handoff, invece le Soft Proactive cercano un compromesso tra la potenza del segnale ed il numero di handoff.

Il sistema di previsione è stato creato quindi seguendo queste due principali linee guida e pertanto, attualmente, l'informazione necessaria sul tipo di scheda che si sta usando è riconducibile alla strategia di gestione dell'handoff implementata nella scheda stessa.

Una volta a conoscenza del tipo di strategia adottata dalla scheda Wireless, si può decidere la politica di previsione da adottare.

Il sistema di previsione prevede tre stati:
LowProb: se è altamente improbabile un handoff nel prossimo futuro.
HighPro: se è altamente probabile un handoff nel prossimo futuro.
MedProb: negli altri casi.

La definizione di questi tre stati cambia a seconda di quale strategia stiamo seguendo.

Hard Proactive:

LowProb: se l'RSSI dell'AP attuale è più altro di quello di ogni altro segnale presente di almeno una soglia fissata chiamata HST (Hysteresis Superior Threashold).

HigProb: se esiste almeno un AP visibile che abbia l'RSSI più alto di quello dell'AP attuale meno una certa soglia chiamata HIT (Hysteresis Inferior Threashold).

MedProb: negli altri casi.

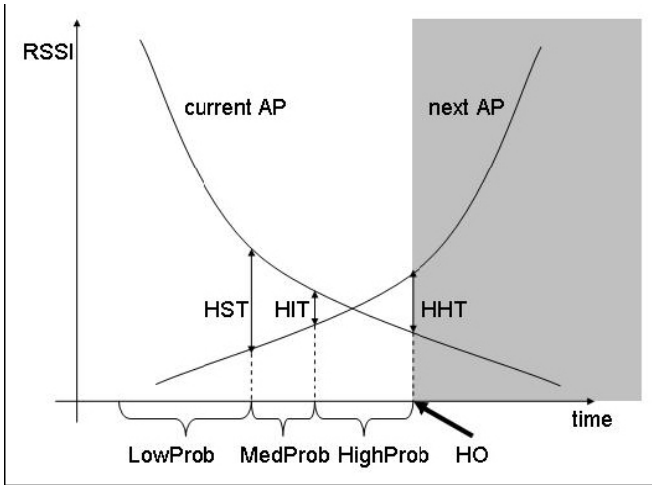


Fig 4.11 – Hard Proactive

Per commentare il grafico possiamo prendere ad esempio una situazione piuttosto semplice: supponiamo di esserci appena connessi e di essere praticamente sotto l'AP e che sia visibile unicamente un altro AP ad una certa distanza.
Dopodichè supponiamo di cominciare a muoverci verso l'altro AP.
Inizialmente i due segnali sono molto differenti tra loro e pertanto la possibilità di un cambio di AP è estremamente bassa. Fintato che la differenza tra i due segnali timane inferiore all'HST l'utente rimane in LowProb.
Preseguendo verso l'altro AP l'RSSI del primo cala mentre l'altro cresce fintanto che la differenza non scende sotto l'HST.
Mentre l'utente prosegue verso l'AP successivo rimane in MedProb fintanto che la differenza tra il segnale dell'AP corrente e l'altro rimane tra HST e HIT.
Quando la differenza tra i due segnali si fa inferiore all'HIT si entra in uno stato di HighProb con un certo anticipo rispetto all'avvento dell' handoff

Soft Proactive

LowProb: se l'RSSI dell'AP corrente è maggiore di ogni altro RSSI di almeno una soglia pari all'HST oppure se è superiore ad un valore chiamato FST (Fixed Superior Threshold)

HighProb: se l'RSSI è inferiore ad un certo FIT (Fixed Inferior Threshold) ed esiste almeno un RSSI che differisca da quello corrente per una soglia inferiore all'HIT.

MedProb: negli altri casi.

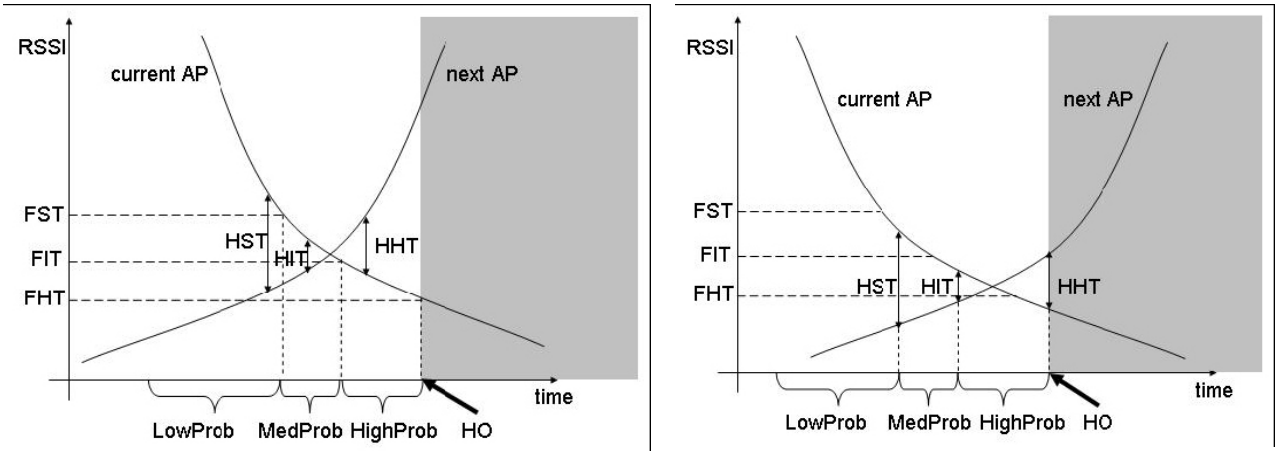


Fig 4.12 – Soft Proactive

Per capire meglio la differenza tra le due politiche facciamo qualche considerazione: è chiaro che una politica Soft tende a conservare maggiormente la connessione con un AP rispetto ad una Hard ma quello di cui non ci si rende conto ad una prima analisi è che le differenze di comportamento tra le due dipendono significativamente dall'ambiente in cui ci si trova.

Se la rete Wireless è tale per cui tra gli AP esiste una distanza considerevole allora in pratica le due politiche coincidono: quando ci allontaniamo da un AP il segnale degrada e ben presto il limite imposto dalla FIT viene meno, rendendo vincolante unicamente la differenza di segnale.

Se invece prendiamo una rete dove esistono molte sovrapposizioni tra aree di AP diversi, una rete magari predisposta per un alto numero di utenti, allora ci accorgiamo che una politica Hard darebbe luogo ad un handoff anche se il segnale attuale è ancora buono.

Tirando le somme possiamo dire che le politiche Hard prendono in considerazione unicamente la differenza tra gli RSSI provocando con il pericolo di provocare handoff inutili laddove il segnale sia comunque buono mentre le politiche Soft prendono in

considerazione di effettuare l' handoff solo dopo che il segnale è sceso sotto un certo livello, dopodichè si comportano esattamente come le prime.

4.4.2 - Grey Model

Finora ho parlato di RSSI come se fossero i segnali campionati dalla scheda in un preciso istante. In realtà il sistema di previsione sopra presentato non utilizza gli RSSI così come la scheda li restituisce ma li "filtra" attraverso il Grey Model.
I segnali possono infatti essere soggetti a numerosi disturbi, fluttuazioni e rumori che possono fuorviare rispetto alla situazione reali. Il Grey Model è uno strumento matematico di cui non riporto i dettagli ma che consente operare come un filtro passa basso e di eliminare parte del rumore presente in ogni segnale, ottenendo così curve simili a quelle riportate in Fig 4.12. Utilizzando gli RSSI senza un filtro otterremmo dei valori che non cambiano con continuità ma assumerebbero l'andamento di una linea spezzata che renderebbe inutile il meccanismo di previsione: i valori infatti dipenderebbero eccessivamente dall'istante di campionamento, con la possibilità che il rumore falsifichi la reale situazione dell'utente. Inoltre il Grey Model tiene conto anche della storia passata del segnale in modo da capire l'andamento del segnale stesso e, in un certo senso, predire il suo possibile andamento.
Un parametro da configurare nell'uso del Grey Model è appunto il numero di campioni utilizzare per effettuare la previsione, ovvero quanta storia recente del segnale considerare nel valutare il possibile andamento del segnale.
Più campione prendiamo in considerazione, più l'andamento risulterà regolare e più lento a variare.

4.4.3 - Conferma dello stato

Il sistema di previsione comprende inoltre un altro meccanismo che, sebbene possa apparire analogo, presenta delle differenze e che si è dimostrato efficace.
Sebbene il Grey Model riesca ad eliminare la parte aleatoria dei segnali e consideri la storia passata per valutare il prossimo futuro non può eliminare la componente intrinseca nel fatto che è comunque una persona ad utilizzare la scheda Wireless e quindi potrebbe avere comportamenti del tutto imprevedibili.

Per questo ho pensato di introdurre una conferma dello stato, ovvero, quando si passa da uno stato ad un altro, prima di rendere effettivo il cambiamento il nuovo stato dev'essere confermato per un certo numero di volte.
Un esempio immediato lo si può avere pensando ad un utente che decida di stazionare in una area di confine tra due AP ovvero dove i loro RSSI si fanno confrontabili: capita spesso che una persona non stia ferma su se stessa ma che oscilli intorno ad un punto centrale.
Questo potrebbe dare modo ad una serie di cambiamenti di stato successivi totalmente inutile che in questo modo viene i parte evitato.

Il pericolo è, ovviamente, quello di ritardare troppo la previsione.

4.4.4 – Configurazione della previsione

Alla luce di quanto esposto è chiaro che il sistema di previsione non può essere configurato staticamente. Sulla base della conoscenza della scheda i parametri utilizzati per la previsione cambiano, alcuni vengono considerati ed altri no a seconda della strategia della scheda.
In conclusione faccio un rapido riassunto di tutti i parametri che influenzano il sistema di previsione:

- Numero di campioni per il Grey Model: in pratica quanta della storia precedente del segnale considerare per il calcolo del valore attuale.
- Numero di conferme per il cambiamento di stato: numero di conferme successive di un cambiamento di stato affinché sia considerato attendibile.
- HST (Hysteresis Superior Threshold)
- HIT (Hysteresis Inferior Threshold)
- FST (Fixed Superior Threshold)
- FIT (Fixed Superior Threshold)

La conclusione è che non può esistere una soluzione perfetta per tutte le schede e, soprattutto, per tutte le situazioni ma è necessario cercare un compromesso tra tutti questi fattori.

4.5 – Proxy UDP

Il presupposto è che la comunicazione tra Server e Proxy avvenga tramite rete fissa e che quindi, pur avvenendo anch'essa tramite UDP, sia comunque abbastanza sicura. Non ci occuperemo quindi di garantire le prestazioni in tal senso. Inoltre consideriamo che il Server rispetti i tempi di invio dei pacchietti.
Il Proxy UDP o anche solo Proxy si inserisce tra il Client che richiede il servizio ed il Server che gestisce il filmato.
Suo compito è quindi di fare da intermediario tra i due in modo da rendere il più semplice possibile il lavoro di entrambi.
In questo modo il Server che gestisce il filmato, che potrebbe avere richieste anche da diversi Proxy, ha il solo compito di trasmettere i frame del filmato ad una velocità costante senza curarsi di eventuali pacchetti persi.
Si è inoltre scelto di utilizzare solo connessioni di tipo UDP poiché più leggere e meno impegnative come risorse rispetto a quelle di tipo TCP.
Il Proxy è quindi una bacinella che si riempie ricevendo frame dal Server e si svuota inoltrando i frame al Client.
Non è prevista alcuna comunicazione tra Proxy e Server che esuli dalla richiesta di servizio e dal termine della richiesta stessa.

Potrebbe far riflettere il fatto che non esista alcun meccanismo di richiesta di alcuni pacchetti persi da parte del Proxy quando invece il Client può richiederlo al Proxy.

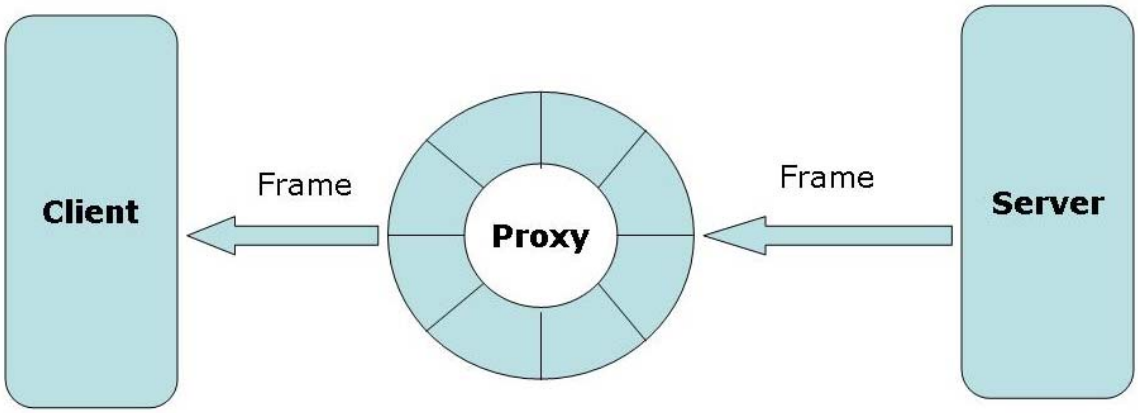


Fig 4.13 – Proxy con buffer circolare

4.5.1 – Proxy Manager

Il Proxy Manager è l'entità incaricata di ricevere le richieste di Log In da parte dei Client e di gestire Proxy che hanno il compito di gestire il servizio video.
Come detto in precedenza tutte le comunicazioni tra Client e Proxy avvengono tramite Socket di tipo UDP e pertanto anche il Proxy Manager riceve le richieste in questa forma.
I Client non vengono riconosciuti ne tramite il loro IP ma unicamente con il loro ID ed è questa l'unica informazione del Client che interessa al Proxy Manager.
Ogni pacchetto inviato dai Client ha come primo campo l'ID di sessione associato all'utente.
Il Proxy Manager, una volta appresa questa informazione, controlla in una propria tabella l'esistenza di un Proxy che stia servendo quel Client: se non esiste allora nessun Proxy sta servendo quell'utente e provvede a crearne uno che da quel momento in poi si occuperà di quel Client.

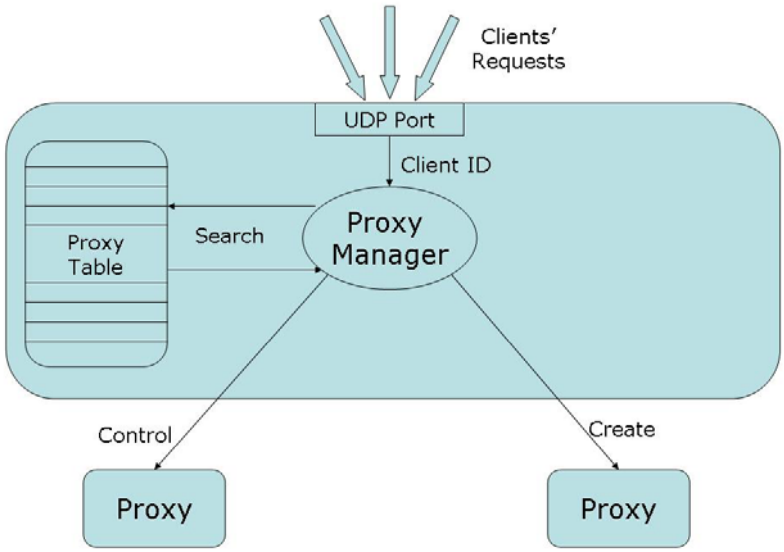


Fig 4.14 – Struttura lato Proxy

Il Proxy Manager inoltre esegue periodicamente un controllo su tutti i Proxy presenti nella propria tabella e controllare il loro stato di attività: se il Proxy ha terminato la sua esecuzione ma non è ancora stato rimosso il Proxy Manager provvede a farlo e ad aggiornare la propria tabella.
Dato che ogni Proxy, conservando frame video, necessità di una quantità non indifferente di memoria questo è un controllo che ho ritenuto opportuno fare anche se ogni Proxy ha il compito di escludersi dalla tabella una volta terminato il proprio compito.

4.5.2 - Proxy

Il Proxy è un componente attivo che viene creato dal Proxy Manager e che viene risvegliato ad ogni richiesta ricevuta dal Client. Suo compito è gestire le richieste del Client una volta creato appositamente per lui.
Alla creazione il Proxy registra l'ID del Client che quindi diventa anche il suo ID ed accetterà solo pacchetti con quell'ID.
Alla sua creazione il Proxy attiva un'entità chiamata AckController che ha il compito di controllare periodicamente l'arrivo dei pacchetti ACK da parte del Client.
Oltre a questo il Proxy apre un canale di comunicazione verso il Client su cui poi resterà in ascolto delle richieste del Client. Le socket create sono di tipo UDP e quindi non vi è alcun controllo sull'arrivo a destinazione delle richieste. L'esecuzione di una richiesta può durare una determinata quantità di tempo durante la quale il Proxy non può restare in ascolto sulla connessione creata. Ricordo che tutte le socket sono di tipo UDP e quindi se un pacchetto viene inviato mentre il Proxy non è in ascolto il pacchetto viene perso.
Per questo il Proxy crea un'altra entità con l'unico compito di ricevere le richieste e girarle al Proxy che le conserverà in un vettore per non perderle.
La comunicazione tra i due è quindi unilaterale ed avviene solo dal Client verso il Proxy.
Per questo è necessario che sia il Client a mandare gli ACK ed il Proxy a controllare l'arrivo di questi con una periodicità prefissata.
Esiste una tolleranza in questa gestione poiché possibile che qualche pacchetto venga perso nella rete oppure che il Client tenti di inviarlo mentre sta avvenendo un Handoff.
Pertanto non basta la perdita di un pacchetto per scatenare l'interruzione del servizio.
Quando si accorge che il Client è scomparso l'AckController invoca la funzione "close()" sul Proxy.
Con questa funzione il Proxy rilascia tutte le risorse occupata e si cancella dalla tabella del Proxy Manager

L'unica comunicazione da parte del Proxy verso il Client avviene quando il Proxy comunica l'avvenuto Log In e quando il Client richiede il filmato ed il Proxy risponde con le informazioni necessarie per la sessione RTP e per la riproduzione del filmato.

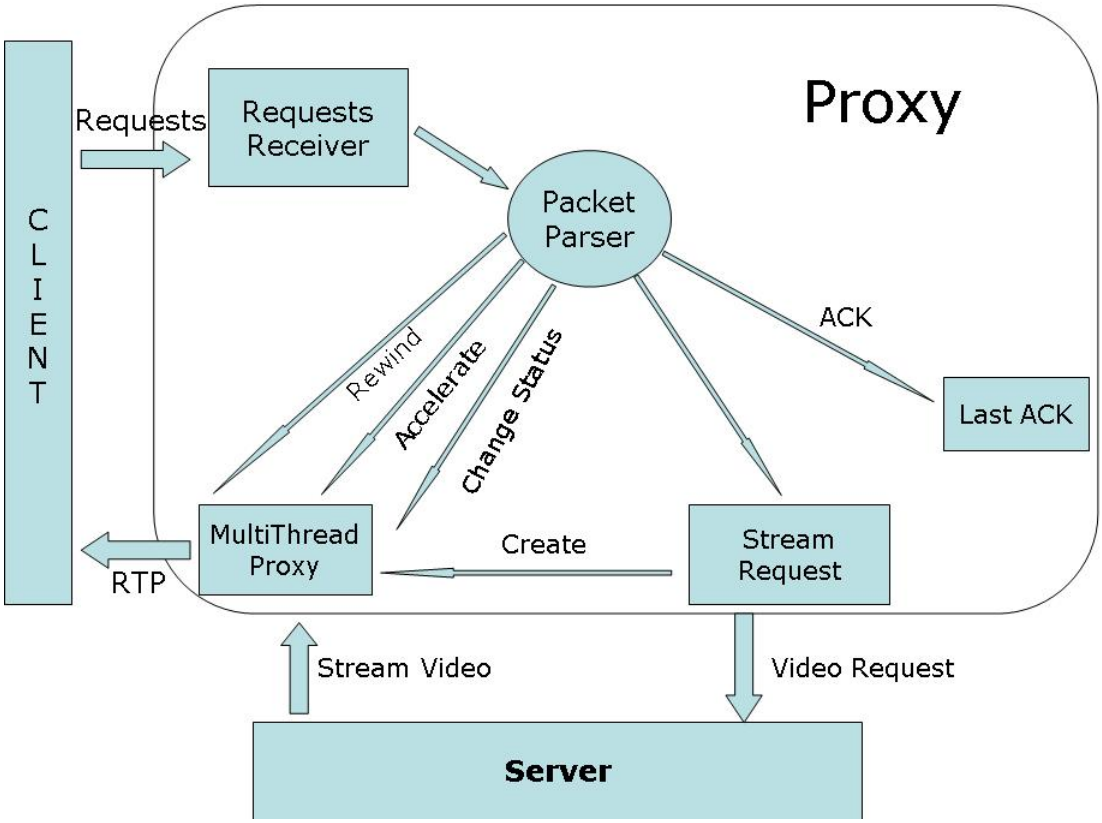


Fig 4.15 – Comunicazioni del Proxy

La figura illustra i principali componenti del Proxy e le interazione tra di loro. Ora vediamo in dettaglio cosa succede quando il Client effettua le sue richieste.
Ad ogni pacchetto giunto il proxy controlla per prima cosa l'ID per controllare che sia diretto a lui; nel secondo campo invece c'è l'indicazione della richiesta. Le funzionalità invocabili sul Proxy sono le seguenti:

4.5.2.1 –Log In

Log In: alla creazione del Proxy il Proxy Manager gli passa il messaggio di Log In del Client. In base a questo il Proxy acquisisce informazioni relative al Client stesso. Sarà lui, e non il Manager Proxy, a dare conferma al Client dell'avvenuto Log In aprendo così un canale diretto di comunicazione con il Client stesso.

4.5.2.2 – Stream Request

indica la richiesta del Client di uno stream video. Il Proxy quindi crea
un componente chiamato MultiThreadProxy che gestisce i frame video provenienti dal Server e li spedisce al Client; fatto ciò il Proxy si connette con il Server in cui risiede il filmato e richiede la trasmissione del filmato con protocollo RTP.

Lo scambio di informazioni tra Proxy e Server è simile a quello tra Client e Proxy: in pratica si scambiano alcune informazioni di controllo per il flusso video come le rispettive porte usate per il protocollo RTP.

4.5.2.3 - Change Status:
il Client segnala un cambiamento di stato. Il Proxy adegua le proprie risorse allargando o stringendo il buffer a seconda dello stato comunicato e del tipo di scheda del Client: come già detto distinguiamo le schede a seconda che utilizzino strategie Hard Proactive oppure Soft Proactive. Infatti con le schede da noi utilizzate abbiamo riscontrato un notevole divario nella durata degli handoff tra un tipo di scheda e l'altra constatando che quella che segue politiche Hard Proactive (Orinoco) ha tempi decisamente più brevi. Nel caso questa informazione sia nota è quindi consigliabile non prendere eccessive risorse nel caso HP e quindi distinguere i diversi casi. In realtà i Client possono appartenere a tre categorie divise per strategie di handoff, di cui le prime due sono Client che usano strategie Jard Proactive e Soft Proactive, ed una di default che in questo caso indica la mancata conoscenza del tipo di strategia adottata. Nel caso la scheda non sia riconosciuta il Proxy tenta di "imparare" che tipo di scheda sta servendo e tenta di adattarsi. Supponiamo che di partenza il Proxy consideri sufficienti 15 frame di riserva per servire le richieste del Client. Se ogni richieste viene soddisfatta "riavvolgendo" il filmato di soli 7 frame allora il Proxy ridurrà pian piano il numero di frame di scorta. Se al contrario molte richieste non possono venire soddisfatte allora probabilmente il Proxy si trova a servire una scheda più lenta ad effettuare handoff e comincerà a conservare più frame.

4.5.2.4 –Rewind
il Client comunica la perdita di un pacchetto comunicando il timestamp dell'ultimo frame ricevuto. In seguito inoltra la richiesta al MultiThreadProxy ed attende il risultato che sarà uguale ad numero di frame "riavvolti". Il meccanismo di Rewind, secondo quanto viene percepito dal Proxy, non fallisce mai ma può dare dei risultati significativi: un riavvolgimento pari a zero è indice o di un errore nella richiesta del Client o di una mancanza di frame preposti al meccanismo di Rewind. Dall'altro lato, un riavvolgimento pari al numero di frame conservati per l'handoff indica che la richiesta potrebbe essere andata oltre alle riserve: infatti se anche la richiesta non può essere soddisfatta completamente, il Proxy rispedisce tutti i frame che ha a disposizione. In questo caso, se il Client risulta sconosciuto, il Proxy allarga la "finestra" di frame conservati poiché è evidente che il Client servito ha esigenze maggiori. Dopo un handoff la situazione che si viene a creare è di un Client che ha il buffer in parte vuoto ed un Proxy che invece sembra avere un buffer più grande di quello prefissato per l'effetto del Rewind. E' quindi opportuno ed utile ad entrambi per ripristinare le condizioni di lavoro ideali che il Proxy spedisca una serie di pacchetti più velocemente per tornare alla condizione di partenza e per far si che il Client riempia nuovamente il suo buffer. Quindi il risultato del Rewind viene usato come misura del numero di frame da spedire più velocemente.

4.5.2.5 - Accelerate:
quando arriva una richiesta di questo tipo il Proxy cambia la velocità di spedizione dei pacchetti per adeguarla alle richieste del Client.

Secondo lo schema logico del programma la velocità normale del filmato ed il meccanismo di accelerazione associato al rewind potrebbero bastare a soddisfare le richieste del Client ma si possono verificare condizioni per cui il Client è costretto a chiedere al Proxy di accelerare momentaneamente la spedizione o di rallentarla.
Una volta effettuato l'handoff il Client fa richiesta al Proxy affinché vengano rispediti alcuni pacchetti persi; il Proxy provvede a rispedire i frame richiesti ed a velocizzare la trasmissione di un numero di pacchetti pari a quelli riavvolti. La velocità con cui questi frame vengono rispediti non può però essere eccessiva se non vogliamo che il Client li scarti: le classi predisposte per il protocollo RTP hanno un buffer privato che usano per ordinare i pacchetti in arrivo; se arrivano pacchetti quando tale buffer è pieno RTP li scarta. Non potendo fare ipotesi sulle prestazioni del Client il Proxy non può quindi accelerare troppo la trasmissione di pacchetti. Mentre tale ritrasmissione avviene il Client continua comunque a consumare pacchetti e quindi anche rispedendo più velocemente quei frame "riavvolti" il buffer del Client si troverà ad avere un parte del Buffer vuoto.

Per chiarificare faccio un esempio: supponiamo che il filmato in questione abbia un framerate pari ad 8 frame/sec (quindi 125 ms tra un frame ed il successivo). Supponiamo che il Client ed il Proxy abbiano un Buffer di 24 frame al momento dell'handoff (ricordo che il Buffer del Proxy è variabile).
Supponiamo che il Buffer del Proxy tenga come riserva 16 frame (pari a 2 secondi).
Nel momento in cui la scheda Wireless perde connessione dall'AP il Client continua a consumare frame ed il Proxy continua a trasmetterne. Per semplicità facciamo sì che l'handoff duri 2 secondi: al termine il Client avrà solo 8 frame nel suo Buffer mentre il Proxy continuerà ad avere il Buffer pieno con 16 già trasmessi (quelli conservati per il Rewind). Appena effettuata la riconnessione il Client si accorge di aver perso dei pacchetti e spedisce al Proxy la richiesta di ritrasmetterli. Supponiamo che non ci siano ritardi in queste richieste e che il Proxy soddisfi immediatamente la richiesta spostando il puntatore di lettura indietro di 16 frame. A questo punto il Client avrò ancora solo 8 frame per la lettura mentre il Proxy avrà 24 frame tutti da rispedire e 0 per il Rewind. Se ora si potesse trasmettere istantaneamente quei 16 frame in esubero la situazione tornerebbe ala normalità sia da una parte che dall'altra. Supponiamo invece che il Proxy trasmetta a velocità raddoppiata impiegando quindi circa 1 secondo per trasmetterne 16. In quel secondo il Client avrà comunque consumato 8 frame per rendere a video il filmato e ed il Proxy avrà ricevuto altri 8 frame dal Server(che conserva in Buffer separati, come vedremo in seguito)

Invece di migliorare questo meccanismo cercando una perfezione forse non raggiungibile si è preferito consentire al Client di chiedere l'accelerazione della trasmissione.
Per questo motivo esiste anche la richiesta di ripristino della velocità normale come termine del transitorio dovuto all'handoff.
La richiesta invece di rallentamento è invece una misura di emergenza nel caso, ad esempio, la richiesta di ripristino della velocità normale venga persa: in questo caso il buffer RTP potrebbero venir riempito e quindi scartare alcuni pacchetti provocando

richieste di Rewind inutili.

4.5.2.6 - Acknowledgement:
questa non è in realtà una richiesta ma la segnalazione di presenza del Client. Ogni volta che il Proxy riceve un pacchetto di questo tipo semplicemente registra il tempo di arrivo. Una entità parallela, chiamata AckController, controlla il tempo registrato per l'ultimo ACK con la stessa periodicità con cui gli ACK stessi devono essere spediti. Se l'AckController riscontra che il Proxy non riceve ACK da un tempo maggiore a 2,5 volte l'intervallo previsto tra un ACK e l'altro invoca la funzione di chiusura sul Proxy. Ovviamente la tolleranza può essere cambiata in modo da adattarla a particolari condizioni.

4.5.3 – MultiThread Proxy

Questo componente ha il compito di ricevere uno stream multimediale tramite protocollo RTP e di ritrasmetterlo verso un Client.
In pratica questo è il componente che realizza la "bacinella" in cui vengono conservati i frame da spedire al Client.
Questo componente comprende altre entità come un Parser ed un Multiplexer. Il suo nome deriva dal fatto che il primo Proxy scritto per questo genere di applicazione era un unico thread(processo leggero) che incapsulava tutte le varie funzioni che ora sono divise. Questo ha permesso un maggiore controllo sulle operazioni svolte con l'obiettivo di raggiungere i nostri scopi.
Le entità attive sono il Parser, il Multiplexer, l'RTP Receiver e l'RTP Sender.

Il MultiThread Proxy è in realtà un controllore di altre entità che offre un'interfaccia unica a chi lo crea per effettuare alcune operazioni.
I principali componenti sono:
- RTP Receiver
- RTP Sender
- Parser RTP
- Parser
- Multiplexer
- Circular Buffer.

Alla partenza il Multithred Proxy prepara il Receiver RTP sulla porta concordata tra Proxy e Server.
In seguito il Receiver viene passato all'RTPparser il quale a sua volta viene gestito dal Parser.
Compito del Parser è estrarre i frame multimediali dalla traccia per poi inserirli in un buffer circolare.

Il Multiplexer è il componente incaricato di estrarre i frame dal Buffer circolare e prepararli per essere spediti dall'RTP sender.

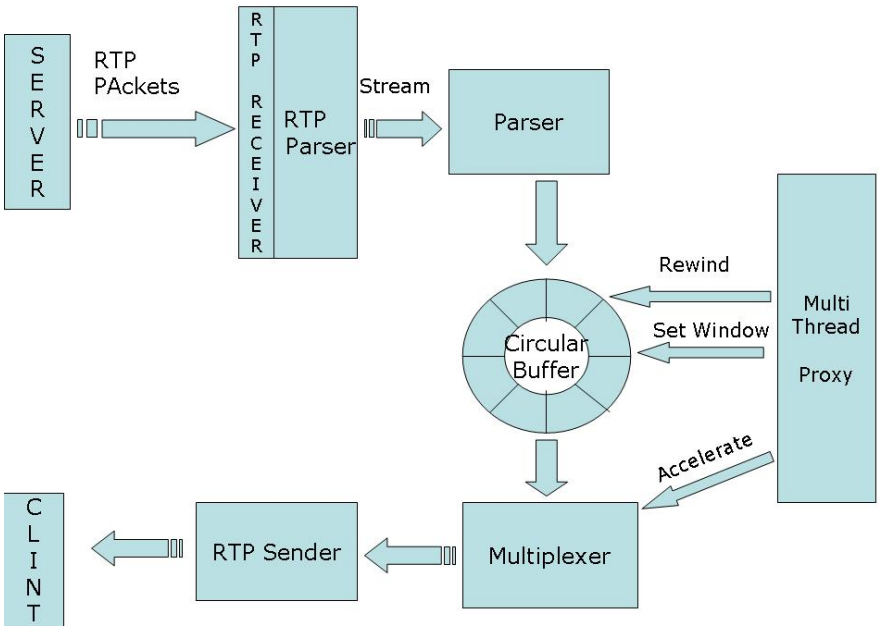


Fig 4.16 – Catena pacchetti RTP lato Proxy

4.5.3.1 - CircularBuffer

Il CircularBuffer è l'elemento che più incorpora il meccanismo di ritrasmissione dei pacchetti e di conservazione degli stessi.
In generale può essere visto come un normale buffer circolare in cui gli elementi, anche se già letti, vengono conservati finché non vengono soprascritti da scritture successive.
Il buffer internamente non è altro che un array di puntatori di oggetti "Frame" .
Il buffer ha inizialmente una certa dimensione, che corrisponde a quella dell'array di puntatori.
Inizialmente il puntatore per la scrittura ed il puntatore per la lettura puntano al medesimo blocco del Buffer. Ad ogni frame immesso il puntatore di scrittura avanza così come quello di lettura ogni volta che il Multiplexer legge un frame. In breve si ottiene la situazione seguente:

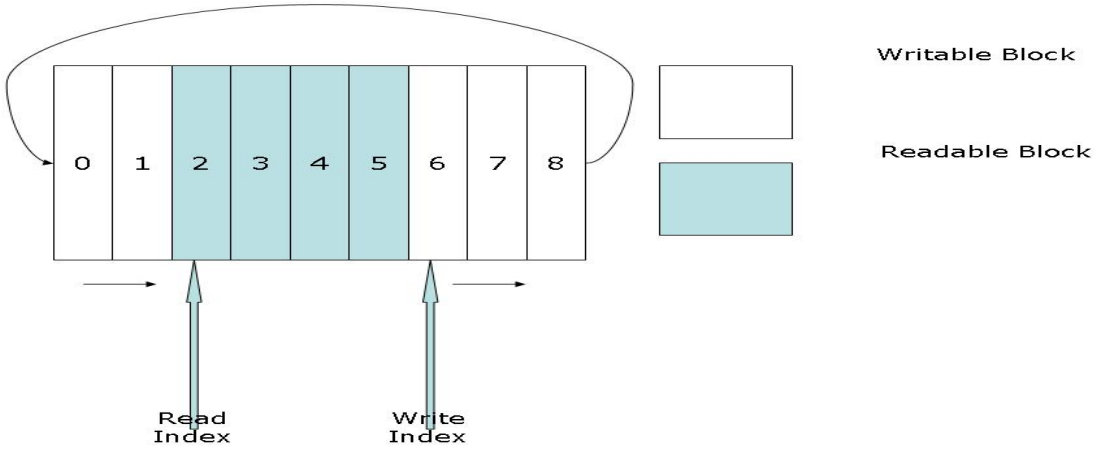


Fig 4.17 – Buffer circolare

I Frame già letti sono frame su cui si può scrivere.
Ovviamente se l'indicatore READ raggiunge WRITE il Buffer è vuoto ed è pieno nella situazione contraria.
In questo caso il Buffer funziona come un qualsiasi Buffer circolare e non ha nulla di particolare.
Tornando al nostro caso di interesse possiamo notare come i frame già letti costituiscano la riserva di frame di cui noi necessitiamo nel caso il Client abbia perso pacchetti in seguito ad un handoff: possiamo tornare indietro di tanti pacchetti quanti sono quelli disponibili per la scrittura (questo a regime: all'inizio dell'applicazione tali frame sono semplicemente "vuoti", non ancora scritti).
Si è quindi imposto che un certo numero di frame venga sempre lasciato disponibile per effettuare queste operazioni che ho chiamato di "Rewind": si aggiungono un certo numero di puntatori all'array che chiamiamo "Rewind Window" e che non risultano nella dimensione del Buffer.
Quando creiamo il CircularBuffer dovremo quindi specificare dimensione del Buffer stesso e della Rewind Window: in questo modo la memoria occupata sarà quindi maggiore rispetto al caso precedente ma per quanto riguarda la funzionalità base del Buffer non c'è alcuna modifica:

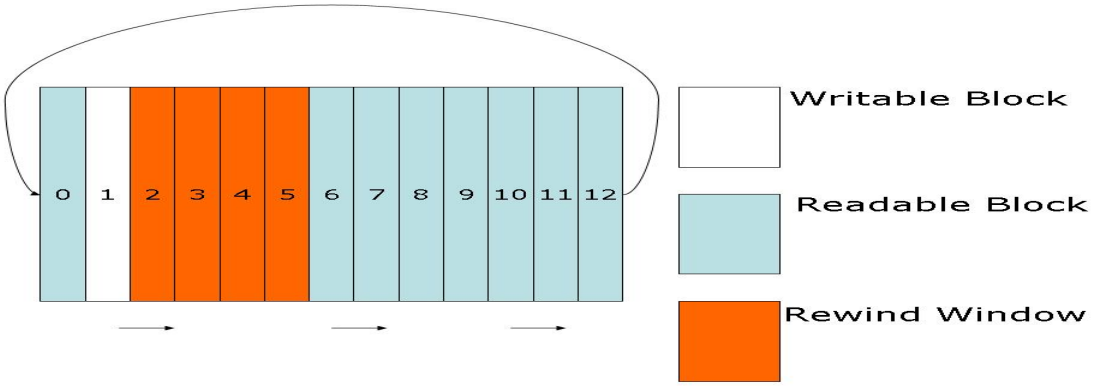


Fig 4.18 – Buffer circolare con finestra di rewind

Rispetto alla figura precedente sono stati aggiunti quattro blocchi ma sono stati riservati come Rewind Window, quindi, dal punto di vista dell'operatività in quanto buffer, la sua dimensione non è stata modificata.

La Rewind Window può essere modificata durante l'esecuzione, ovvero, in pratica, il Buffer può essere ristretto oppure allargato.
Questo per consentire un'occupazione di memoria intelligente in base alla situazione attuale del Client: come già detto si prevedono tre stati possibili e ad ognuno si associa una dimensione della Rewind Window.
Faccio notare come, sebbene sia possibile, non si va mai ad intervenire sulle dimensioni operative del Buffer in quanto queste non vengono coinvolte nelle operazioni di Rewind.

Meccanismo di Rewind

Quando il Client effettua una richiesta di ritrasmissione spedisce anche il timestamp dell'ultimo pacchetto ricevuto.
Questo è necessario affinché il Buffer possa capire da dove ricominciare a trasmettere i frame persi. Ogni pacchetto che il Buffer riceve viene "marchiato" con un timestamp che segue una certa progressione e che viene usato, appunto, per riconoscere tale frame quando si effettua un riavvolgimento.
Quando si effettua la richiesta il Buffer torna indietro fintanto che o trova il frame richiesto oppure raggiunge il puntatore di scrittura: va quindi fatto notare che la Rewind Window è il minimo dei pacchetti che si vogliono presenti per effettuare riavvolgimenti ma che, nel caso sia possibile, può essere effettuato un riavvolgimento maggiore: nel caso in figura ci sono quattro frame di Rewind Window ed uno già letto, quindi in totale il riavvolgimento possibile è di cinque frame.
Inoltre, anche se non si riesce a soddisfare pienamente la richiesta del Client perché il frame richiesto è già stato riscritto, il Buffer torna comunque indietro per quanto possibile, minimizzando l'errore dovuto all'handoff.

Il riempimento del Buffer viene valutato come la distanza tra il puntatore di scrittura e quello di lettura: quando si effettua un riavvolgimento il puntatore di lettura torna indietro causando un apparente riempimento del Buffer che supera le dimensioni inizialmente prefissate. Per questo è necessario, subito dopo un Rewind, spedire più velocemente in frame in eccesso al Client: in questo modo si ripristina il livello del buffer del Client e la Rewind Window.
E' chiaro infatti che due richieste consecutive di Rewind difficilmente possono venire soddisfatte in quanto i frame di riserva vengono spesso consumati dalla prima richiesta per più della metà.

Variazione delle dimensioni.

E' possibile, come detto, variare le dimensioni di Buffer.
Il meccanismo non è complicato in quanto il buffer è un array di puntatori ad oggetti "Frame" e quindi è sufficiente costruire un altro array e copiare i puntatori aggiungendo o sottraendo alcune posizioni.

In realtà ad aumentare e diminuire è la sola Rewind Window ma il meccanismo per una variazione del Buffer operativo è del tutto analogo.
Chiaramente l'aumento di dimensione non causa alcun problema poiché si tratta, logicamente, solo di aggiungere delle posizioni che poi si andranno a riempire.
Al contrario, lo svuotamento, può ed anzi determina certamente la perdita di alcuni frame.
Poniamo per esempio che il buffer sia originariamente di 15 frame, 10 operativi e 5 di Rewind Window e che in seguito ad una richiesta del Client divenga di 20 frame, 10 operativi e 10 di Rewind. Se avviene un handoff in questo frangente il Client chiederà di ritrasmettere alcuni frame.
Come abbiamo visto durante un riavvolgimento le condizioni base vengono meno: ovvero si ha, per qualche istante, una condizione anomala per cui i frame operativi superano la dimensione originaria. Supponiamo che la richiesta di Rewind richieda tutta la finestra di Rewind e che quindi la condizione seguente sia di 20 frame operativi e 0 di rewind.
In questo caso il buffer risulta pieno fino a che i frame operativi non tornano al numero previsto, ragion per cui è utile velocizzare l'operazione di ritrasmissione al fine di non dover scartare i pacchetti inviati dal server.
In una situazione simile non è possibile effettuare un'operazione di riduzione della finestra di Rewind in quanto si andrebbero ad eliminare frame ancora da trasmettere.
Quindi all'invocazione della funzione che ha il compito di ridurre la finestra, se le condizioni non sono quelle adatte, viene impostato un valore chiamato "targetWindow".
Quando questo valore è maggiore di 0 ad ogni scrittura o lettura il Buffer ha il compito di controllare la possibilità di effettuare il ridimensionamento della finestra e, se possibile, farlo.
Se più richieste si sovrappongono, ovvero un'altra richiesta di cambio dimensione è invocata mentre è ancora pendente la precedente, solo l'ultima viene presa in considerazione.
Questo meccanismo svincola il thread che opera la richiesta dall'attesa di avvenuta operazione aumentando la possibilità di parallelismo tra i vari thread.

4.5.3.2 - Multiplexer

Il Multiplexer è incaricato di prelevare i frame dal CircularBuffer e codificarli in modo da essere spediti come stream RTP.
E' il componente che tra tutti è responsabile del rispetto della frequenza di trasmissione e quindi dev'essere implementato in modo che questa venga massimamente assicurata.

Rewind
Il Multiplexer mette a disposizione una funzione che permette dei accelerare la trasmissione per un numero finito di frame. Questa funzione è stata pensata per essere eseguita subito dopo ad un Rewind. Se un'altra richiesta viene effettuata mentre è ancora pendente una precedente, semplicemente le richieste si sommano.

Il Multiplexer prevede inoltre una funzione che possa modificarne la velocità di esecuzione in modo da ripristinare il livello del Buffer nel Client.
Il Multiplexer gestisce anche l'RTP transmitter a cui passa i pacchetti una volta effettuata la computazione degli stessi, anche se è più corretto dire che è il transmitter a prenderseli non appena sono pronti.

Capitolo 5 – Il Simulatore di ambienti Wireless

5.1 – Simulazione ambiente

Durante lo sviluppo di strumenti per supportare la continuità di servizio durante connessioni di tipo wireless si è reso evidente il fatto che molte valutazioni potevano essere fatte unicamente sperimentandole poiché dipendenti dalla velocità di esecuzione delle macchine e anche per la presenza di alcuni fattori aleatori.
Come abbiamo visto, ad esempio, il numero dei parametri da configurare per la predizione è elevato e soprattutto non può esistere una configurazione ottima per ogni caso.
Lo scopo del progetto è trovare una soluzione che offra un buon compromesso tra costo della soluzione stessa ed i risultati ottenuti.
Infatti le prestazioni del sistema di previsione possono variare a seconda dell'ambiente circostante, da come si muove l'utente e dalla scheda wireless dell'utente stesso.
E' quindi chiaro che non esiste una configurazione ottimale per tutte le situazioni ma che va cercata una configurazione che assicuri mediamente buone prestazioni.
Serviva quindi uno strumento in grado di poter garantire una sperimentazione con configurazioni variabili ma soprattutto simulazioni con un numero di utenti elevato: oltre al fatto che, pur avendo una rete wireless a disposizione, effettuare numerose sperimentazioni era costoso in termini di tempo ed era praticamente ancora più difficile sperimentare il carico sul sistema dovuto alla presenza di molti utenti contemporaneamente.
Va subito detto che lo scopo di questo simulatore è di rispecchiare la realtà quel tanto che basta per poter sperimentare le soluzioni trovate e non pretende di essere veritiero in qualsiasi situazione per quando i valori rilevati siano stati confrontati con quelli reali e corretti fino a che potessero essere plausibili.
La soluzione stata trovato nell'implementazione di un simulatore che permettesse ti provare le applicazioni fingendo di muoversi in una rete wireless.
Il simulatore integra ed estende un simulatore pre-esistente pensato per simulare il percorso di un solo utente per volta.
Il calcolo della potenza di un segnale può rivelarsi assai difficoltoso non appena cominciamo ad introdurre nell'ambiente la presenza di persone, mobilia, muri, superfici riflettenti ed altri elementi di disturbo.
Il Simulatore è suddiviso in varie parti proprio per permettere di simulare una grande varietà si situazioni ed implementa delle interfacce per permettere la creazione di situazioni ad hoc.

5.1.1 - Contesti

Per valutare le prestazioni di un'applicazione wireless è essenziale comprendere che le condizioni possono essere del tutto variabili da ambiente ad ambiente. Molte reti wireless sono costruite secondo una topologia che permettere di coprire il massimo dell'area con il minor numero di AP.

Ci sono però molte situazioni in cui si preferisce adottare altre configurazioni come ad esempio in alcuni corridoi di passaggio dove non è previsto che le persone stazionino oppure una sala conferenze dove magari è previsto che l'area di due o più AP si sovrappongano per garantire un servizio efficiente ad utenti ravvicinati.

Per consentire di ottenere una pluralità di contesti è stata creata un'interfaccia che tutti i contesti devono rispettare.
Ogni contesto può avere la forma di un rettangolo di qualsivoglia dimensione ed una lista di AP con relativo limite di potenza espresso in db. Questo non è limitante in quanto praticamente ogni contesto reale è costituito da una stanza alle cui parete sono agganciati degli AP. Per simulare un contesto aperto è invece sufficiente predisporre una stanza molto grande e piazzare gli AP in mezzo ad essa.

Uno dei contesti più frequenti è quello chiamato "Contesto ad Esagoni" in cui l'area di un AP è paragonata ad un esagono e gli AP sono disposti in modo da formare una sorta di alveare con le aree degli AP.

5.1.1.1 – Contesto ad esagoni

Le simulazioni sono state fatte prendendo come contesto ambienti con AP lontani ed AP vicini in modo da testare la validità della soluzione nei contesti più vari. Come ho detto durante la spiegazione del progetto non esiste una configurazione dei valori ottima per ogni situazione. La disposizione degli AP segue però, generalmente, un certo criterio che è quello della massima copertura dell'area con il minor numero di AP senza però lasciare aree scoperte. Il contesto più canonico viene chiamato "contesto ad esagoni": in questo tipo di contesto l'area di copertura di un AP può essere vista come un esagono. Questa è ovviamente una approssimazione poiché la forma più simile ad un'area di copertura di un AP è un cerchio. Il contesto ad esagoni permette però di non lasciare scoperta nemmeno una porzione di area e di far sovrapporre al massimo due aree di copertura, minimizzando così lo spreco.
Evidenziando la posizione degli AP quello che si ottiene è una figura simile a questa.

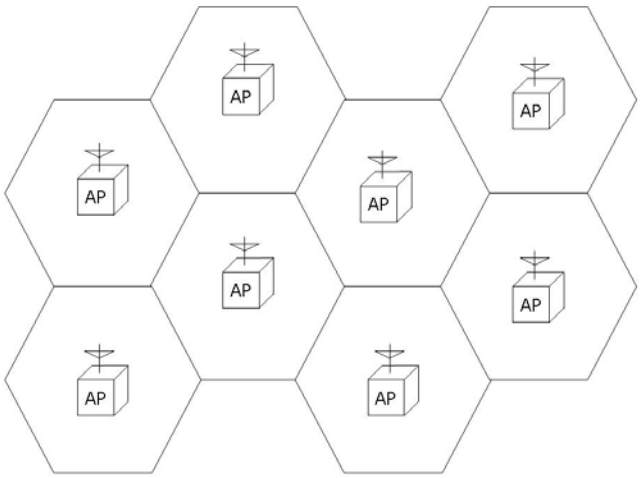


Fig 5.1 – Contesto esagoni

Se proviamo a tracciare una circonferenza con centro sull'AP e raggio che parte dal centro ed arriva ad uno dei vertici dell'esagono ci accorgiamo che le aree degli AP, idealmente si, sovrappongono al massimo due alla volta.

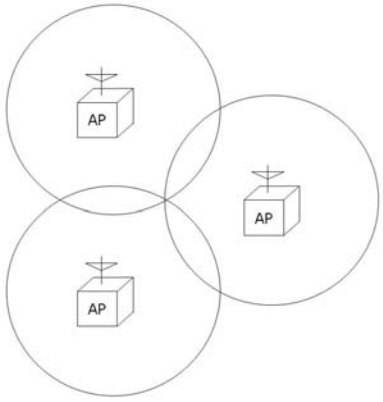


Fig5.2 – Sovrapposizione aree AP in contesto esagoni

Ovviamente questo avviene solo idealmente e nella realtà dei fatti ci sono punti coperti da più di due AP. Questo comunque è il contesto più utilizzato e quindi si è utilizzato una situazione simile per simulare la applicazioni.
Si è quindi configurata una stanza stretta e lunghissima con gli AP disposti ad esagoni. Quello che si voleva simulare era essenzialmente il passaggio da un AP all'altro in maniera continua per far si che la simulazione fosse valida ma il più breve possibile. Si è quindi cercato di riproporre situazioni simili affrontati in modi diversi.
Il primo passo è stato quello quindi di creare una situazione che permettesse ad un utente di non dover mai fermarsi per tornare indietro per semplicità (ai fini della simulazione sarebbe stato analogo ma comportava più difficoltà nella gestione della traiettoria).
Si è poi scelto di utilizzare una traiettoria che congiungesse due punti scelti a caso uno all'inizio ed uno alla fine del corridoio. La traiettoria è stata ideata in modo che non segua una vera e propria retta ma che abbia piccole variazioni costanti. Anche al velocità dell'utente non costante ma variabile. In questo modo le traiettorie erano spesso diagonali rispetto al corridoio anche se non in maniera rettilinea. Questo comportava l'attraversamento di aree simile con modalità e velocità diverse ogni volta.
Come spiegato il simulatore tiene in memoria parte della storia passata per predire quella futura: se il moto avviene sempre in un'unica direzione è ovviamente più facile prevedere lo spostamento. Per questo, in maniera casuale, l'utente viene fatto fermare per qualche secondo in modo da cancellare la storia passata relativa allo spostamento.
Questo ha generato molta più casualità nel moto dell'utente e ovviamente un peggioramento delle prestazione che sono poi state migliorate durante le prove.

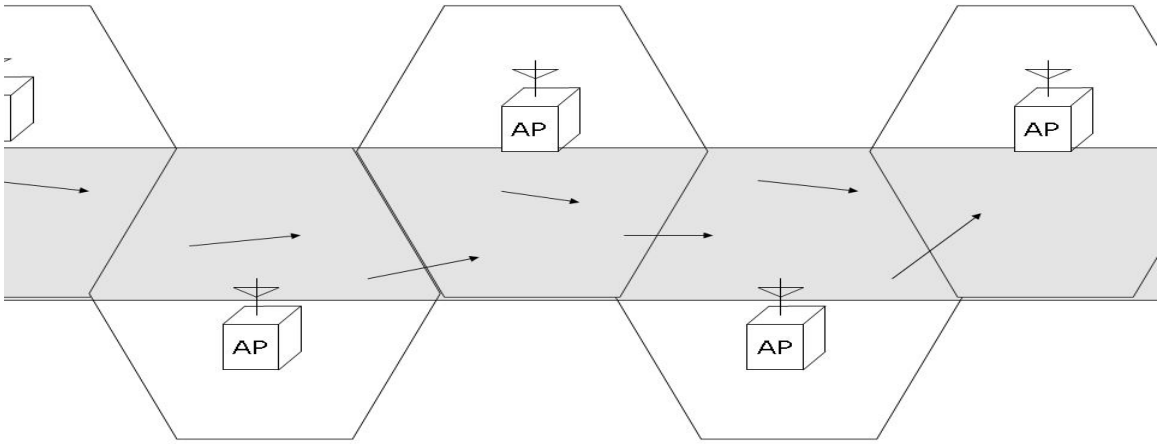


Fig 5.3 – Il corridoio ad esagoni

5.1.2 - Traiettorie

Dopo aver scelto il contesto in cui effettuare la simulazione è anche necessario decidere in che modo far muovere l'utente.
Per questo il simulatore prevede una serie di traiettorie che simulano lo spostamento dell'utente all'interno del contesto creato per la simulazione.
Le traiettorie sono oggetti passivi che vengono interrogati di volta in volta per recuperare la posizione successiva dell'utente.

Caratteristica comune di ogni traiettoria è quella di avere impostata una velocità: non essendo oggetti attivi la velocità deve essere calcolata in base alla frequenza delle richieste sull'oggetto: se ad esempio sappiamo che il simulatore interrogherà le traiettorie ogni secondo allora regoleremo la velocità in base a questa informazione.

Sono state create diversi tipi di traiettoria per avere più varietà si simulazione.

- Rettilinea: impostando un punto di partenza ed uno di arrivo questa traiettoria simula un semplice spostamento rettilineo da un punto all'altro
- Gaussiana: simile ad una traiettoria rettilinea tranne per il fatto che l'utente devia dal percorso più breve tra due punti deviando di un angolo che viene calcolato seguendo una curva Gaussiana
- Continua: in questo caso è sufficiente specificare le dimensioni del contesto e l'utente descriverà delle traiettorie seguendo una variazione dell'angolo di tipo Gaussiano; rispetto alle precedenti ha il vantaggio di non fermarsi ma di continuare in maniera continua tornando indietro quando incontra un muro.
- Traiettoria da punto a punto casuale: simile alle precedenti ma con alcuni accorgimenti per tentare di avvicinarsi al movimento umano. Si è tenuto conto che mediamente le persone no si muovono completamente a caso, specie con un

portatile in mano, ma hanno una precisa direzione od intenzioni. Per questo è possibile fissare un punto di partenza ed uno di arrivo. La simulazione farà si che "mediamente" l'utente segua una traiettoria rettilinea ma con un angolo di deviazione variabile piuttosto piccolo in modo da non ottenere ne un automa perfetto ne una persona che rimbalza da una parte all'altra di una stanza. Inoltre è stato previsto che l'utente possa fermarsi per qualche secondo restando praticamente in un punto. Questa opzione è stata pensata in virtù del fatto che il predittore mediamente tiene conto della storia passata degli ultimi dieci secondi. Pertanto una sosta di qualche secondo è in grado di fargli cancellare la storia passata.

5.1.3 - La visualizzazione

Il simulatore permette di visualizzare il contesto e gli utenti che si muovono all'interno di esso per dare un effetti visivo di quello che sta succedendo. La visualizzazione è molto semplice: si tratta di una finestra dallo sfondo bianco in cui vendono visualizzati gli AP ed i Client in movimento. Ogni utente possiede un coloro diverso ed è collegato graficamente con una retta all'AP a cui è virtualmente collegato.

5.1.4 - La simulazione

Una volta impostati il contesto e la traiettoria da seguire la simulazione parte e comincia a calcolare posizione e potenza dei segnali per ogni utente.
Non c'è un numero massimo di utenti previsto in quanto dipende dalla potenza della macchina stessa.
L'unico vincolo è che la computazione di un singolo passo duri meno della durata prevista del passo stesso: se ad esempio impostiamo la simulazione per far avanzare gli utenti ogni secondo ma i calcoli previsti impegnano la macchina per un tempo maggiore possono sorgere dei problemi.
Alcune applicazioni sono esenti da questo tipo di problema e semplicemente il tempo di simulazione risulterà maggiore al tempo misurato (o minore, nel caso opposto). Altre applicazioni, come quelle real-time, necessitano invece che questi tempi siano rispettati.

Il simulatore parte estrapolando dal contesto i dati relativi agli AP presenti. Dopodichè, ad ogni ciclo, il simulatore calcola, per ogni utente, i segnali che riceve e la potenza di ogni segnale oltre all'AP al quale è connesso.

I dati relativi ad un AP vengono passati attraverso una struttura dati chiamata "LocationOss" che contiene una stringa che identifica il MAC dell'AP ed un valore che indica la potenza del segnale in db. Va fatto notare che il valore degli RSSI è quello reale cambiato di segno ovvero più il valore è alto più in realtà è basso.

Questa scelta è stata fatta poiché il contrario avrebbe complicato l'uso di strumenti già esistenti ma è utile sottolinearla in quanto tutte le soglie sono espresse tenendo conto di questo fatto, per cui un valore pari a 72 è peggiore di uno pari a 64.

Visto che anche tutti i dati e le soglie del predittore sono espresse in questo modo è utile sottolinearlo per non creare confusione in chi legge il codice e tenta di comprenderne il funzionamento.

5.1.5 - Trasmissione dati

Ad ogni ciclo vengono calcolati i valori degli RSSI di ogni AP presente relativamente ad ogni utente considerando anche una soglia oltre la quale l'AP non viene considerato visibile. Quindi tutti i valori raccolti vengono inseriti in una struttura a Vettore e preparata per essere passata al livello soprastante.
Inizialmente il Simulatore è configurato per tentare la connessione con un processo ricevente per spedire i dati ad ogni ciclo di simulazione.
La trasmissione dei dati avviene attraverso una Socket TCP che viene tenuta attiva durante tutta la durata della simulazione.
L'altra scelta era quella di creare un'area dati comune tra i due processi dove il Thread soprastante andasse a leggere i dati.
Si è invece optato per questa soluzione poiché utilizzare una Socket TCP all'interno della stessa macchina non è poi così diverso dal creare un'area dati comune e consente di tenere i due processi separati in modo da far computare la simulazione ad una macchina specifica ed eseguire la simulazione dei programmi su altre macchine all'interno della stessa rete.
Facendo l'ipotesi piuttosto realistica di utilizzare macchine collegate in rete locale, la trasmissione dati, vista anche la ridotta entità della stessa, avviene con una velocità tale da non essere vincolante per le prestazioni, mentre avviare Simulatore ed i processi Client su una macchina può risultare pesante per la computazione.

5.1.6 - Simulatore come scheda Wireless

Usato in questo modo il Simulatore ricostruisce l'ambiente circostante in maniera pulita, fornendoci solo i dati che i nostri sensori capterebbero in un'area coperta da una rete Wireless.
Il Simulatore prevede inoltre la possibilità di inglobare la scelta di una scheda Wireless sull'AP a cui essere connessa.
In pratica, ad ogni ciclo, valutando la potenza dei segnali ricevuti da ogni Client e la politica della scheda del Client stesso, il Simulatore decide a quale AP l'utente sia collegato in quel momento.
Le due possibili politiche sono state configurate prendendo in esame due schede diverse tra loro: una, la Cisco Aironet 350, che segue una politica Soft Proactive e quindi ha tempi di handoff più lunghi, e la Orinoco Gold che invece segue politiche Hard Proactive e quindi ha durate di handoff più brevi.

I valori in base a cui il Simulatore prende decisioni sono stati configurati seguendo le indicazioni sperimentalmente raccolte con queste due schede.
Questa modalità non simula esattamente l'handoff in quanto ci dice unicamente, istante per istante, a quale AP siamo "logicamente" collegati. Questo vuol dire che nella simulazione non si tiene conto del tempo di riconnessione ma si esprime una valutazione sull'AP a cui la scheda decide di essere connessa in quell'istante. Il programma che raccoglie i dati dal simulatore dovrà quindi comprendere il cambiamento di AP e a quel punto valutare in maniera indipendente l'handoff.

E' chiaro che in questo modo la simulazione può darci indicazione solo su due tipi di schede e che quindi può risultare limitante. Comprendendo però le due politiche principali di gestione dell'handoff e ricordando che l'effetto principalmente visibile a livello applicativo è la durata dell'handoff, il Simulatore permette di valutare in maniera credibile la correttezza di una soluzione e di fare stime sensate su come caratterizzare la soluzione stessa a livello di parametri.

5.2 – Simulazione schede Wi-Fi

5.2.1 - Principi guida

Il Simulatore, per come è stato presentato nel capitolo precedente, porta con se alcuni svantaggi evidenti che sono stati risolti introducendo strati intermedi.
La Simulazione, come visto, procede a cicli scanditi da un clock prefissato, così i valori dei risultati della simulazione cambiano a scadenze precise.
E' quindi logico pensare che siano questi eventi, il cambio dei valori, a risvegliare i processi che poggiano sulla simulazione per attivare il meccanismo atto a simulare l'handoff.
Inoltre il Simulatore ad ogni ciclo computa i valori per tutti i client presenti, ognuno dei quali però non deve essere consapevole di essere all'interno di una simulazione, ma deve vedere il mondo come se dovesse interfacciarsi ad una vera scheda Wireless.
La soluzione ideata è quella di uno strato intermedio che fosse a conoscenza sia delle caratteristiche del simulatore sia delle richieste della Client Stub.
Proprio perché si inserisce tra la Client Stub ed il Simulatore questa classe è stata chiamata Simulator Stub.

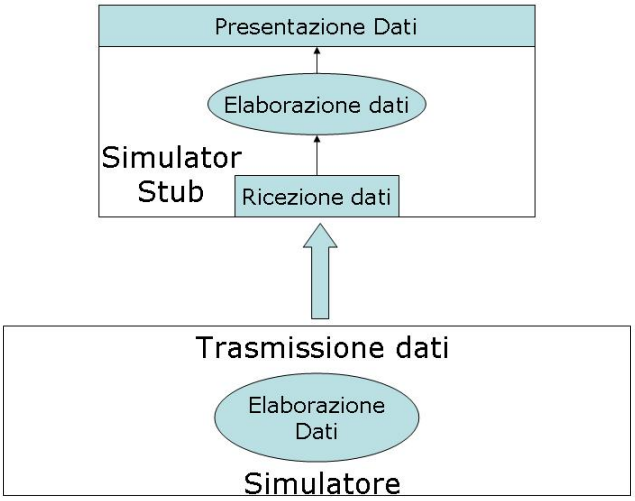


Fig 5.4 – Comunicazione Simulatore – Server Stub

La struttura della simulazione è stata pensata affinché fosse decentralizzata rispetto all'esecuzione delle applicazioni Client: solitamente la simulazione avviene localmente con l'applicazione da testare. Questo ovviamente limita la potenza della simulazione stessa. Il nostro obiettivo era di ottenere la Simulazione scomposta in più parti che potessero essere ospitate su macchine diverse e che potessero comunicare tra loro. Questo è stato ottenuto inserendo la Simulator Stub che funge da canali di comunicazione tra la simulazione dell'ambiente wireless, che quindi può risiedere in una macchina a parte, e i componenti che emulano le schede wireless che invece risiedono nella macchina Client.

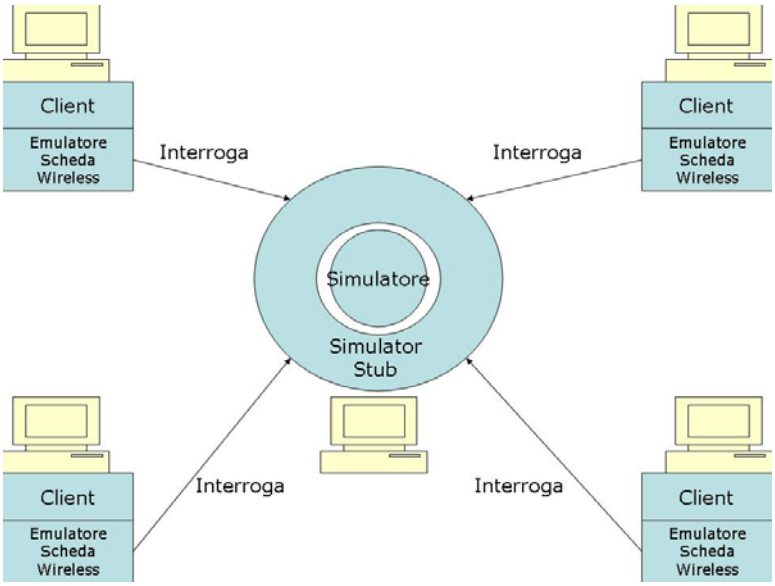


Fig 5.5 – Simulazione in LAN

Come mostrato in figura 5.5, la simulazione può così essere divisa tra computer presenti all'interno della stessa LAN che assicura tempi di comunicazione molto veloci e che quindi non rappresentano certo un limite alle prestazioni della simulazione.

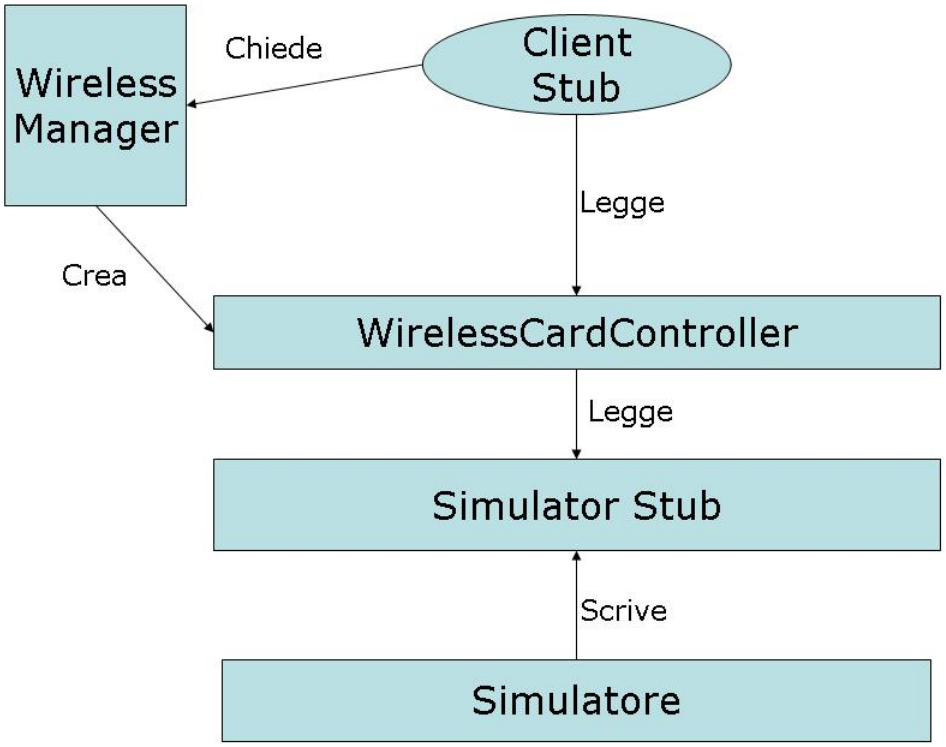


Fig 5.6 – Creazione WirelessCardController

5.2.2 - WirelessCardController

Al fine di svincolare completamente il programma Client dalla presenza di una simulazione, sono state pensate una serie di realizzazioni che possono presentarsi con la stessa interfaccia e reagire in maniera simile sia in presenza di una reale connessione, sia durante una simulazione.
E' impensabile che la Client Stub possa essere predisposta per lavorare con qualsiasi scheda venga montata su un portatile o palmare, e quindi è conveniente che essa dialoghi con un processo che presenta determinate funzioni sempre presenti e che ritornino sempre valori di un certo tipo.

Il WirelessCardController è stato pensato appunto per questo scopo ed è in realtà un'interfaccia che presenta alcune semplici funzioni:

- isReady: questa funzione ha il compito di comunicare la presenza di una connessione ad internet o meno: in questo modo il programma può esentarsi dal controllare lo stato di connessione del PC in cui risiede, cosa che sarebbe ardua simulare esternamente.

- getCardKind: come si evince dal nome, il WirelessCardController ha anche il compito di fornire indicazioni su quale scheda sia stata montata e fornire tali indicazioni alla Client Stub che così può regolarsi di conseguenza.
- isClosed: indica semplicemente che la connessione per qualche motivo è stata interrotta in maniera definitiva o non ripristinabile in tempi utili per l'applicazione.
- getLocation: ritorna l'AP a cui la scheda è attualmente connessa. Anche mentre la scheda sta gestendo un handoff, se interrogata, risponde fornendo l'ultimo AP a cui è stata collegata.
- getLocationsVector: fornisce la lista degli AP visibili con i relativi valori di RSSI. Va detto che questa funzione può comportarsi in maniera differente a seconda delle varie schede: alcune schede effettuano uno scanning dei segnali presenti periodicamente e quindi potrebbero scegliere di non effettuare un nuovo scanning a questa richiesta ma passare semplicemente il valore presente. Altre schede invece vengono risvegliate da questa richiesta non essendo attive in tal senso: anche in questo caso la scheda può scegliere di passare l'ultimo valore ottenuto oppure attendere la terminazione dell'operazione di probing che però può durare qualche secondo.
  Anche in questo caso la diversità tra le varie schede giustifica la presenza di un'interfaccia comune che la Client Stub deve interrogare al fine di ottenere i dati svincolandosi dalla conoscenza intrinseca dell'Hardware.

5.2.3 – Wireless Manager

Questo componente ha un'unica funzione statica ed ha il compito di fornire alla Client Stub un WirellessCardManager.
Sebbene apparentemente semplice è la classe che è a conoscenza della reale situazione in cui si trova il sistema, ovvero se è in atto una simulazione oppure una connessione Wireless vera e propria. A seconda di come viene impostato il Wireless Manager sceglierà quale tipo di WirelessCardManager passare alla Client Stub che quindi ignorerà completamente su cosa stia poggiando.

5.2.4 - Simulator Stub

La Simulator Stub è il componente incaricato di ricevere informazioni dal Simulatore e di ricevere interrogazioni da parte di altri elementi. Logicamente si colloca tra il WirelessCardController ed il simulatore e, per come è costruita, potrebbe benissimo risiedere su una macchina diversa da quella del Simulatore e da quella del Client.
La Simulator Stub è in realtà composta da diversi componenti ognuno con un compito specifico. L'intera classe ha comunque il compito di ricevere le informazioni da parte del Simulatore e offrire una struttura che risponde alle interrogazioni svincolando così gli eventi del Simulatore da quelli del Client.

Il nucleo principale della Classe è quindi costituito da due Server: il primo si mette in ascolto e riceve i dati passati dal Simulatore, il secondo invece accetta le richiesta da parte di un particolare tipo di WirelessCardController che vedremo in seguito.
Queste due classi condividono le aree dati dove vengono registrati i dati passati dal Simulatore.
Il secondo Server riceve, come detto, le richieste di connessione da parte di un tipo particolare di WirelessCardController.
Ad ogni connessione viene attivato un processo in grado di gestire la comunicazione con il CardController; ad ognuno di questi processi viene passata una parte dell'area dati in cui vengono scritti le informazioni passate dal Simulatore e rispondono unicamente a tre richieste:

- Il tipo di Card simulata
- MAC ed RSSI dell'AP a cui si è attualmente collegati
- Lista di MAC e RSSI degli AP attualmente in vista dell'utente.

5.2.5 - SimWirelessController

E' una classe privata del Wireless Manager ed implementa l'interfaccia WirelessCardController. Il Wireless Manager, si è predisposto per la simulazione, alla richiesta della Client Stub crea un SimWirelessController e lo passa come fosse un WirelessCardController.
All'avvio il Controller comunica con il Simulator Stub per poter richiedere i dati dalla simulazione.
Per prima cosa il Controller chiede il tipo di Card e tenta e l'AP a cui è connesso nella simulazione. Dopodichè si dichiara attivo, e logicamente corrisponde all'individuazione di una connessione Wireless.
Ciclicamente chiede al Simulator Stub le informazioni sugli AP visibili e sull'AP con attualmente è stabilita la connessione. Queste informazioni vengono memorizzate all'interno del controller per poter essere lette dalla Client Stub. Le letture della Client Stub sono svincolate dalle letture che il Controller opera sul Simulator Stub. Questo simula il fatto che le schede Wireless facciano probing in maniera indipendente e che forniscano gli ultimi dati raccolti ad una richiesta del Client. Quando il Controller si accorge del cambiamento di AP fa partire un'altra entità che ha il compito di interrompere la comunicazione del Client con il Proxy
Ogni volta che avviene un handoff il Controller richiama una funzione interna che, in base al tipo di scheda, calcola una durata composta da una tempo fisso più un tempo variabile.
In questo modo anche la durata dell'handoff subisce delle variazione aleatorie simulando così la realtà dei fatti. L'incaricato di interrompere la connessione è un componente separato risvegliato solo all'occorrenza e che ogni volta viene informato della durata dell'handoff.
Questo perché in una situazione reale le richieste di scanning e la gestione dell'handoff vengono gestiti separatamente l'uno dall'altro, ovvero la Client Stub è in grado di richiedere informazioni sugli AP visibili anche durante un handoff ed è quindi utile, allo

scopo di provare un'applicazione, che anche nel contesto della simulazione il comportamento sia analogo.
In questo modo l'applicazione Client può essere praticamente sviluppata senza tener conto di una possibile simulazione.
.

5.3 – Socket Wrapper

Lo scopo dell'emulazione è quello di fornire al progettista uno strumento di sviluppo che gli permetta nel contempo ti testare la validità delle sue applicazioni senza doverle provare in un contesto reale ma di progettarle senza aggiungere parti relative alla simulazione stessa, progettarle cioè nello stesso modo in cui lo farebbe se non esistesse il simulatore.
Abbiamo già visto che a questo scopo sono stati previsti una serie di strumenti per calcolare il mondo circostante e fornire le informazioni all'applicazione in maniera totalmente trasparente alla simulazione.
Rimane però il problema che in queste applicazioni è sempre centrale l'utilizzo della connessione ad una rete di PC, che possa essere Internet oppure una semplice rete locale.
Inoltre spesso le applicazione sfruttano componenti pre-esistenti, componenti in cui potrebbe non essere possibile inserire correzioni opportune che tengano conto della realtà simulata e, anche qualora fosse possibile, risulterebbe un'operazione piuttosto scomoda.
Si è quindi pensato di "ingannare" il sistema in maniera tale da poter controllare la connessione di ogni applicazione Java senza che questa potesse esserne consapevole.
Questo ha condotto alla creazione di una classe che inglobasse le Socket originali Java e che mettesse a disposizione tutte le funzioni più alcune aggiuntive volute per il progetto.

5.3.1 - Socket Wrapper

Per spiegare questo meccanismo è necessario spiegare brevemente come Java fornisca certi strumenti per estendere la Java Virtul Machine solitamente distribuita.
In particolare le classi Socket e DatagramSocket di Java (Socket di tipo TCP e UDP rispettivamente)comprendono entrambe due riferimenti statici, uno ad una SocketImpl e l'altro ad una SocketImplFactory che è un puntatore statico(per le DatagramSocket si aggiunge il prefisso Datagram che in seguito ometterò per brevità). Ogni volta che creiamo una Socket il costruttore della classe invoca una funzione di SocketImpl per ottenere un oggetto che poi rappresenta il nucleo della Socket stessa.
La classe Socket comprende quindi un nucleo formato da un'altra classe e che, quasi in tutto, costituisce la vera socket, ovvero il componente che è in grado di comunicare con il mondo esterno.
I progettisti di Java hanno però previsto che tale classe potesse essere sostituita con un'altra che svolga le stesse funzioni.
Ogni volta che si invoca la costruzione di una Socket questa controlla se ha già disponibile una SocketImpl. Nel caso questo non sia vero provvede a caricarne una

chiedendola ad una classe statica chiamata SocketImplFactory. Nel caso anche questa non sia già stata settata la classe provvede a caricarla utilizzando un Class Loader.

Una volta ottenuta una SocketImpl la classe Socket la utilizzerà per tutto il resto della sessione.
La classi Socket e DatagramSocket prevedono che possa essere configurata esternamente la SocketImplFactory. Come abbiamo detto questa classe viene interpellata unicamente una volta per fornire una SocketImpl, ed è quindi necessario che la prima operazione che il programma deve effettuare è sostituire la SocketImplFactory con una appositamente creata per far si che tutte le Socket create dal programma che richiedono la SocketImpl la richiedano alla classe modificate e non all'originale.
La prima istruzione del programma, qualora volessimo simulare la sua presenza in una rete Wireless, dovrà essere l'istruzione in cui modifica il valore statico della SocketImplFactory nella classe Socket.
Va fatto notare che, sebbene la SocketImpl sia un valore denominato "Statico" la sua variazione riguarda unicamente il processo che ha invocato tale funzione: ovvero, se facessimo partire due processi che eseguono lo stesso codice a meno di quella istruzione di partenza, l'uno chiamerebbe le Socket modificate mentre l'altro processo continuerebbe ad usare le normali Socket .

Faccio notare che questa singola riga di codice è l'unica variazione che tiene conto della presenza di una simulazione e che va messa sempre all'inizio del programma (o meglio, va messa prima della creazione di una qualsiasi Socket logicamente, ma è più semplice sicuro e pratico aggiungerla come prima riga del codice). Se proprio non si vuole inserire alcuna riga di codice si può sempre scrivere un breve programma che abbia come unico scopo quello di modificare il contesto inserendo le nostre classi aggiuntive, e poi lanciare l'applicazione vera e propria.

Il progettista può quindi, a ragione, scrivere l'applicazione senza tener conto del fatto che possa essere simulata in quanto il sistema prevede che tutto ciò che lui vede come esterno possa essere simulato, dalla presenza in una rete Wireless alla perdita di pacchetti durante l'handoff.
Per avere il controllo sulle Socket create dall'applicazione Client (in questo caso) sono state scritte due classi che svolgono il ruolo di Wrapper , ovvero incarto o involucro, per le Socket stesse in quanto non sono altro che un classe "guscio" che contiene all'interno una vera Socket e, salvo alcune eccezioni, demandano le richieste esterne alla vera Socket contenuta.

5.3.2 - HackDatagramSocketImpl

E' la classe che estende la DatagramSocketImpl, il prefisso "Hack" è stato messo per sottolineare il fatto che in una certa misura ci si è intromessi nel funzionamento standard del sistema andando a prendere il controllo di ciò che normalmente è lasciato alle librerie Java.

DatagramScoketImpl è in realtà una classe astratta proprio perché potesse essere sostituita da altre classi come si è scelto di fare per la simulazione.
Al momento della creazione il costruttore di tale classe crea una "PlainDatagramSocketImpl" che è la classe Java che estende ed implementa la classe astratta DatagramSocketImpl ed utilizzerà tale classe per soddisfare le richieste provenienti dall'esterno.
La classe "Hack" è in realtà piuttosto semplice poiché, una volta creata la vera Socket, non fa altro che presentare un'interfaccia del tutto identica con in più una funzione che ha il solo scopo di configurare un valore interno.
Questo valore è di tipo booleano e governa il comportamento delle "receive" (la funzione che mette la Socket in attesa di ricevere un pacchetto) e delle "send"(funzione che invia un pacchetto).
In pratica si è fatto in modo di "intercettare" i pacchetti, ovvero di rendere nulle le funzioni richieste quando configuriamo le socket in un certo modo: durante le send semplicemente si raccoglie il pacchetto ma non lo si inoltra al destinatario, ritornando immediatamente al chiamante, mentre le receive entrano in un ciclo in cui continuano a ricevere i pacchetti senza però che la funzione torni al chiamante.
Quindi, nel momento in cui decidiamo di riportare le condizioni alla normalità, la receive esce dal suo ciclo infinito e ritorna ma solo con l'ultimo pacchetto ricevuto, simulando il comportamento durante un handoff durante il quale la Socket non è consapevole di essere sconnessa.

L'altra particolarità è che ad ogni creazione la HackDatagramSocketImpl segnala la propria esistenza alla HackDatagramSocketImplFactory assicurando così la possibilità di poter essere bloccata esternamente.

5.3.3 - HackDatagramSocketImplFactory

Questa classe implementa l'interfaccia DatagramSocketImlFactory e quindi è pensata per sostituirla durante la simulazione.
L'interfaccia prevede un'unica funzione chiamata "createDatagramSocketImpl" che, come si evince dal nome, ha il compito di fornire una DatagramSocketImpl.
Ovviamente in questo caso questa funzione ritorna una HackDatagramSocketImpl consentendoci di avere quel controllo di cui si parlava.
Diversamente dalla classe originale ora però questa funzione ha il compito di fornire alcuni strumenti di monitoraggio per consentire l'interruzione, anche mirata, delle Socket create.

Ogni istanza della classe DatagramSocketImpl, alla sua creazione, comunica la sue esistenza alla Factory che quindi ne tiene traccia tra le Socket attive in un vettore interno.
Analogamente, quando la Socket viene chiusa, lo segnala alla Factory che provvede a cancellarlo dal suo elenco.
La factory, possedendo delle funzioni statiche, può essere invocata da chiunque senza possedere un riferimento.

Due sono i metodi principali riconducibili alla simulazione: closeAll e openAll che hanno la capacità, rispettivamente, di interrompere tutte le connessioni esistenti e di riaprirle.
Originariamente la classe è stata pensata per essere monitor sulle connessioni e quindi possiede alcuni strumenti di controllo e supervisioni, consentendo anche l'interruzione di una singola Socket e la riapertura della stessa.
Poi, durante lo sviluppo, è stato chiaro che, per il nostro tipo di simulazione, aveva più senso una funzione che chiudesse ogni connessione aperta dall'applicazione: simulando un processo su una macchina collegata in una rete tramite dispositivo Wireless, nel momento in cui la scheda perde la connessione tutte le Socket aperte rimangono al buio per tutta la durata dell'handoff e quindi ha poca utilità, limitatamente a questo caso, operare una chiusura selettiva.

5.3.4 - Il Bootclassloader

I costruttori di Java ci hanno fornito questi potenti strumenti per governare le risorse del nostro sistema ma si sono in qualche modo tutelati da intrusioni esterne: è infatti possibile sostituirsi nella factory unicamente grazie alla funzione messa a disposizione ed è abbastanza semplice farlo in maniere cosciente.
Il problema per noi era rendere questa operazione il più trasparente possibile all'applicazione e, come visto, lo scopo è stato raggiunto . Sia la classe Socket che le altre due classi in gioco appartengono alla package denominato java.net.
Quasi tutte le funzioni che ogni classe invoca sull'altra sono "protected" ovvero possono essere invocate solo da funzioni appartenenti allo stesso package. Questo implica che le classi create per la simulazione dovranno appartenere allo stesso package denominato java.net.
Java però si protegge dalle intrusioni esterne che non siano in qualche modo espressamente volute: le classi Java vengono caricate attraverso uno strumento chiamato Class Loader.
Il class loader è uno strumento che Java mette a disposizione anche al programmatore ma in realtà esistono tre tipi di class loader: uno incaricato di caricare le classi originali java (quelle che stanno in un package il cui nome incomincia con "java"), uno incaricato di caricare le classi "esterne" ed uno messo a disposizione del programmatore.

Il Bootstrap Loader o loader nativo, come viene a volte chiamato, è l'unico a poter caricare le classi "java" e nessuno può assumere il controllo di tale loader se non la Java Virtual Machine.
L'unico modo quindi per caricare le classi create mettendole all'interno del package java.net è far si che sia il Bootstrap Loader stesso ad occuparsi del loro caricamento.
Ci sono due soluzioni per questo:

- modificare il package originale Java inserendo a mano le due classi aggiuntive: questo non comporta nulla di pericoloso ma rimane comunque una operazione che modifica un package nativo Java ed inoltre è necessario ripetere l'operazione su ogni macchina che ospiti la sumulazione.
- Creare un package denominato java.net ed aggiungerlo al bootclasspath: il Bootstrap Loader cerca seguendo in un percorso chiamato appunto

"bootclasspath". Aggiungendo al percorso l'indicazione della località della nostra libreria e chiamando il package java.net il Loadaer nativo provvederà a cercare e caricare la classe nel momento della creazione di una sua istanza.
Per fare questo è sufficiente aggiungere al comando "java" l'opzione "-Xbootclasspath/a:<nome del jar>.

5.4 – Il Log Runner.

La simulazione così ottenuta è di grande importanza perché permette di provare senza perdite di tempo ed in maniera semplice la validità di una soluzione.
Inoltre, cose che si è rivelata molto utile nella stesura della tesi, permette di fare test con un grande numero di Client ed automatizzare la ripetizione di questi test consentendo così di cercare la soluzione numerica anche per via empirica.
Rimane però il limite della correttezza formale del Simulatore stesso che, attualmente, non è in grado di riprodurre in maniera esatta la realtà delle situazioni. Inoltre, come detto, tutto si basa sulle situazioni che noi possiamo configurare con gli strumenti messi a disposizione per creare nuovi ambienti.
Sarebbe invece interessante avere a disposizione un modo per poter verificare il comportamento di alcune applicazioni anche in una vera rete Wireless.
La soluzione, oltre quella banale di prendere il portatile in mano e farsi un giro all'interno di una rete Wireless, può essere cercata nella registrazione di percorsi fatti all'interno di una rete stessa.
La situazione può essere paragonata a quella di un pilota che, per imparare un tracciato e fare esperienza delle curve presenti, passa del tempo in alcuni simulatori molto simili a Videogiochi molto sofisticati.
Per fare queste simulazione i progettisti registrano filmati dei giri fatti dalle macchine attraverso telecamere montate sulle vetture e poi riproducono la stessa visuale all'interno della simulazione.
Ovviamente l'idea non è nulla di così complesso, ma il concetto è praticamente identico: si tratta infatti di capire cosa la Client Stub posso percepire del mondo esterno e registrarlo in un percorso reale.

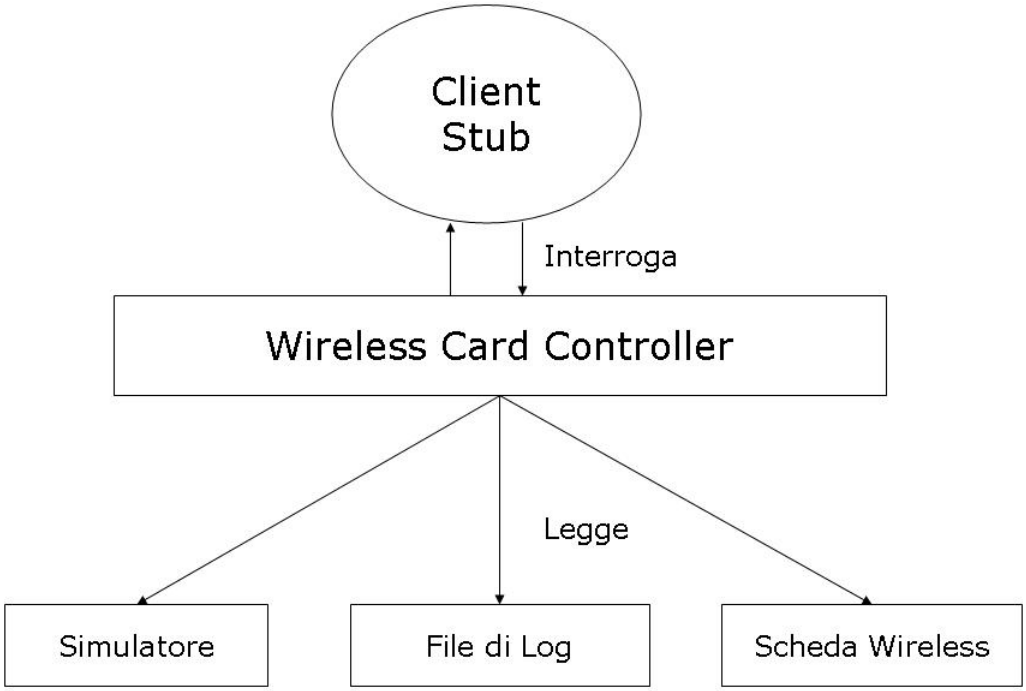


Fig 5.7 – Indipendenza del Client dal contesto simulato

5.4.1 - Il file di Log

Quello che la Client Stub può "vedere" attraverso le sue interrogazioni la scheda di rete sono principalmente i valori degli RSSI degli AP visibili. L'altra cosa che influisce sull'applicazione, ma che è invisibile ad essa, sono i momento di buio provocati dagli handoff.
Un programma su postazione fissa e collegato alla rete spedisce pacchetti UDP contenenti un'informazione numerica (molto banalmente il numero di sequenza) ad una frequenza prefissata di 100 ms, ad un processo che risiede su un portatile o comunque un dispositivo mobile dotato di interfaccia Wireless.
Il processo sul dispositivo mobile ha il compito di ricevere i pacchetti e registrare questi eventi. Oltre a questo ha il compito di richiedere periodicamente, circa una volta al secondo, il probing degli AP presenti e registrarlo.
Il risultato è un file in cui sono presenti tutte le registrazioni degli AP visibili con relative potenze e di tutti i pacchetti presenti. Osservando i pacchetti UDP non pervenuti grazie al numero di sequenza è possibile calcolare il tempo di handoff con uno scarto medio di 100 ms: essendo la frequenza di trasmissione pari ad un pacchetto ogni 100 ms, l'errore medio nel calcolare l'inizio e la fine dell'handoff è appunto di 50 ms.
Utilizzando una scheda Cisco che ha tempi di handoff nell'ordine dei secondi, si è ritenuto sufficiente l'approssimazione adottata.

Chiaramente sarebbe sufficiente aumentare la frequenza di trasmissione per ottenere maggiore precisione.
Nella registrazione dell'evento si tiene ovviamente traccia del tempo di sistema, in modo da temporizzate gli eventi.
In seguito questo file viene rielaborato e si ottiene così un file di Log in cui gli eventi sono registrati riga per riga in questo modo:

- Tipo di evento: probing oppure handoff
- Tempo espresso in millisecondi e relativamente all'inizio della registrazione.
- Descrizione dell'evento: nel caso sia handoff seguono semplicemente un tempo di inizio ed uno di fine. Nel caso di probing seguono il MAC dell'AP a cui è attualmente collegata la scheda e la lista dei MAC degli AP visibili con relativi RSSI.

Es. di file di Log:

```
"probing 93014 00028aa8995d 00028aa8991f -65.0 00028aa89913 -54.0 ...
 probing 94005 00028aa8995d 00028aa89913 -50.0 00028aa8991f -69.0 ...
 handoff 94996 98702
 probing 94996 00028aa8995d 00028aa89913 -60.0 00028aa8990b -75.0 ..."
```

5.4.2 - Log Runner

Questa classe implementa l'interfaccia WirelessCardController e viene pertanto passata alla Client Stub dal Wireless Manager al posto della SimWirelessController.
E' un oggetto attivo (Thread) e pertanto deve essere lanciato. Una volta partito chiede i diritti di lettura sul file di Log. Il Log Runner crea inoltre un altro Thread chiamato Wall che ha il compito di bloccare le Socket e di risvegliarle ed in pratica è il fautore dell'effetto handoff. Come visto i processi di Probing e di gestione dell'handoff possono avvenire separatamente, pertanto i Thread che gestiscono le informazioni relative agli RSSI e l'handoff devono procedere paralleli.
Come possiamo notare anche dall'esempio sopra riportato, l'handoff ed il probing possono essere anche contemporanei. Altra cosa interessante da notare è che nel probing dopo l'handoff, l'AP collegato sia ancora quello precedente, visto che l'handoff ancora non è stato terminato.
La classe in se è piuttosto semplice: ciclicamente effettua il parsing della riga cambiando i valori dell'AP attuale ed di quelli visibili. Ad ogni ciclo comincia la lettura della riga successiva per poter calcolare il tempo in cui deve rimanere inattivo prima di operare il prossimo cambiamento.
Quando incontra una riga che segnala l'handoff semplicemente configura il Wall per la durata prevista ed in seguito lo risveglia al tempo opportuno.

Capitolo 6 – Risultati sperimentali e sviluppi futuri

In quest'ultimo capitolo illustrerò i risultati che si sono avuti sperimentalmente commentandoli per rendere comprensibile la loro analisi.
In primo luogo darò una descrizione di come si sono effettuate le simulazioni, del contesto creato per esse e dei parametri che le caratterizzano. Quindi mostrerò i risultati ottenuti dal lato Client e lato Proxy commentandoli con opportuni grafici che mettano in evidenza i risultati.
Infine cercherò di mettere in luce i possibili sviluppi futuri che si aprono dopo questo lavoro per dare un'idea della direzione verso cui ci si potrebbe muovere.

6.1 – L'ambiente di simulazione

6.1.1 – Struttura della simulazione

La simulazione, a livello di applicazione, è stata pensata per riprodurre una situazione reale. Sia i processi Client che Proxy che Server risiedono sulla stessa macchina e quindi la comunicazione avviene molto velocemente e senza possibilità di perdita dei pacchetti.
Per prima cosa quindi si avviano il Server ed il Proxy che restano in attesa di richieste da parte delle Applicazioni Client. Quindi si avvia il simulatore per creare il contesto ai Client. Le scelte possono essere due: o avviare la simulazione di un contesto scelto in cui poi si faranno muovere diversi Client, oppure far leggere i dati dai file di Log. Ovviamente per avere simulazioni eterogenee servirebbe un alto numero di file di Log tutti diversi tra loro, il che non li rende utili per una simulazione che ha il fine di valutare statisticamente i risultati dove si predilige l'alto numero di prove diverse tra loro. Si è quindi scelto di utilizzare un contesto simulato ed di simulare venti Client per volta. Il limite dei Client non è fisso ma dipende dalla potenza della macchina in cui viene fatta girare la simulazione. Una volta avviati i Client questi effettueranno le loro richieste di stream video ma in ordine casuale, in modo da non avere venti richieste contemporanee ma nemmeno venti richieste equidistanti tra loro. La simulazione prosegue inviando i dati verso i Client e gestendo le loro richieste, registrando i dati delle applicazione Client e Proxy in file per poi poter essere letti.

6.1.2 – Server

Normalmente il Server gestisce le richieste una per volta, preparando un processo leggero (thread) che gestisca ogni richiesta separatamente. Durante la simulazione a noi non interessa valutare le prestazioni del Server poiché non rientra tra i nostri obiettivi, ma le

prestazioni di Client e Proxy. Il Server fa un uno intenso delle CPU che può rallentare l'intera simulazione quando i Client crescono in numero. Per questo motivo il Server è stato modificato facendo si che un solo processo legga e computi i dati video per tutti i Proxy: la prima richiesta viene gestita normalmente, creando un processo che prenda i diritti di lettura del filmato e cominci a processare i dati per poi inviarli. Quando un nuovo Proxy effettua la sua richiesta il Server non crea un nuovo processo ma chiede a quello già creato di spedire i dati ad un gestore che altro non farà che spedirli al Proxy e così fa per ogni richiesta successiva. Viene quindi creato un processo gestore per ogni Proxy richiedente per spedire al lui i frame video e per monitorare lo stato di presenza del Proxy e, nel caso non sia più presente, chiudere la comunicazione verso di lui. In questo modo il processo dei dati avviene una volta sola ad opera di un unico thread in modo che l'utilizzo della CPU non sia fortemente dipendente dal numero di richieste servite.

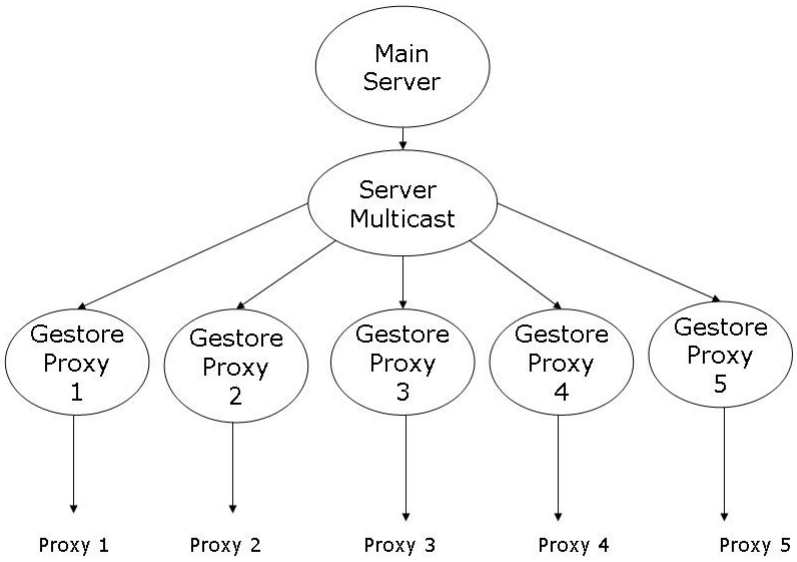


Fig 6.1 – Server Multicast

Questo non altera la validità della simulazione in quanto agli dal lato Proxy non si avverte alcuna differenza: ad una richiesta corrisponde l'erogazione del servizio richiesto. Ovviamente non tutti i Proxy riceveranno il filmato dall'inizio ma questo non ha molta importanza ai fini dei test. Per quanto riguarda le prestazioni dei Proxy non vi è alcuna differenza se le richieste vengono servite da un Server unicast normale oppure uno multicast come quello presentato. Quello che invece otteniamo è una simulazione molto più leggera in quanto la CPU è massimamente impegnata dai processi che decodificano il filmato che in caso di server unicast sono n per n Client mentre col server multicast rimane 1 per n Client.

6.2 – Simulazioni e risultati

6.2.1 – Impostazioni delle simulazioni

Per la simulazione si è utilizzato un QuickTime movie come filmato e si è scelto di impostare come frequenza di riproduzione 8 frame per secondo, quindi un frame ogni 125 ms. Questo non è limitante in quanto non pregiudica la validità delle simulazioni come specificato in seguito
Come evidenziato durante l'analisi dell'implementazione i parametri configurabili all'interno della simulazione sono molti e quindi questo ha richiesto un numero notevole di simulazioni.
Faccio un breve riassunto di questi parametri per poi spiegare come essi sono stati fatti variare.
LATO CLIENT:

- Numero di campionamenti per il Grey model
- Numero di conferme per il cambio stato
- Differenza tra i valori predetti degli RSSI dell'AP attuale e del successivo

LATO PROXY

- Numero di frame conservati in stato di HIGH PROB
- Numero di frame conservati in stato di LOW PROB

Le tipologie di schede simulate sono due, una che utilizza una politica di gestione dell'handoff Hard Proactive ed una che utilizza una politica Soft Proactive.
Nel caso della scheda Hard Proactive si è considerata una durata dell'handoff che potesse variare tra i 440ms ed i 500ms
Nel caso della scheda Soft Proactive si sono considerati handoff della durata variabile tra gli 880 ms ed i 1000 ms.
Questi dati sono stati valutati sperimentando il funzionamento della scheda Cisco (Soft Probative) ed Orinoco (Hard Proactive).
Non tutti questi parametri sono però correlati tra di loro ai fini della prestazione della simulazione. Gli obiettivi da raggiungere sono sostanzialmente due: la corretta predizione dei un handoff da parte del Client e la corretta risposta ad una richiesta di Rewind.
Possiamo notare che i parametri lato Proxy servono essenzialmente al secondo obiettivo mentre i parametri lato Client influenzano unicamente la predizione.
I parametri del Proxy possono essere fissati con una certa precisione una volta che si conosca la frequenza del filmato ed una stima della durata degli handoff della scheda wireless in dotazione al Client che si sta servendo.
Ad esempio, per la scheda Orinoco è stato prevista una durata massima di handoff di 500ms. Visto che la frequenza del filmato, e quindi quella di trasmissione, è di un frame ogni 125 ms con un rapido calcolo si evidenzia che i frame persi sono mediamente 4. A questo dobbiamo aggiungere un certo ritardo che intercorre tra la spedizione del pacchetto ed il ricevimento della richiesta, ritardo che comprende la ricezione da parte del Client, la valutazione e la spedizione della richiesta.

Oltre a questo sappiamo che talvolta un frame può essere diviso in più slot del buffer. Questo implica che la richiesta di 4 frame possa tradursi nella ritrasmissione di più di 4 slot del buffer.
Sperimentalmente si sono fissate le dimensioni delle finestre per il Rewind in questo modo:
SCHEDA CISCO:

- Large window:8 slots
- Small window:5 slots

SCHEDA ORINOCO

- Large window:14 slots
- Small window:5 slots

Chiaramente un diverso filmato cambierebbe queste impostazioni ma non cambierebbe la validità della soluzione. Si è pertanto scelto di effettuare tutte le prove con questo filmato e queste impostazioni di cui si dovrà tenere conto quando si andranno a valutare i risultati numericamente.
Nel Client si è scelto di impostare la dimensione del Buffer a 30 slots: è sicuramente una misura sovrabbondante per questa situazione ed anche qui, per ottimizzare l'utilizzo delle risorse, si potrebbe operare un calcolo in base alla frequenza del filmato ed alla durata stimata dell'handoff in maniera simile a quanto fatto per il Proxy.
L'obiettivo era però, in questo caso, provare la validità della soluzione e riuscire a configurare i parametri della previsione per ottenere dei buoni risultati, e non minimizzare l'utilizzo delle risorse da parte del Client.
Quindi si è preferito tenere un buon margine di sicurezza che ci consentisse di cercare il raggiungimento degli obiettivi senza curarci delle dimensioni del Buffer del Client.

6.2.2 – Comportamento lato Client

Qui di seguito riporterò dei grafici per meglio illustrare meglio il comportamento del Client. Lo scopo lato Client è quello di riuscire ad identificare la perdita di informazioni e gestire la conseguente richiesta

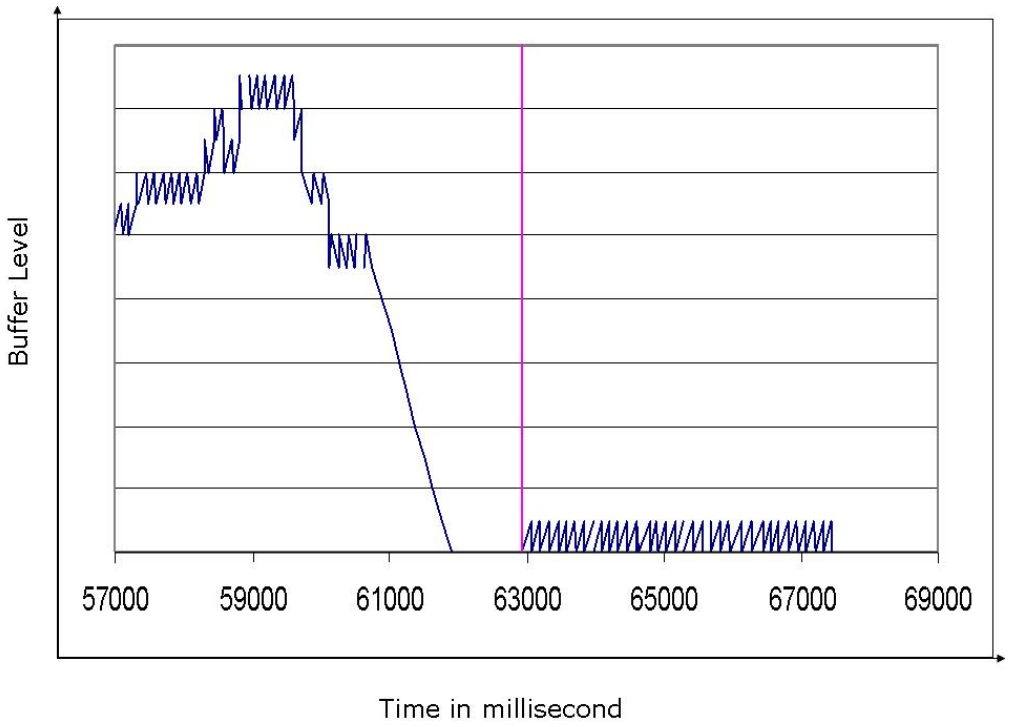


Fig 6.6 – Andamento del Buffer Client senza ritrasmissione dati

In questo grafico viene presentala la situazione senza alcun controllo, ovvero senza nessun meccanismo per il recupero dei dati trasmessi. Nell'asse orizzontale è riportato il tempo in millisecondi mentre in quello verticale il livello di riempimento del buffer del Client. Quello che possiamo osservare da questo è grafico è che il livello del Buffer ha delle leggere fluttuazioni ma rimane mediamente costante. Quando il tempo giunge verso i 61000 ms il livello diminuisce linearmente per l'handoff in corso ed è il segno che il Client sta consumando gli slot del buffer costantemente ma senza che nessuno li sostituisca. Il Client, in questo caso, termina i frame a sua disposizione e quindi attende. La linea verticale a metà del grafico indica il momento in cui il Client ricomincia a ricevere frame, ovvero l'avvenuta riconnessione con l'AP. Come si può notare alla ripresa il grafico assume un andamento a dente di sega che non cresce più di livello poiché la velocità del Proxy è impostata sulla velocità normale del Client. Questa è la situazione prima di questo lavoro ed è evidente il problema in cui un'applicazione di questo tipo può incappare dopo appena un handoff.
Vediamo cosa succede quando introduciamo il meccanismo in grado di riconoscere la perdita dei dati e di effettuare la richiesta al Proxy.

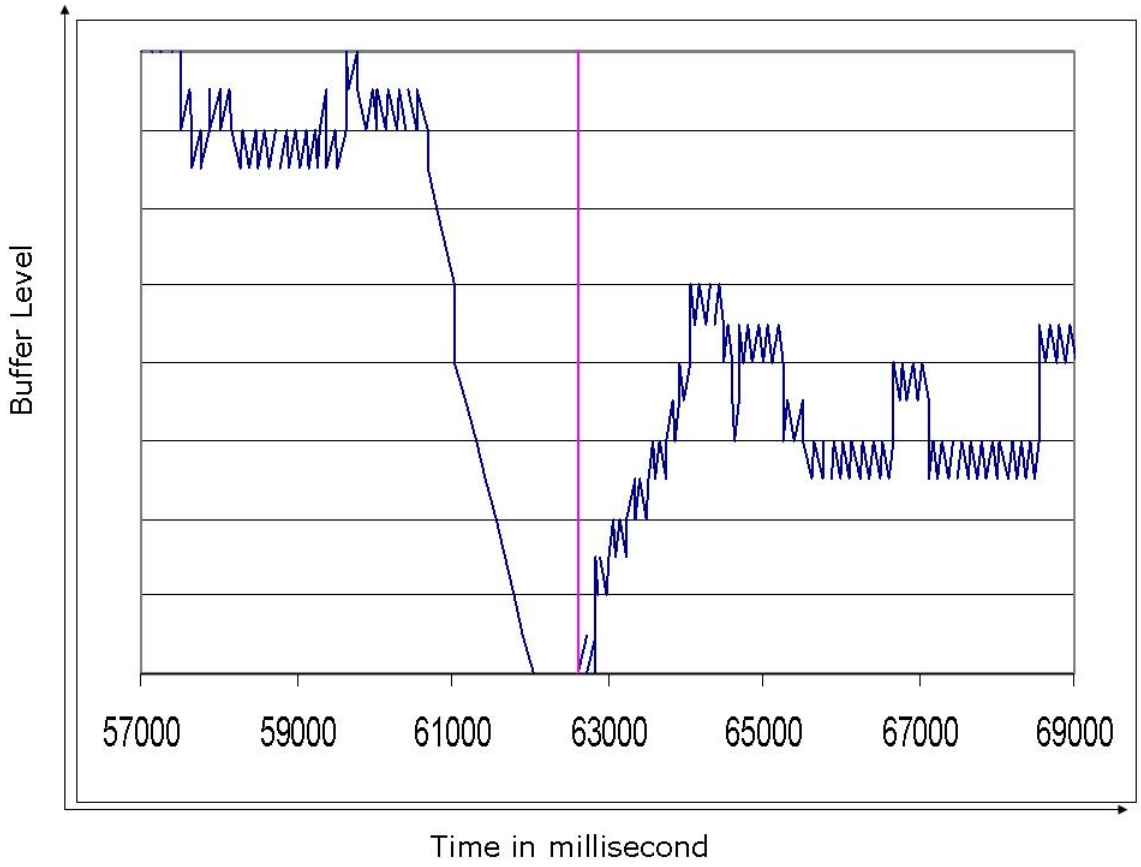


Fig 6.7 – Andamento del buffer del Client con meccanismo di rewind

In questo caso, del tutto simile al precedente come situazione, notiamo dopo essersi accorto della perdita dei dati, il livello del Buffer comincia a ricrescere. Questo perché il Client ha spedito una richiesta al Proxy ed è stato servito. Il Proxy, dopo aver effettuato il Rewind, spedisce i dati più velocemente per un certo numero di frame. Questo causa la salita ripida che si osserva subito dopo l'invio della richiesta (che è segnalata dalla linea verticale). Possiamo notare come il livello del Buffer non venga però ripristinato al suo massimo: questo perché manca il meccanismo che richiede al Proxy un cambio di velocità per sopperire alla mancanza di frame. Il Proxy, alla richieste, effettua un riavvolgimento di un numero n di frame. In seguito aumento la velocità di trasmissione per lo stesso numero n in modo da riempire nuovamente il Buffer del Client. Possiamo notare che anche la salita ripida è a dente di sega, il che significa che mentre riceve il Client consuma (cosa che non si verifica lungo la discesa, dove il Client consuma e basta). Questo fa si che dopo l'operazione di rewind rimanga comunque un certo "buco" nel buffer del Client. E' quindi necessario introdurre un meccanismo che si accorga del livello critico del Buffer e che richieda al server di aumentare la velocità per poi segnalare l'avvenuto ripristino del livello ideale.

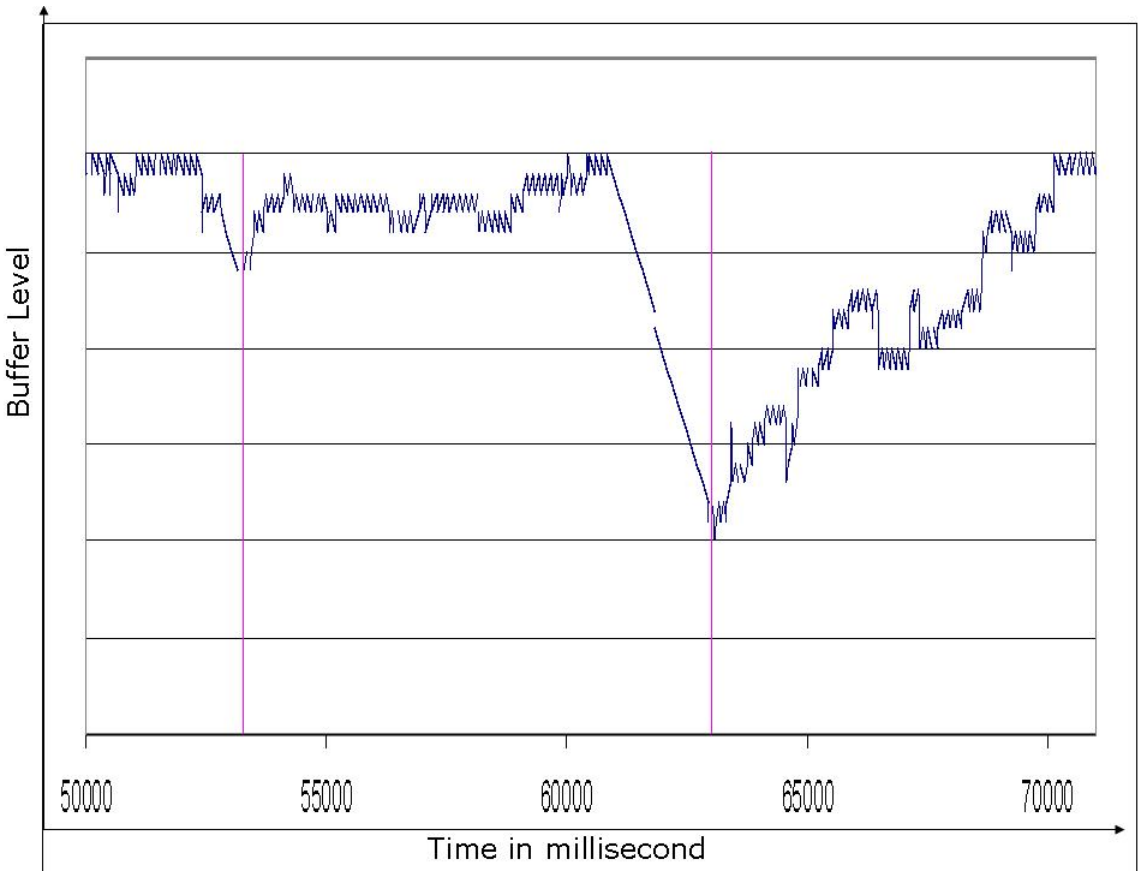


Fig 6.8 – andamento del Client Buffer con meccanismi di rewind e cambio velocità

In questo grafico osserviamo due handoff: il primo, di durata più breve, ed il secondo decisamente più lungo. Notiamo che dopo la richiesta effettuata il Client torna in breve tempo a ripristinare il suo livello ottimale di lavoro. Questo avviene in virtù del controllo sul livello del Buffer e dell'aumento di velocità nella trasmissione da parte del Proxy.
In realtà in questi grafici si mette in luce il buon comportamento del Server: anche se osserviamo cosa accade dal lato Client in realtà il grafico sottolinea come il Proxy è in grado di soddisfare le richieste mentre il merito del Client sta unicamente nell'inviarle.
Non ho differenziato i Client a seconda delle schede perché l'unica differenza risiede, a livello visibile da questa applicazione, nella durata degli handoff.

6.2.3 – Comportamento del Proxy

Osserviamo ora cosa succede dal lato del Proxy.
Visualizzeremo di seguito i grafici del Buffer del Proxy evidenziando quando arriva una richiesta e quando viene soddisfatta.
Illustreremo grafici relativi ai casi di Client con scheda Cisco, scheda Orinoco ed una scheda non conosciuta.
Ricordo brevemente come è composto il buffer del Proxy così da poter meglio spiegare i grafici. Il Proxy utilizza un proxy circolare per conservare i frame da spedir al Client. La

parte del Proxy che unicamente questa utilità, ovvero la parte che contiene i frame scritti ma non ancora spediti verrà chiamato “usage buffer”.
Una parte del buffer invece è destinata a conservare un certo numero dei frame già spediti per permettere il invio dei dati quando il Client lo richiede;questa parte verrà chiamata “rewind window” per sottolineare il suo ruolo.
La somma dei due verrà chiamata “total buffer”in modo da ricordare che è logicamente diviso in più parti.
Prima di vedere le differenze tra i vari casi mettiamo in luce cosa accade nel Proxy quando il Client effettua le richieste.

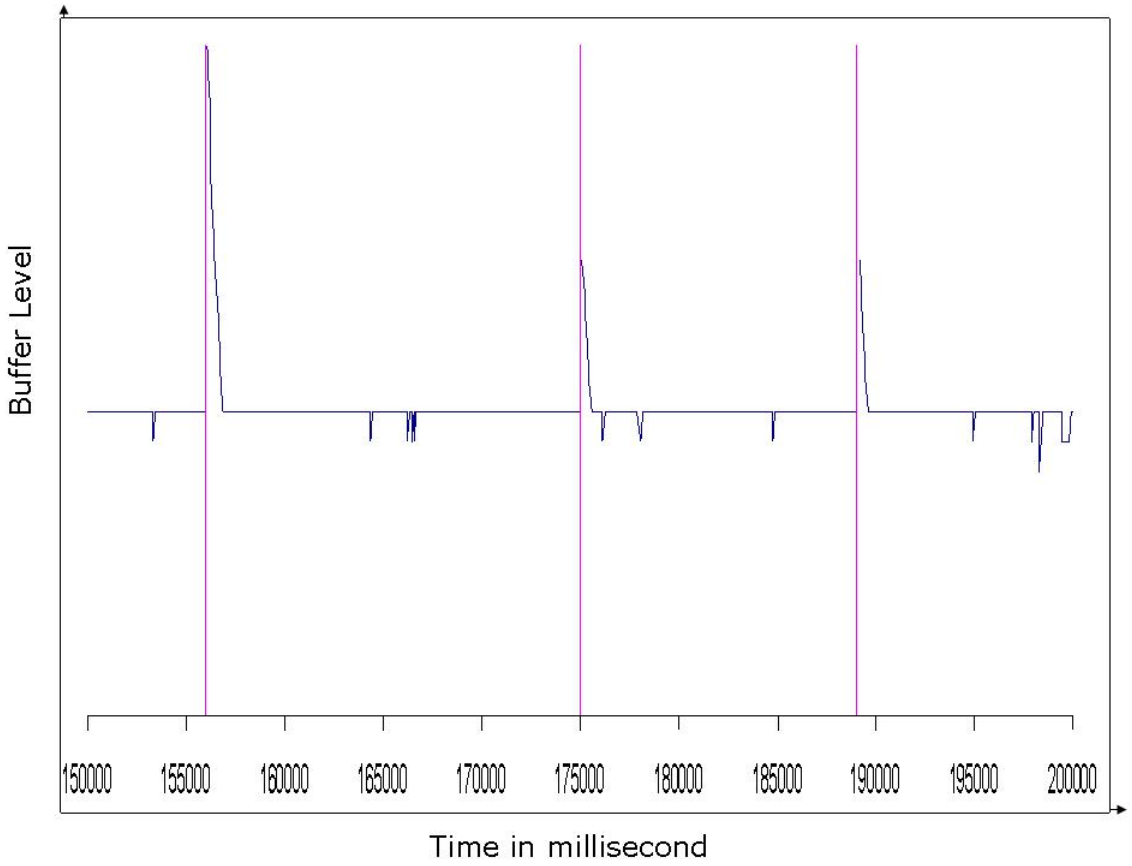


Fig 6.9 – Buffer del Proxy durante le richieste di rewind

In questo grafico è messo in evidenza il livello del usage senza comprendere la rewind window. Le linee verticali indicano l’arrivo delle richieste di Rewind. Notiamo come il livello del buffer sia praticamente costante lungo il corso del normale lavoro mentre cresca verticalmente all’arrivo di una richiesta: questo perché, riavvolgendo il filmato, il Proxy rende di nuovo utilizzabili frame che aveva già consumato con il conseguente rialzo del livello. Subito dopo però il Proxy riporta il buffer al livello normale. Vediamo ora come il Proxy gestisce le previsione del Client:

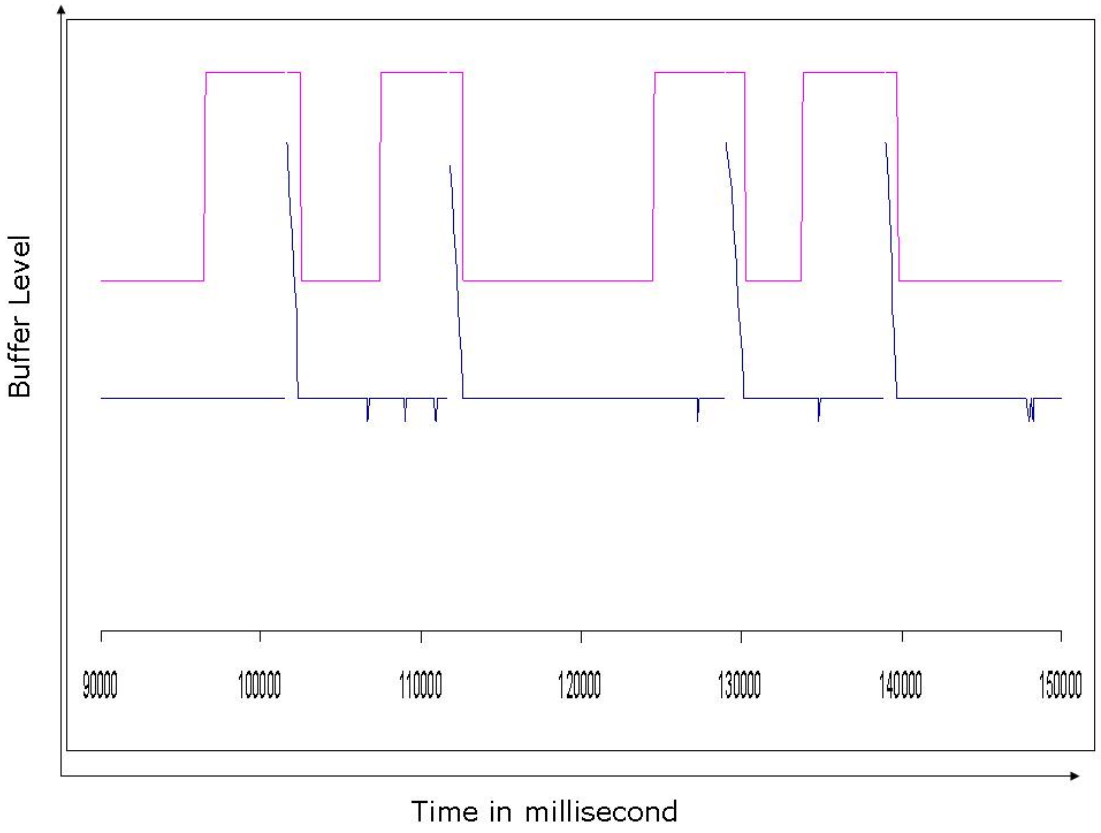


Fig 6.10 – Dimensione della reiwnd window durante il servizio

Questo grafico mette mostra nella linea inferiore lo usage buffer mentre quella superiore rappresenta il total buffer. La differenza tra i due è la rewind window. Quando il Client passa da uno stato di bassa probabilità di handoff ad una di alta probabilità invia uuna richiesta al Proxy che provvede ad allargare la rewind window per avere frame sufficienti a servire la richiesta del Client. Appena il Client effettua l'handoff entra subito in uno stato di bassa probabilità come è evidenziato dalla riduzione della rewind window subito dopo che la richiesta è stata servita (laddove lo usage buffer presenta delle creste). In questo caso tutte le richieste sono state servite correttamente e lo si può notare dal fatto che le punte più alte dello usage buffer sono sempre più inferiori del total buffer. Ovviamente il buffer usage non può superare in valore quello total, ma quando lo raggiunge significa che abbiamo utilizzato tutti i frame a nostra disposizione e quindi c'è probabilità che la richiesta non sia stata servita pienamente.
Vediamo ora di mettere in luce la differenza che si evidenzia servendo schede diverse

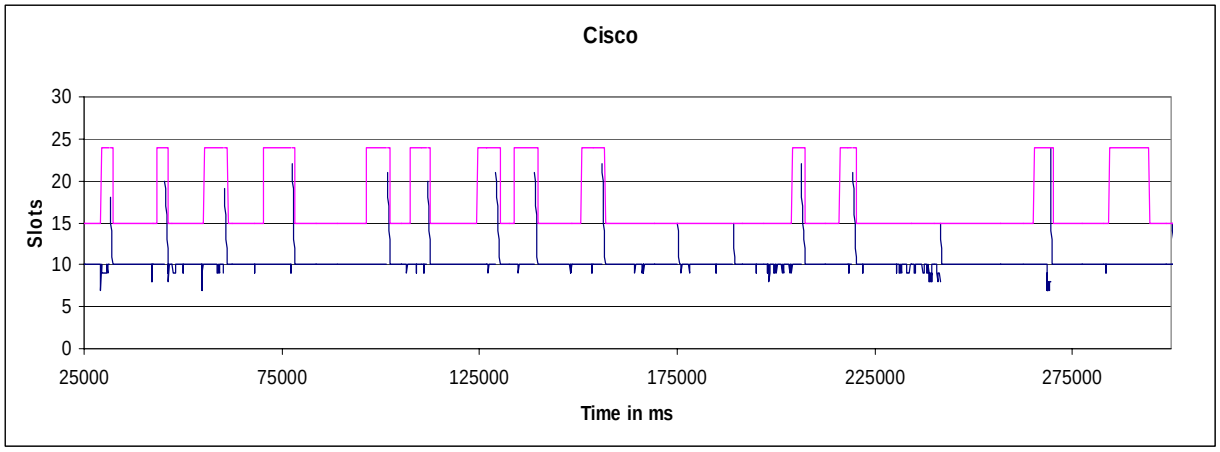


Fig 6.11 – Proxy buffer con scheda Cisco like

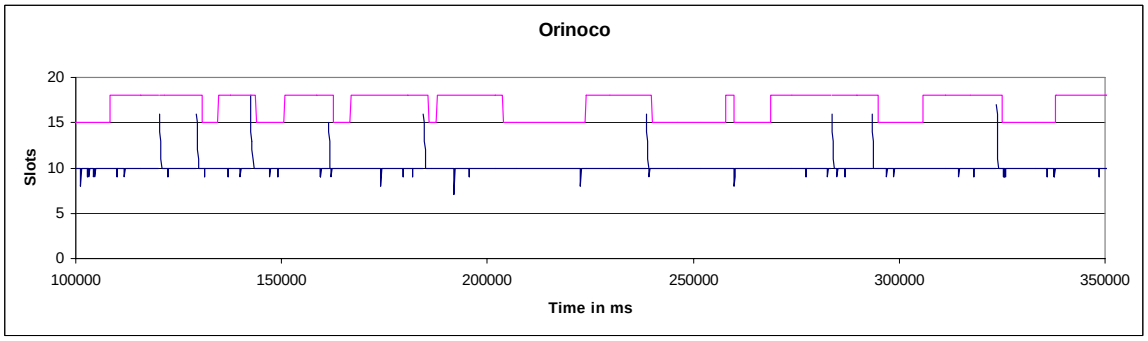


Fig 6.12 – Proxy buffer con scheda Orinoco like

Il primo grafico in figura 6.11 rappresenta il buffer di un Proxy che serve un Client che dotata di una scheda Cisco e che quindi adotta una strategia Soft Proactive mentre il secondo, in figura 6.12, riguarda una scheda Orinoco che utilizza una strategia Hard Proactive. La principale differenza tra i due è la dimensione della rewind window che per il caso Cisco è maggiore. Questo, lo ricordo ,è dovuto ad una conoscenza del Proxy che, riconoscendo la scheda, può configurare la dimensione della finestra in maniera precisa e fissa.
Quando invece il Proxy non riconosce la scheda deve partire con una configurazione di default. L'obiettivo è servire il Client al meglio ma senza utilizzare risorse inutili. Il Proxy quindi tenterà di imparare le necessità del Client imparando dalle sue richieste.

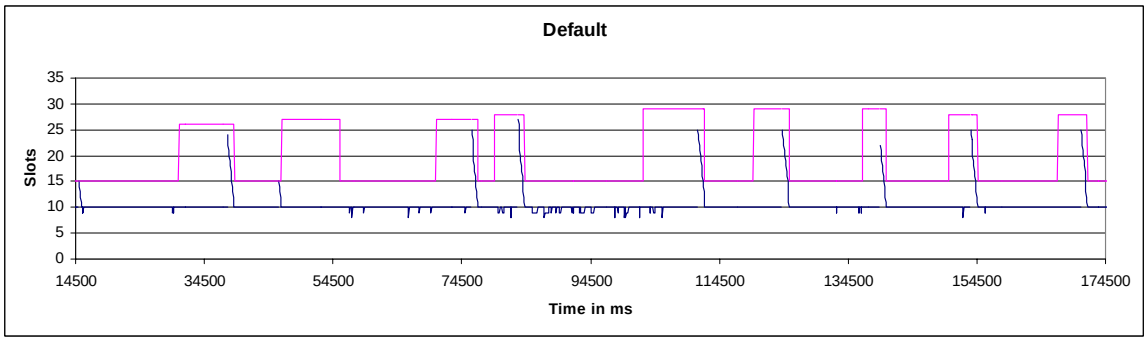


Fig 6.13 – Proxy buffer adattivo con scheda non conosciuta

Come si nota dal grafico, che riporta il valore del total buffer e dello usage buffer, il valore della rewind window non è fisso ma varia:la prima e la terza richiesta vengono soddisfatte ma con un margine minimo. Questo provoca l'aumento della dimensione della rewind window. In seguito unna richieste di richieste minori fanno si che questa dimensione si riduca per non sprecare memoria inutilmente.

6.2.4 - Indicatori di prestazione

Abbiamo messo in evidenza la validità della soluzione proposta ma abbiamo tralasciato il costo della stessa. Introdurre un Proxy come passo intermedio comporta un costo aggiuntivo in termini di memoria che si deve cercare di minimizzare.
Esistono ovviamente dei costi minimi non riducibili dovuti dovuti all'erogazione del servizio stesso. Elementi che introducono uno spreco sono:

- Predizioni sbagliate: ovvero quando il Client prevede un handoff che poi non avviene. Questo causa la richiesta da parte del Proxy di memoria per aumentare la dimensione della rewind window.
- Anticipo della previsione: riempire la rewind window richiede un certo numero di secondi. E' quindi necessario che la previsione arrivi sufficientemente in anticipo da permettere il riempimento della finestra di rewind. Se però arriva troppo in anticipo il proxy occupa memoria inutilmente per un certo lasso di tempo.

Servire correttamente ogni richiesta sarebbe possibile se aumentassimo di molto il numero delle risorse occupate. Una soluzione troppo costosa non è però una buona soluzione. Bisogna quindi cercare di minimizzare i tempi di previsione ed il numero di previsioni sbagliare senza aumentare troppo il numero delle richieste non soddisfatte.
Per valutare le prestazioni si sono così introdotti alcuni indicatori che ci aiutassero a comprendere dove correggere:

Efficienza:numero degli handoff verificatisi/numero di quelli previsti
Efficacia:numero delle richieste servite correttamente/numero degli handoff verificatisi
Errore:numero degli handoff non previsti/numero handoff totali.
Tempo medi di previsione:la media dell'anticipo con cui viene fatta la previsione.

Le simulazioni sono state fatte tenendo conto di questi risultati cercando il miglior compromesso tra questi valori. Farò ora presentazioni dei vari risultati in diverse occasioni per meglio comprendere le variazioni introdotte dal cambiamento dei parametri e commendandole opportunamente.

6.2.5 – Simulazioni

Come detto , solo alcuni parametri necessitavano di dover essere calibrati per via sperimentale attraverso simulazioni con configurazioni differenti. I valori del Proxy, ad esempio, riguardavano unicamente la durata massima di un handoff ed è quindi stato possibile configurarli in maniera indipendenti. L'Efficacia è un indice che tiene conto

soprattutto di come il Proxy risponde alle richieste del Client mentre gli altri indici riguardano soprattutto la prestazione della predizione.
L'Efficacia è quindi un indice indipendente dagli che invece sono correlati tra di loro. Tra i parametri della predizione ci siamo concentrati su due di essi, che particolarmente influenzano la predizione.
Chiaramente si è cercato un tradoff tra questi valori ed il tempo medio di previsione: infatti si è notato che mentre si cerca di migliorare l'efficienza il tempo di previsione sale.
Entrambi i parametri servono per misurare se stiamo sprecando memoria o meno: l'efficacia ci dice quante volte facciamo una previsione corretta, ed è chiaro che una previsione sbagliata comporta un'occupazione inutile di memoria. Allo stesso modo una previsione troppo in anticipo fa impegnare risorse per più tempo del necessario.
Il primo parametro è il "numero dei campioni per il grey model": identifica la storia passata conservata. Un alto numero di campioni renderà la previsione più lenta ma meno soggetta al rumore del segnale. Un basso numero di campioni rende la previsione molto più reattiva ma può causare un maggior numero degli handoff.
L'altro parametro, che chiameremo "finestra di previsione", è la differenza tra la previsione dell'RSSI dell'AP corrente e l'RSSI dell'AP successivo tale da far scattare la predizione di un cambiamento: se la prediciamo che il valore dell'RSSI dell'AP successivo supera di un certo valore quello dell'AP attuale allora consideriamo possibile l'handoff.
Tale valore sarà però spesso negativo:
Finestra di previsione = RSSI dell'AP corrente – RSSI dell'AP successivo. Dato che la previsione viene effettuata quando il secondo è di valore più elevato del primo, il valore di tale finestra può essere negativo.
Vediamo ora qualche risultato commentandolo per rendere più facile il confronto.

Prova 1:

Dati
Scheda:Cisco
Numero di campioni:8
Finestra di previsione:-3

Indici
Efficienza: 85.462555%
Efficacia: 96.969696%
Errore: 10.0%
Tempo medio di previsione: 7074.459ms

Noteremo, nel corso delle prove, come l'efficacia non vari in maniera significativa tra le varie prove.
Difficile è commentare questi dati senza riferimenti. Notiamo però che l'Efficienza è forse più bassa di quanto desiderabile e l'errore è al limite della tolleranza generalmente accettata.

Prova 2:

Dati

Scheda:Cisco
Numero di campioni:13
Finestra di previsione:-3

Indici
Efficienza: 99.45055%
Efficacia 97.31183%
Errore: 15.094339%
Tempo medio di previsione: 6288.40625ms

Abbiamo aumentato di molto il numero dei campioni del Grey model per sottolineare la variazione. Aumentando questo parametro otterremo sicuramente previsioni più lente e sicure, in quanto maggiormente analizziamo il passato del Client con il rischio però di mancare diverse previsioni. Notiamo infatti un aumento consistente dell'Efficienza dovuto al fatto che quasi tutte le previsioni sono corrette. Notiamo inoltre l'abbassamento del tempo di previsione di quasi un secondo. Di contro osserviamo come l'errore aumenti considerevolmente, proprio perché le previsioni possono risultare non tempestive. Vediamo ora cosa succede variando l'altro parametro.

Prova 3:
Dati
Scheda:Cisco
Numero di campioni:8
Finestra di previsione:-6

Indici
Efficienza: 98.54369%
Efficacia 96.55172%
Errore: 0.4901961%
Tempo medio di previsione 8623.995ms

Facciamo il paragone con la Prova 1: abbiamo aumentato la finestra di previsione, quindi ci aspettiamo di essere più tempestivi nella predizione degli spostamenti. Notiamo infatti un aumento notevole dell'Efficienza, anche se aspettavamo di avere molte più previsioni non vere.. Notiamo anche il drastico abbattimento dell'errore. L'unica pecca sembra essere l'aumento del tempo di previsione che peggiora le prestazioni in termini di memoria occupata. Questo dato diventa rilevante quando un filmato occupa molta memoria.

Prova 4:
Dati
Scheda:Cisco
Numero di campioni:10
Finestra di previsione:-3

Indici
Efficienza: 87.65958%
Efficacia 97.409325%
Errore: 8.530806%
Tempo medio di previsione 6121.51282051282ms

Questi sono risultati ottenuto utilizzando valori intermedi per entrambi i parametri. L'Efficacia rimane alta, ma abbiamo già detto che è un parametro non fortemente dipendente dagli altri. L'errore rimane entro livelli accettabili e l'efficienza rimane comunque buona. Il tempo di previsione è tra quelli minimi ottenuti
Sperimentalmente si è visto che è difficile ridurre molto il tempo di previsione senza alterare troppo gli altri valori e questo è uno dei migliori compromessi ottenibili.

6.2.6 – Conclusioni

Alla fine di questo lavoro possiamo trarre delle conclusioni.
Il sistema così progettato funziona, nel senso che raggiunge lo scopo che ci eravamo prefissati, ovvero la gestione dell'handoff.
Il simulatore riesce a farci sperimentare queste applicazioni in massa e senza molte risorse disponibili ed è quindi un utile strumento di sviluppo. Inoltre le applicazioni possono essere scritte senza considerare la simulazione stessa (a meno di una riga di codice iniziale).
Gli indici hanno evidenziato che si può trovare un buon compromesso tra correttezza della soluzione e costo in risorse.
In conclusione possiamo dire che la linea intrapresa è promettente anche se certamente può essere migliorata con un lavoro di rifinitura ed ampliata per essere più completa.

6.3 - Sviluppi futuri

Dopo aver illustrato il progetto ed i risultati vediamo i possibili sviluppi aperti da questo lavoro: ogni tesi di questo tipo parte da uno state dell'arte e lo modifica dandogli una direzione ed aprendo nuovi scenari possibili. Vale la pena quindi tentare di andare avanti con lo sguardo per comprendere le direzioni percorribili.
Dividerò lo scenario per argomenti e non per divisione del progetto poiché certe parti interessano sia il lato Client, sia il lato Proxy sia il Simulatore

6.3.1 – Conoscenza sulle schede wireless

Un passo senza dubbio importante sarebbe sviluppare una conoscenza più approfondita sul comportamento delle schede wireless. Questo interessa le applicazioni Client , Proxy ed il Simulatore. Al Client una conoscenza più specifica sarebbe utile per poter migliorare i meccanismi di previsione legati alla scheda stessa: non possiamo pretendere

di conoscere le politiche esatte dei ogni scheda perché, giustamente, alcuni venditori non le rendono note dato che spesso costituisce il lato caratteristico della scheda stessa. E' però possibile acquisire una conoscenza "empirica" che possa preparare il Client a gestire gli handoff. Sarebbe possibile configurare la predizione in maniera più precisa, cosa che aiuterebbe il Proxy a gestire meglio le richieste dovute ad handoff, ma anche gestire meglio le risorse sul Client stesso una volta avuta la stima della durata stimata degli handoff.
Stessa conoscenza sarebbe applicabile alla simulazione per emulare il comportamento delle stesse schede e così poter testare l'applicazione Client: la procedura sarebbe quindi quella di acquisire la conoscenza della scheda per simularla, ed a quel punto modificare l'applicazione Client usando la Simulazione. Ricordo inoltre che utilizzando i file di Log è possibile una simulazione molto verosimile anche se impostata su una singola esperienza sempre uguale a se stessa.
Dal lato Proxy questa conoscenza aiuterebbe certamente a non sprecare risorse, potendo predisporre di partenza un numero adeguato di frame di riserva, minimizzando errori e costi. Il risultato sarebbe quello di un archivio contenente i dati principali di ogni scheda conosciuta da consultare al momento del bisogno. Sarebbe anche possibile mantenere questo archivio unicamente lato Proxy ed a quel punto il Client chiederebbe le informazioni direttamente al Proxy una volta connesso e specificato la scheda montata sulla macchina ospite.

6.3.2 – Adattamento alle risorse

Abbiamo visto come il sistema possa essere predisposto per adeguarsi alle risorse effettivamente alle risorse disponibili al Client. Il Client dovrebbe quindi acquisire più informazioni ad esempio controllando la CPU e la memoria disponibile. Inoltre potrebbe chiedere al Proxy un filmato che utilizzi i codec nativi già presenti nel sistema. Il Proxy, da parte sua, potrebbe mettere a disposizione una serie di versioni del filmato stesso oppure occuparsi lui stesso della conversione da un codifica ad un'altra. Anche durante la riproduzione il Client potrebbe decidere che sta impiegando troppe risorse e che quindi è più utile ridurre la qualità del filmato. L'adattamento dinamico, durante la sessione, alle caratteristiche del Client è un elemento che si aggiungerebbe all'adattamento statico in fasi di associazione del Client.

6.3.3 – Previsione

La previsione è un tema centrale per l'ottimizzazione nell'uso delle risorse lato Proxy e quindi è conveniente investire sul miglioramento di questo sistema. Come abbiamo detto un passo importante sarebbe quello di ottenere una conoscenza più approfondita sulle schede e soprattutto su una varietà di schede più ampia. Nel capitolo sulle reti 802.11 si è messo in evidenza come cambiare AP a volte può voler dire cambiare sottorete. Una handoff tra AP di una stessa sottorete oppure uno tra AP di sottoreti diverse non hanno la stessa durata: la riassociazione in una diversa sottorete comporta delle operazioni aggiuntive che allungano la durata dell'handoff. Il Client potrebbe richiedere al Proxy

informazioni sugli AP da lui visibili per capire di che tipo sia l'handoff previsto, oppure passare al Proxy, oltre alla previsione, l'indicazione dell'AP previsto ed il Proxy regolarsi sui frame da conservare in base alla sua conoscenza della posizione degli AP.
Attualmente viene considerato unicamente la potenza del segnale degli AP per effettuare considerazioni su un possibile cambio AP e senza dubbio è uno dei parametri più importanti. Altre situazioni potrebbero però far tenere in considerazione altri fattori come la perdita dei messaggi di ACK spediti a livello MAC

6.3.4 – Integrazione del Proxy

Come abbiamo detto il progetto non è stato ideato a se stante ma si colloca all'interno di un progetto più ampio che mira alla creazione di una infrastruttura che si occupi delle necessità dei Client mobili. Al momento il Proxy non è integrato in tale sistema anche se ideato per esserne parte. Il prossimo passo è quindi quello di integrare questo servizio all'interno di un Proxy più ampio e generale che segua il Client in tutte le sue richieste e nei suoi spostamenti. Finché il Client rimane connesso con AP appartenenti alla stessa sottorete non c'è alcuna necessità di spostare il processo Proxy, ma quando avviene un cambiamento di contesto che porta , ad esempio, il Client a cambiare AP sarebbe conveniente riuscire spostare anche il processo Proxy. Per questo il Proxy progettato in questa tesi dovrebbe essere reso "portabile" da una macchina all'altra.

6.3.5 – La simulazione

La simulazione attualmente prende in considerazione la potenza dei segnali degli AP e l'interruzione della connessione durante un AP fatta con connessione Connectionless. E' vero che le connessioni di tipo TCP sono tolleranti a questo tipo di situazioni perché prevedono un meccanismo di ritrasmissione e che in questo contesto non erano di nostro interesse, ma per rendere più completo l'ambiente simulato è utile costruire classi Wrapper che sostituiscano le normali Socket Java con classi create da noi seguendo il medesimo meccanismo utilizzato per le Socket UDP.
Questo comporta difficoltà maggiore perché il protocollo TCP è più complesso di quello UDP ma comporterebbe di poter controllare per intero tutte le connessioni aperte e quindi rendere totalmente invisibile alle applicazioni il fatto di trovarsi all'interno di un contesto simulato, anche quando utilizzassero connessioni TCP.
Il Simulatore al momento calcola gli RSSI con una funzione che tiene conto della potenza del segnale d'origine e della distanza dell'utente, introducendo un rumore gaussiano sopra il segnale per simulare i normali disturbi.
Riflessioni, ostacoli ed interferenze peggiorano questa semplice schematizzazione del sistema. Al momento il Simulatore non ha ne la pretesa ne la funzione di offrire un contesto realistico in tutto e per tutto ma potrebbe essere interessante chiedere la collaborazioni di esperti nelle telecomunicazioni per migliorare la fedeltà ai contesti reali.

Conclusioni

Nel corso di questa tesi sono stati approfonditi due temi principali, ovvero la progettazione di una infrastruttura che garantisse la continuità di servizio di applicazioni multimediali in ambienti wireless e lo sviluppo di un ambiente simulato con l'obiettivo di provare la validità dell'infrastruttura ideata e di facilitare il lavoro del progettista.

Il primo risultato importante è di aver risolto il problema delle interruzioni video durante gli handoff introducendo un'architettura proxy-based al posto della classica Client-Server, che riuscisse a ritrasmettere i dati persi dal Client durante le sue disconnessioni rendendogli invisibile il passaggio da un AP all'altro. Un altro obiettivo raggiunto è l'integrazione di un'entità intelligente che, raccogliendo opportunamente informazioni sulla macchina ospite, adatta l'applicazione alle esigenze specifiche del dispositivo. Il sistema di previsione ha messo in luce la possibilità di gestire le ritrasmissioni di dati senza occupare inutilmente risorse quando non necessario e senza peggiorare sensibilmente la soluzione introdotta.
Parallalemante si è riusciti a creare un ambiente simulato che ha permesso il test dell'infrastruttura proxy-based su scenari eterogenei e con device diverse tra loro. La struttura del simulatore inoltre permette di poter effettuare prove utilizzando diverse macchine all'interno di una LAN decentralizzando il peso computazionale della simulazione stessa, rendendo così possibile la valutazione di un contesto con un grande numero di utenti su macchine diverse. Un altro aspetto raggiunto emulando il comportamento delle schede wireless agli occhi dell'applicazione multimediale è quello di consentire al progettista di scrivere il codice dell'applicazione direttamente nella forma finale, senza includere parti relative alla simulazione dell'applicazione stessa.

Nell'ipotesi di avere la conoscenza dei dispositivi wireless in dotazione all'utente, il meccanismo di previsione funziona bene e senza lo spreco eccessivo di risorse da parte del Proxy. L'infrastruttura prevede di adattarsi anche ad un Client sconosciuto, ma questo ovviamente comporta un dispendio in termini di risorse occupate. Si è inoltre constatato come sia difficile ottenere le informazioni di contesto del Client sia per la molteplicità dei Client stessi, sia per la carenza di API espressive che offrano strumenti per l'interrogazione dell'hardware di cui il Client è dotato Questa mancanza di conoscenza si riflette parallelamente sul numero di schede emulabili e quindi sulla possibilità di testare la validità della applicazioni. L'applicazione che simula l'ambiente wireless al momento non prende in considerazione il deterioramento del segnale all'allontanarsi di un utente dal proprio AP: sfruttando connessioni locali i messaggi inviati arrivano sempre con successo e velocità ai destinatari, cosa non vera in un contesto reale.

La direzione intrapresa ha comunque dimostrato di essere valida, raggiungendo gli obiettivi che ci si era prefissati e fornendo anche strumenti per ottenere il compromesso voluto tra efficacia dell'infrastruttura e costo in termini di risorse. L'ambiente simulato ha fornito un aiuto essenziale per poter sperimentare il progetto in tempi contenuti e senza la necessità di avere disponibile un numero elevato di utenti ed una rete wireless

adatta agli esprimenti. Questo apre la strada a nuovi sviluppi tesi a migliorare l'infrastruttura progettata.

Dai dati è risultato evidente che il meccanismo di previsione funziona tanto meglio quanto più approfondita è la conoscenza della macchina dell'utente. La previsione è inoltre il componente che rende accettabile l'introduzione di un Proxy tra Server e Client perché offre la possibilità di non sprecare inutilmente risorse.
Una direzione importante sarà quindi quella di reperire più informazione riguardo le schede in commercio e le loro strategie di handoff in modo da migliorare le prestazioni della predizione. L'ambiente emulato si è poi rivelato uno strumento molto valido nello sviluppo dell'infrastruttura ma presenta ancora limiti che sarebbe auspicabile superare. L'evoluzione più desiderabile è quella di introdurre gli elementi di disturbo che ancora non vengono considerati come la perdita dei dati lontano dagli AP ed i limiti di banda che sono caratteristici delle reti wireless.

Appendice A – Standard 802.11

Designation	Title	Description	Status
802.11	Original WLAN in 2.4GHz band	The original base document; included raw data rates of 1 and 2 Mbits/second using either frequency hopping or direct sequence spread spectrum (DS/SS) at 2.4GHz or infrared.	Ratified and published
802.11a	54Mbps WLAN in 5GHz band	Amendment that added the use of higher bit rates using orthogonal frequency division multiplexing (OFDM) at the 5 GHz band	Ratified and published
802.11b	11Mbps WLAN in 2.4GHz band	Amendment that added the use of higher bit rates (up to a raw data rate of 11Mbits/second) to the original DS/SS 2.4GHz radio, but did not use OFDM.	Ratified and published
802.11c	MAC bridging supplement	This group developed material that was incorporated into IEEE 802.1D; no changes were made to IEEE 802.11	Material was incorporated into 802.1D and ratified.
802.11d	Multiple regulatory domain support (international roaming)	Added support (MIBs, beacon elements, country definitions) to support more country regulatory specifications. Also added support that would allow the AP to notify the client which country to support.	Ratified and published
802.11e	Quality of Service	Two competing schemes for implementing QoS were proposed; neither proposal could unilaterally be accepted, so the group accepted both, and merged the two proposals into one standard.	At the Sponsor Ballot phase
802.11F	Inter-AP protocol	Using a central coordinating entity, APs could find out about each other, and then open up a secure tunnel among themselves. Subsequent investigation of the protocol has possibly shown that it is insecure. Since IEEE 802.11 thought they were limited to the MAC and PHY levels, this protocol at the	Ratified on a Trial-Use basis as a Recommended Practice; due to be withdrawn June 2005

		levels above the MAC level could only be a recommended practice.	
802.11g	54Mbps WLAN in 2.4GHz band	Adding the OFDM modulation of 802.11a to the 2.4GHz band, extending the raw data rate from the maximum 11Mbits/s of 802.11b to 54Mbits/s.	Ratified and published
802.11h	Spectrum Managed 802.11a (Dynamic Freq Selection, Tx Power Control)	Several regulatory domains already have existing applications that use frequencies in the 802.11a frequency bands. These applications could not be moved to another frequency, neither could they be interfered with. 802.11h specified behaviors for APs and clients to dynamically avoid the specific frequencies in the 802.11a band that are in use by the existing applications.	Ratified and published
802.11i	Security	Defined an interim encryption protocol to patch existing WEP equipment, and defined an encryption protocol based on AES for new equipment. Also defined the usage of authentication using the 802.1X port-based authentication framework, and recommended that client-network authentication use the EAP authentication protocol framework with EAP types.	Ratified and published
802.11j	Japan regulatory support	Specified the use of the 4.9-5GHz frequency band as defined by the Japanese regulatory agency.	Ratified and published
802.11k	Radio resource measurements (support of external RF tools used in WLAN mgmt)	Specifying additional MIBs that will expose various radio and environmental measurements. The goal is to have a standard set of measurements available that network management tools can use.	Failed the first letter ballot
802.11m	Maintenance & interpretation of IEEE 802.11 standards	This group is rolling together 802.11, 802.11a, 802.11b, 802.11d, 802.11g, 802.11h, 802.11i, and 802.11j into a single specification. The group also does interpretation requests on an on-going basis: requests for interpretation of ambiguous/puzzling aspects of the standard may be submitted to this group, which will then attempt to answer the request.	Working on combined edition of 802.11 standards
802.11n	High throughput PHY & MAC at 2.4GHz	This group was chartered to develop a high throughput (data rates greater than 100Mbit/s) in the 2.4GHz band.	Going through down-select process on

		Proposed solutions all use Multiple Input/Multiple Output (MIMO) technology, where multiple spatially separated antennas are used per station.	submissions
802.11p	Wireless Access for Vehicular Environments (WAVE)	This group is taking a specification that has already been written and is attempting to get it accepted by IEEE 802.11. This existing specification is somewhat based on 802.11a, and is designed to allow wireless networking between vehicles and the environment the vehicles are in. Possible uses are updating vehicles with road status, facilitating vehicle telematics, and vehicle-road interactions.	Just started work
802.11r	Fast BSS Transition	Moving a client from one AP to another used to be trivial to do, and could be done quickly. With the addition of QoS and security to the standard (along with future enhancements), the process of moving from one AP to another has become more complex, and takes more time to accomplish. This group is attempting to speed up the process of moving from one AP to another, with the goal of minimizing the gap in data connectivity.	Selecting from submissions
802.11s	ESS Mesh Networking	This group is developing a method to dynamically mesh together APs into a single wireless ESS. By meshing together APs, the APs do not need to be hard wired to the network infrastructure and by dynamically generating the mesh, APs can drop out or join the mesh without requiring manual reconfiguration. The ESS mesh should be completely transparent to a client; it will not know if the AP that it is associated with is a traditional hard-wired AP or a meshed AP.	Developing documents for selection process
802.11u	Wireless Interface to External Networks	This group is attempting to define various interfaces in 802.11 to facilitate the interoperation between 802.11 and other types of wired and wireless networks. Potential networks are the various flavors of cellular data, as well as other wired networks. Interoperation issues such as billing and handoff are planned to be addressed.	Just started work

802.11v	Wireless Network Management	This group is developing a way for APs to actively manage the clients that are associated with it. This group is a follow up to TGf, and is hoping to use the information exposed by TGf to facilitate active management. This task is somewhat of a change in philosophy for 802.11, in that the original idea in 802.11 was that the clients control the association process.	Just started work
802.11T	Wireless Performance Prediction	This group is trying to develop a Recommended Practice (which means that it will not become part of the 802.11 standard) that will define a standard method for measuring the air characteristics of 802.11-based units. The hope is that using standard measurement techniques will allow units from different manufacturers to be easily compared for air characteristics, and that predictions could be made as to the performance of 802.11 units in installations.	Just started work

Bibliografia

[80211WN] Gast M., " 802.11 Wireless Networks: The Definitive Guide", April 2002 , ISBN 0-596-00183-5

[80211WL] Roshan R., Leary J., " 802.11 Wireless LAN Fundamentals", December 23, 2003, ISBN 1-58705-077-3

[80211FA] O'Hara B, "802.11F Inter-AP Protocol",
si veda
http://www.ieee802.org/21/archived_docs/Documents/OtherDocuments/Handoff_O'Hara.pdf

[80211FS] "IEEE Trial-Use Recommended Practice for Multi-Vendor Access Point Interoperability via an Inter-Access Point Protocol Across Distribution Systems Supporting IEEE 802.11™ Operation", July 14, 2003
Si veda http://systems.cs.colorado.edu/downloads/802-standards/ieee-802.11f.pdf

[JAVACD] Dabbs Halloway S., "Component Development for the Java™ Platform", December 14, 2001, ISBN 0-201-75306-5

[RFC1889] RTP: A Transport Protocol for Real-Time Applications. RFC1889,
si veda http://www.ietf.org/rfc/rfc1889.txt

[JMFPG] Java Media Framework, Programmers Guide:
http://java.sun.com/products/java-media/jmf/2.1.1/specdownload.html

[JMFH] Java Media Framework home page: http://java.sun.com/products/javamedia/jmf/

[ProMig] Bellavista P, Corradi A, Foschini L, "Java-based Proactive Buffering for Multimedia Streaming Continuity in the Wireless Internet", *IEEE WoWMoM*,2005

[QoSAda] Chan J., Zhou S., Seneviratne A., "A QOS ADAPTIVE MOBILITY PREDICTION SCHEME FOR WIRELESS NETWORKS*", Global Telecommunications Conference, 1998. GLOBECOM 98. The Bridge to Global Integration. IEEE Volume 3*,1998.

[QoSCap] Chan J., De Silva R., Zhou S., Seneviratne A., "A Framework for Mobile Wireless Networks with an Adaptive QoS Capability", *MoMuC98* ,Berlin, 1998

[MUM] Bellavista P, Corradi A, Foschini L, "MUM: a Middleware for the Provisioning of Continuous Services to Mobile Users", *IEEE International Symposium on Computers and Communications* (*ISCC'04*), Alexandria, Egypt, June 29-31, 2004

[StrPer] Kuang T., Williamson C., "RealMedia Streaming Performance on an IEEE 802.11b Wireless LAN", *Proceedings of IASTED Wireless and Optical Communications (WOC) Conference*, Banff, AB, Canada, , July 2002

[VoDSup] Bellavista P, Corradi A, Foschini L, "How to Support Internet-based Distribution of Video on Demand to Portable Devices", *IEEE 7th Int. Symp. On Computers and Communications (ISCC'02)*, 2002.

[PreRSSI] Bellavista P, Corradi A, Giannelli C., "Predicted RSSI-based Adaptive Buffering for Continuous Services in the Wireless Internet", *The 2005 International Conference on High Performance Computing and Communications* (HPCC-05), Sorrento, Italy, September 2005.

[UbiPro] Bellavista P, Corradi A, Stefanelli C., "The Ubiquitous Provisioning of Internet Services to Portable Devices", *IEEE Pervasive Computing*, Vol. 1, No. 3, 2002.

[FastHO] Dutta A., Madhani S., Chen W., "Fast-handoff Schemes for Application Layer Mobility Management",*15th IEEE International Symposium on Volume 3,* Sept. 2004

[AnHO] Mishra A., Shin M., Arbaugh W., "An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process",*ACM SIGCOMM Computer Communication Review (ACM CCR)*, Volume 33, Issue 2, April 2003

[SLHO] Cui Y., Nahrstedt K., Xu D., "Seamless User-level Handoff in Ubiquitous Multimedia Service Delivery", *Multimedia Tools and Applications Journal, Special Issue on Mobile Multimedia and Communications and m-Commerce,* vol. 22, pp. 137-170, Kluwer, 2004

[VoDSer] Bruneo D., Villari M., Zaia A., Puliafito A., "VOD services for mobile wireless devices", *Computers and Communication*, 2003. (ISCC 2003)

[CodMob] Fuggetta A., Pietro Picco G., Vigna G., "Understanding Code Mobility", *IEEE Transactions on Software Engineering*, Vol. 24, No. 5, 1998.

[MoAgMi] Bellavista P, Corradi A, Stefanelli C., "Mobile Agent Middleware for Mobile Computing" , *IEEE Computer*, Vol. 34, No. 3, 2001.